

Package: cmrlifecycle (via r-universe)

October 15, 2026

Type Package

Title Audit the Revision and Production Lifecycle of NASA CMR Granules

Version 0.0.1

Description Queries NASA's Common Metadata Repository (CMR) granule search API and distinguishes CMR metadata revision churn from actual changes to the underlying science data file, using UMM-JSON fields such as revision-id, ProviderDates, and DataGranule ProductionDateTime. Built to support decisions about when a downstream mirror (for example a Cloud-Optimized GeoTIFF conversion published to source.coop) needs to be regenerated, rather than reacting to every CMR metadata touch.

License MIT + file LICENSE

URL <https://hypertidy.github.io/cmrlifecycle/>

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports dplyr, jsonlite, tibble, utils

Suggests knitr, pkgdown, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Language en-US

Repository <https://hypertidy.r-universe.dev>

Date/Publication 2026-09-02 07:16:13 UTC

RemoteUrl <https://github.com/hypertidy/cmrlifecycle>

RemoteRef HEAD

RemoteSha 28cb641bfeb2d52dbb2de3b432101c4e04f3b5e8

Contents

cmr_all_collections	2
cmr_audit_revisions	3
cmr_collection_liveness	5
cmr_find_collections	6
cmr_find_granules	8
plot_cmr_lifecycle	9
Index	10

cmr_all_collections	<i>Fetch minimal metadata for every CMR collection matching a search</i>
---------------------	--

Description

Pages through [cmr_find_collections\(\)](#) with page_num, pausing delay seconds between requests, until every matching collection has been fetched (or max_pages is hit). Useful for a full catalog sweep – e.g. [cmr_all_collections\(\)](#) with no filters returns minimal metadata for all ~45,000 CMR collections, in ~23 requests at the default page_size.

Usage

```
cmr_all_collections(  
  keyword = NULL,  
  short_name = NULL,  
  provider = NULL,  
  page_size = 2000,  
  delay = 0.3,  
  abstract = FALSE,  
  max_pages = Inf,  
  granule_count = TRUE  
)
```

Arguments

keyword, short_name, provider	As in cmr_find_collections() , all optional here. If both keyword and short_name are NULL (the default), keyword = "" is used, which CMR treats as "match everything" – i.e. the default sweeps the <i>entire</i> CMR catalog.
page_size	Collections per request (CMR max, and the default here, is 2000).
delay	Seconds to pause between requests. Not a documented CMR rate limit – just a politeness default for an unauthenticated, multi-request sweep. Set to 0 to disable.
abstract	Keep the abstract text column. Default FALSE: for a full-catalog sweep the abstracts add real memory/parse cost for a "which collections exist" use case; the network request still fetches them (CMR doesn't support field projection here),

	this just drops them from the result. Pass TRUE to keep them, as <code>cmr_find_collections()</code> does.
<code>max_pages</code>	Safety cap on the number of requests issued. Inf by default; set lower if you're experimenting with a filter and want to sample rather than fetch everything.
<code>granule_count</code>	As in <code>cmr_find_collections()</code> . Default TRUE, but note this means <i>every</i> page issues two requests (one for collection metadata, one for granule counts) instead of one – for a full-catalog sweep that's ~46 requests rather than ~23. Set FALSE for a faster sweep if you don't need it.

Details

CMR's classic `page_num/page_size` paging is only documented to work up to roughly 1,000,000 records (`page_num * page_size`); this function stops with a warning if it would exceed that, rather than silently looping. That ceiling is far above the current collection count, so it's a safeguard, not an expected code path.

Value

A `tibble::tibble()`, same columns as `cmr_find_collections()` (minus `abstract` unless `abstract = TRUE`), one row per collection. Reports progress via `message()` after each page.

See Also

Other collection functions: `cmr_collection_liveness()`, `cmr_find_collections()`

Examples

```
# every CMR collection, minimal columns (~23 requests, a few seconds
# of `delay` pauses on top of network time)

all_colls <- cmr_all_collections()

# narrower sweeps are quick
cmr_all_collections(provider = "LAADS")
```

<code>cmr_audit_revisions</code>	<i>Audit the CMR revision/production lifecycle of a collection's granules.</i>
----------------------------------	--

Description

Audit the CMR revision/production lifecycle of a collection's granules.

Usage

```
cmr_audit_revisions(
  short_name,
  from = as.Date("2024-01-01"),
  to = Sys.Date(),
  provider = NULL,
  gran_date_fun = .default_gran_date_fun,
  bounding_box = NULL,
  polygon = NULL
)
```

Arguments

<code>short_name</code>	CMR collection short name
<code>from, to</code>	Date range to sweep, chunked into 3-month CMR requests
<code>provider</code>	Optional CMR provider id (see <code>cmr_find_granules()</code>)
<code>gran_date_fun</code>	Function of the assembled tibble (with columns <code>granule_ur</code> , <code>temporal_start</code> , <code>data_url</code> , ...) that returns the granule's nominal date. Defaults to parsing a leading YYYYMMDD out of <code>granule_ur</code> , which holds for GHRSSST-style products (MUR, AVHRR_OI, ...); for collections with opaque GranuleURs, pass e.g. <code>function(df) as.Date(df\$temporal_start)</code> .
<code>bounding_box, polygon</code>	Optional spatial filters, passed through to every chunked <code>cmr_find_granules()</code> call – see <code>cmr_find_granules()</code> for their formats. Mutually exclusive.

Value

A `tibble::tibble()` with one row per granule (deduplicated across chunk boundaries), including `gran_date` and the lag columns described in this file's header comment. Warns if `gran_date` comes back entirely NA (see the MOD021KM example in the README).

See Also

Other granule functions: `cmr_find_granules()`, `plot_cmr_lifecycle()`

Examples

```
rev <- cmr_audit_revisions("MUR-JPL-L4-GLOB-v4.1", from = as.Date("2024-01-01"))
plot_cmr_lifecycle(rev)

# opaque GranuleUR (e.g. LAADS swath products) needs an explicit gran_date_fun
cmr_audit_revisions("MOD021KM",
  from = as.Date("2026-08-01"), to = as.Date("2026-08-02"),
  gran_date_fun = function(df) as.Date(df$temporal_start))
```

cmr_collection_liveness

Check whether collections have produced granules recently

Description

Measures liveness directly, via a real `cmr_find_granules()` search over a recent lookback window, rather than trusting UMM-C's collection-level `ends_at_present` flag. That flag's reliability turns out to depend heavily on the provider: a spot-check found PO.DAAC leaves it FALSE on ~95% of its collections (including many demonstrably still-active ones), while LAADS sets it reliably for most of its catalog (see `cmr_find_collections()`'s docs). This function sidesteps that unreliability by checking real recent granule activity instead.

Usage

```
cmr_collection_liveness(
  short_name,
  provider = NULL,
  lookback_days = 60,
  delay = 0.2
)
```

Arguments

short_name	One or more CMR collection short names, as a character vector – or a data.frame/tibble with a short_name column (and optionally a provider column), such as <code>cmr_find_collections()/cmr_</code> output. This lets you pipe a filtered collection search straight in, e.g. <code>cmr_all_collections(provider = "POCLOUD") > dplyr::filter(!ends_at_present) > cmr_collection_liveness()</code> .
provider	Optional CMR provider id(s): a single value applied to every short_name, or a vector the same length as short_name. Ignored (with the data.frame's own provider column used instead) if short_name is a data.frame that has one, unless you pass provider explicitly.
lookback_days	How many days back from today to search for granules. Default 60. Low-cadence collections (annual composites, etc.) can show zero granules in 60 days while still genuinely active – raise this if you suspect that, or cross-check recent_granule_count against the collection's typical production cadence before concluding it's dead.
delay	Seconds to pause between requests when short_name has more than one element. Politeness default, not a documented CMR rate limit; set to 0 to disable.

Value

A `tibble::tibble()` with one row per short_name: short_name, provider (NA if not supplied), lookback_days, recent_granule_count (subject to `cmr_find_granules()`'s page_size cap – a count of 2000 may mean "at least 2000", not exactly 2000), and is_live (recent_granule_count > 0).

See Also

Other collection functions: [cmr_all_collections\(\)](#), [cmr_find_collections\(\)](#)

Examples

```
cmr_collection_liveness("MUR-JPL-L4-GLOB-v4.1")
cmr_collection_liveness(c("VJ146A4", "MOD021KM"), provider = "LAADS")

# straight from a filtered collection search
cmr_all_collections(provider = "POCLOUD") |>
  dplyr::filter(!ends_at_present, is.na(temporal_end), granule_count > 0) |>
  cmr_collection_liveness()
```

`cmr_find_collections` *Find CMR collections by keyword, short name, or provider*

Description

Thin wrapper around CMR's collection search (https://cmr.earthdata.nasa.gov/search/collections.umm_json), useful for discovering a `short_name` to pass to [cmr_audit_revisions\(\)](#) or [cmr_find_granules\(\)](#), or for a quick keyword search across all collections (title, abstract, science keywords, platforms, instruments, ...).

Usage

```
cmr_find_collections(
  keyword = NULL,
  short_name = NULL,
  provider = NULL,
  page_size = 100,
  page_num = 1,
  granule_count = TRUE
)
```

Arguments

<code>keyword</code>	Free-text search term(s), e.g. "sea surface temperature". Matched by CMR against title, abstract, and other collection metadata fields – not an exact match.
<code>short_name</code>	Exact (or CMR-wildcarded, with <code>options[short_name][pattern]=true</code> handled separately – not exposed here) collection short name.
<code>provider</code>	Optional CMR provider id (e.g. "POCLOUD", "LAADS") to restrict the search.
<code>page_size</code>	Max collections to return per request (CMR hard limit 2000).
<code>page_num</code>	Which page of results to fetch (1-based). Mainly useful via cmr_all_collections() , which pages through this automatically – most interactive callers can leave this at the default.

`granule_count` Fetch each collection's live granule count (via a second request to `collections.json?include_granules=true` – `collections.umm_json` has no such field). Default TRUE. This is the most direct way to spot a collection with no UMM-G granules at all (`granule_count == 0`) – some collections (a few AADC ones among them) register a single static download at the *collection* level with nothing underneath, so `cmr_find_granules()` will correctly return nothing for them. Set FALSE to skip the extra request if you don't need it.

Value

A `tibble::tibble()` with one row per collection: `short_name`, `entry_title`, `concept_id`, `provider`, `version`, `processing_level`, `abstract`, `region`, `s3_credentials_endpoint`. `region` is the AWS region (e.g. "us-west-2") a cloud-hosted collection's data lives in – `s3_urls` from `cmr_find_granules()` are only cheaply/directly readable from compute *in that region*; from elsewhere you're reading over the public internet (egress charges, no speed benefit), and some buckets block it outright. Both columns are NA for collections with no direct S3 distribution (only an https mirror). `s3_credentials_endpoint` is required regardless of region: `s3_urls` aren't publicly readable – you exchange an Earthdata Login token there for temporary AWS credentials. `temporal_start`/`temporal_end` are the collection's declared available date range (UMM-C TemporalExtents) – this is provider-entered metadata, not derived from a live granule search, so treat it as a stated range rather than a guarantee; for collections with no UMM-G granules at all (a single static download registered only at the collection level – some AADC collections are like this) it's the only date range CMR has. `temporal_end` is NA when `ends_at_present` is TRUE (an actively-growing collection; any recorded end date would go stale the moment a new granule lands). Be aware `ends_at_present` itself is frequently left FALSE (unset) on collections that are, in fact, still actively growing – a spot-check of 6 GHRSSST-family collections found only 1 correctly declared TRUE; the other 5 all had granules in the last 14 days despite FALSE. Treat FALSE as "unset/unknown" rather than "this collection has stopped" – check recent granules directly (`cmr_find_granules()`) if liveness matters. If more collections match than `page_size` allows, a message reports the total hit count – raise `page_size` (or narrow keyword/provider) to see more, or use `cmr_all_collections()` to page through everything. At least one of keyword or `short_name` must be supplied (pass `keyword = ""` to match every collection, as in `cmr_all_collections()`'s default). The total hit count is also attached as the "hits" attribute, for programmatic pagination.

See Also

Other collection functions: `cmr_all_collections()`, `cmr_collection_liveness()`

Examples

```
cmr_find_collections(keyword = "sea surface temperature", provider = "POCLOUD")
cmr_find_collections(short_name = "MUR-JPL-L4-GLOB-v4.1")
```

cmr_find_granules	<i>Pull one page-set of granules for a date window, as UMM-JSON</i>
-------------------	---

Description

Pull one page-set of granules for a date window, as UMM-JSON

Usage

```
cmr_find_granules(
  short_name,
  t0,
  t1,
  provider = NULL,
  page_size = 2000,
  bounding_box = NULL,
  polygon = NULL
)
```

Arguments

short_name	CMR collection short name (e.g. "MUR-JPL-L4-GLOB-v4.1")
t0, t1	Date range (inclusive), coercible with <code>format()</code>
provider	Optional CMR provider id, to disambiguate short names that exist under multiple providers
page_size	Max granules per request (CMR hard limit ~2000 for this non-scrolling approach)
bounding_box	Optional spatial filter: a length-4 numeric vector c(west, south, east, north) in decimal degrees. Mutually exclusive with <code>polygon</code> . Useful for subsetting swath products (e.g. MOD021KM) to a region of interest rather than pulling every granule that intersects the temporal window worldwide.
polygon	Optional spatial filter: a numeric vector of lon, lat, lon, lat, ... vertices tracing a ring (at least 3 distinct vertices). If the first and last vertex don't match, the ring is closed automatically (with a message) – CMR requires closed polygons. Mutually exclusive with <code>bounding_box</code> .

Value

A `tibble::tibble()` with one row per granule, or NULL if no granules matched. See `cmr_audit_revisions()` for the column dictionary.

See Also

Other granule functions: `cmr_audit_revisions()`, `plot_cmr_lifecycle()`

Examples

```
cmr_find_granules("MUR-JPL-L4-GLOB-v4.1", as.Date("2024-01-01"), as.Date("2024-01-05"))

# subset a high-volume swath product to a region of interest
cmr_find_granules("MOD021KM", as.Date("2026-08-01"), as.Date("2026-08-01"),
  bounding_box = c(140, -45, 155, -35))
```

plot_cmr_lifecycle	<i>Standard lifecycle diagnostic plots for a <code>cmr_audit_revisions()</code> result</i>
--------------------	--

Description

Standard lifecycle diagnostic plots for a `cmr_audit_revisions()` result

Usage

```
plot_cmr_lifecycle(rev)
```

Arguments

rev	Output of <code>cmr_audit_revisions()</code>
-----	--

Value

rev, invisibly (called for its plotting side effects)

See Also

Other granule functions: `cmr_audit_revisions()`, `cmr_find_granules()`

Examples

```
rev <- cmr_audit_revisions("MUR-JPL-L4-GLOB-v4.1", from = as.Date("2024-01-01"))
plot_cmr_lifecycle(rev)
```

Index

* collection functions

- cmr_all_collections, 2
- cmr_collection_liveness, 5
- cmr_find_collections, 6

* granule functions

- cmr_audit_revisions, 3
- cmr_find_granules, 8
- plot_cmr_lifecycle, 9

- cmr_all_collections, 2
- cmr_all_collections(), 5–7
- cmr_audit_revisions, 3
- cmr_audit_revisions(), 6, 8, 9
- cmr_collection_liveness, 5
- cmr_collection_liveness(), 3, 7
- cmr_find_collections, 6
- cmr_find_collections(), 2, 3, 5, 6
- cmr_find_granules, 8
- cmr_find_granules(), 4–7, 9
- plot_cmr_lifecycle, 9
- plot_cmr_lifecycle(), 4, 8
- tibble::tibble(), 3–5, 7, 8