

Package: usgsm2m (via r-universe)

October 5, 2026

Type Package

Title R Interface to the USGS/EROS Machine-to-Machine (M2M)
Application Programming Interface (API)

Version 0.2.0

Description The USGS/EROS Machine-to-Machine (M2M) Application Programming Interface (API) provides a method of searching and acquiring data from the USGS/EROS data inventories. The M2M API provides the same options and functions as EarthExplorer (EE) including searching for data collections and imagery, requesting metadata, and accessing all datasets in the EE catalog. Note that this package is not affiliated with the USGS/EROS, but is a community-developed interface to the M2M API for use in R.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1)

Imports dplyr, httr2, jsonify, magrittr, purrr, R6, rlang, stats,
stringr, tibble, tidyr

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

VignetteBuilder knitr

Config/testthat/edition 3

URL <https://github.com/stevenpawley/usgsm2m>,
<https://stevenpawley.github.io/usgsm2m/>

BugReports <https://github.com/stevenpawley/usgsm2m/issues>

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev libssl-dev

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

Date/Publication 2026-09-04 17:55:31 UTC

RemoteUrl <https://github.com/stevenpawley/usgsm2m>

RemoteRef HEAD

RemoteSha 592e09493680e18883f5cdd004a291ddc0d5723a

Contents

filter_cloud 2

filter_metadata_and 3

filter_metadata_between 3

filter_metadata_value 4

filter_spatial 4

filter_temporal 5

m2m_session 6

M2MDataset 7

M2MDownloadOptions 9

M2MDownloadQueue 12

M2MSceneList 15

M2MSceneSearch 17

M2MSession 19

Index 23

filter_cloud	Create a cloud cover filter for use in a scene search
--------------	---

Description

Create a cloud cover filter for use in a scene search

Usage

```
filter_cloud(min = 0, max = 100)
```

Arguments

- min Minimum cloud cover percentage (0-100)
- max Maximum cloud cover percentage (0-100)

Value

A named list with min and max cloud cover percentages

Examples

```
filter_cloud(min = 0, max = 30)
```

filter_metadata_and	Combine metadata filters with a logical operator
---------------------	--

Description

Combine metadata filters with a logical operator

Usage

```
filter_metadata_and(...)

filter_metadata_or(...)
```

Arguments

... Metadata filters to combine, as built by [filter_metadata_value()] or [filter_metadata_between()].

Value

A named list describing the combined filter.

Examples

```
filter_metadata_and(
  filter_metadata_value("5e83d14fb9436d88", "045"),
  filter_metadata_value("5e83d14ff1eda1b8", "027")
)
```

filter_metadata_between	Create a metadata filter matching a range of values
-------------------------	---

Description

Create a metadata filter matching a range of values

Usage

```
filter_metadata_between(filter_id, first, second)
```

Arguments

filter_id	The field's filter id, from the 'id' column of 'M2MDataset\$filters()'.
first	The lower bound.
second	The upper bound.

Value

A named list describing the filter.

Examples

```
filter_metadata_between("5e83d14fb9436d88", 40, 45)
```

filter_metadata_value	Create a metadata filter matching a single value
-----------------------	--

Description

Metadata filters test dataset-specific fields. Get the available ‘filter_id’ values, and what they mean, from ‘M2MDataset\$filters()’.

Usage

```
filter_metadata_value(filter_id, value, operand = c("=", "like"))
```

Arguments

filter_id	The field’s filter id, from the ‘id’ column of ‘M2MDataset\$filters()’.
value	The value to match.
operand	One of "=" or "like".

Value

A named list describing the filter.

Examples

```
filter_metadata_value("5e83d14fb9436d88", "045")
```

filter_spatial	Create a spatial filter for use in a scene search
----------------	---

Description

Create a spatial filter for use in a scene search

Usage

```
filter_spatial(ll_lon, ll_lat, ur_lon, ur_lat)
```

Arguments

ll_lon	Lower left longitude
ll_lat	Lower left latitude
ur_lon	Upper right longitude
ur_lat	Upper right latitude

Value

A named list with spatial filter parameters

Examples

```
filter_spatial(ll_lon = -120, ll_lat = 40, ur_lon = -119, ur_lat = 41)
```

filter_temporal	<i>Create a temporal filter for use in a scene search</i>
-----------------	---

Description

Create a temporal filter for use in a scene search

Usage

```
filter_temporal(start, end)
```

Arguments

start	Start date in "YYYY-MM-DD" format
end	End date in "YYYY-MM-DD" format

Value

A named list with start and end dates

Examples

```
filter_temporal("2020-07-01", "2020-07-31")
```

m2m_session

Connect to the USGS/EROS M2M API

Description

Authenticates against the M2M API and returns a session object, which is the entry point for everything else in this package. From a session you reach a dataset, from a dataset a scene search, and so on - each step returns an object whose methods are the valid next steps, so the download pipeline is discoverable by tab-completing ‘\$’ in the console.

Usage

```
m2m_session(
  username = Sys.getenv("M2M_USERNAME"),
  token = Sys.getenv("M2M_TOKEN")
)
```

Arguments

username	ERS username. This is the same username that is used to log into Earth Explorer. Defaults to the ‘M2M_USERNAME’ environment variable.
token	Earth Explorer M2M application token. To generate a token, go to https://ers.cr.usgs.gov/ . The token is different from the password used to log into Earth Explorer. Defaults to the ‘M2M_TOKEN’ environment variable.

Details

The usual pipeline is:

```
““ sess <- m2m_session() queue <- sess$ dataset("landsat_ot_c2_l2")$ search(spatial = filter_spatial(-
120, 40, -119, 41), temporal = filter_temporal("2020-07-01", "2020-07-31"))$ products()$ select_bands(c("B4",
"B5"))$ request(label = "my_order")
queue$retrieve("data/") ““
```

Value

An [M2MSession] object.

Examples

```
## Not run:
# Login using environment variables
sess <- m2m_session()

# Login using arguments
sess <- m2m_session(username = "your_username", token = "your_token")

## End(Not run)
```

M2MDataset

A dataset in the M2M catalog

Description

Returned by ‘M2MSession\$dataset()’. Holds a dataset’s metadata and is the starting point for a scene search. Not created directly.

Public fields

`info` A one-row tibble of the dataset’s metadata.

Methods

Public methods:

- `M2MDataset$new()`
- `M2MDataset$alias()`
- `M2MDataset$id()`
- `M2MDataset$filters()`
- `M2MDataset$search()`
- `M2MDataset$related_scenes()`
- `M2MDataset$print()`
- `M2MDataset$clone()`

`M2MDataset$new()`: Wrap dataset metadata. Use ‘M2MSession\$dataset()’ instead.

Usage:

```
M2MDataset$new(session, info)
```

Arguments:

`session` The parent [M2MSession].

`info` A one-row tibble of dataset metadata.

`M2MDataset$alias()`: The dataset’s system-friendly alias, e.g. “landsat_ot_c2_l2”.

Usage:

```
M2MDataset$alias()
```

Returns: A string.

`M2MDataset$id()`: The dataset’s identifier.

Usage:

```
M2MDataset$id()
```

Returns: A string.

`M2MDataset$filters()`: Metadata filter fields available for this dataset. Each row’s ‘id’ is the ‘filterId’ used when building a metadata filter to pass to ‘\$search(metadata =)’.

Usage:

```
M2MDataset$filters()
```

Returns: A tibble of filter fields.

M2MDataset\$search(): Search this dataset for scenes. All filters are optional; with none supplied every scene in the dataset matches.

Usage:

```
M2MDataset$search(
  spatial = NULL,
  temporal = NULL,
  cloud = NULL,
  metadata = NULL,
  max_results = NULL
)
```

Arguments:

spatial A spatial filter, see [filter_spatial()].

temporal A temporal filter, see [filter_temporal()].

cloud A cloud cover filter, see [filter_cloud()].

metadata A metadata filter built from '\$filters()' field ids.

max_results Maximum number of scenes to return. If 'NULL' (the default) all matching scenes are retrieved by paging through results.

Returns: An [M2MSceneSearch] object.

M2MDataset\$related_scenes(): Find the scenes related to a given scene, via the 'scene-search-secondary' endpoint.

Only datasets that define a secondary relationship support this; others raise an 'm2m_error' with code 'DATASET_ERROR'. The related scenes belong to a different dataset, so the returned search is bound to that dataset rather than this one - meaning '\$products()' on the result acts on the correct dataset.

Usage:

```
M2MDataset$related_scenes(entity_id, max_results = NULL)
```

Arguments:

entity_id The 'entityId' of the scene to find relatives of.

max_results Maximum number of scenes to return. If 'NULL' (the default) all are retrieved by paging through results.

Returns: An [M2MSceneSearch] over the secondary dataset.

M2MDataset\$print(): Print a summary of the dataset.

Usage:

```
M2MDataset$print(...)
```

Arguments:

... Ignored.

M2MDataset\$clone(): The objects of this class are cloneable with this method.

Usage:

```
M2MDataset$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

M2MDownloadOptions	<i>Products available for download</i>
--------------------	--

Description

Returned by ‘M2MSceneSearch\$products()’. Holds what the M2M API will let you download for a set of scenes, and lets you narrow it before placing a request. Not created directly.

The API offers downloads at two granularities, and this object exposes both:

* **Individual files** - the bands and per-file products nested inside a product’s ‘secondaryDownloads’, listed by ‘\$bands()’. This is the default selection. * **Whole products** - the scene-level entries the API lists, such as a Level-1 Product Bundle (a single ‘.tar’ for the scene) or a full-resolution browse image, listed by ‘\$scene_products()’. Some of these have no constituent files at all and so never appear in ‘\$bands()’.

Both are requestable. Use ‘\$select_bands()’ to queue individual files, or ‘\$select_products()’ to queue whole products; ‘\$selected()’ always shows what ‘\$request()’ will actually queue.

Public fields

products A scene-level tibble of products, with the individual bands nested in a ‘secondaryDownloads’ list-column.

Methods**Public methods:**

- `M2MDownloadOptions$new()`
- `M2MDownloadOptions$selected()`
- `M2MDownloadOptions$bands()`
- `M2MDownloadOptions$scene_products()`
- `M2MDownloadOptions$select_bands()`
- `M2MDownloadOptions$select_products()`
- `M2MDownloadOptions$filter()`
- `M2MDownloadOptions$request()`
- `M2MDownloadOptions$print()`
- `M2MDownloadOptions$clone()`

`M2MDownloadOptions$new()`: Wrap download options. Use ‘M2MSceneSearch\$products()’ instead.

Usage:

```
M2MDownloadOptions$new(session, products, selected = NULL)
```

Arguments:

session The parent [M2MSession].

products A tibble of products from the download-options endpoint.

selected An optional pre-narrowed band tibble.

M2MDownloadOptions\$selected(): The rows '\$request()' will queue. Individual files by default, or whole products after '\$select_products()'.

Usage:

```
M2MDownloadOptions$selected()
```

Returns: A tibble of downloadable items.

M2MDownloadOptions\$bands(): The currently selected downloads. An alias for '\$selected()', kept because the selection is individual bands in the common case.

Usage:

```
M2MDownloadOptions$bands()
```

Returns: A tibble of downloadable items.

M2MDownloadOptions\$scene_products(): The whole-product entries the API lists for these scenes, one row each, such as a Level-1 Product Bundle or a full-resolution browse image.

These are requestable in their own right and are a different unit from '\$bands()': a bundle is a single '.tar' holding the whole scene, where '\$bands()' would list its contents as separate files. Products with no constituent files, such as browse imagery, appear only here.

Usage:

```
M2MDownloadOptions$scene_products()
```

Returns: A tibble of products, without the 'secondaryDownloads' column.

M2MDownloadOptions\$select_bands(): Select the individual files whose 'displayId' matches any of the given patterns.

Each value is matched as a literal substring of 'displayId', which is how '"B4"' picks out '..._SR_B4' without you writing the whole name. Nothing is treated as pattern syntax, so a value containing punctuation matches itself. See '\$bands()' for the values. Matching nothing warns and lists what was available.

Like '\$select_products()', this selects afresh from everything available at that granularity rather than narrowing an existing selection - use '\$filter()' to narrow. It also does not filter on availability; chain '\$filter(bulkAvailable)' if you want only bulk-downloadable files.

Usage:

```
M2MDownloadOptions$select_bands(patterns)
```

Arguments:

patterns A character vector of patterns, e.g. 'c("B4", "B5")'.

Returns: A new [M2MDownloadOptions] with those files selected.

`M2MDownloadOptions$select_products()`: Switch the selection to whole products rather than the individual files inside them, optionally keeping only those whose ‘productName’ matches one of the given patterns.

Values are matched *exactly*, against either ‘productName’ or ‘productCode’, so a value copied out of ‘\$scene_products()’ selects what you copied. Names contain characters such as parentheses that would otherwise be read as pattern syntax, and they differ between datasets - the bundle is "Landsat Collection 2 Level-2 Product Bundle" for ‘landsat_ot_c2_l2’ but "Standard Format" for ‘corona2’ - so read them off ‘\$scene_products()’ rather than assuming. Matching nothing warns and lists what was available.

For anything looser, chain ‘\$filter()’, which takes arbitrary expressions: ‘\$select_products()\$filter(grepl("Browse", productName))’.

As with ‘\$select_bands()’ this does not filter on availability; chain ‘\$filter(available)’ to drop products the API has marked unavailable.

Usage:

```
M2MDownloadOptions$select_products(products = NULL)
```

Arguments:

products An optional character vector of ‘productName’ or ‘productCode’ values, matched exactly.

Returns: A new [M2MDownloadOptions] with the products selected.

`M2MDownloadOptions$filter()`: Narrow the current selection using ‘dplyr::filter()’ semantics against the tibble returned by ‘\$selected()’.

Usage:

```
M2MDownloadOptions$filter(...)
```

Arguments:

... Expressions passed to ‘dplyr::filter()’.

Returns: A new [M2MDownloadOptions] with the narrowed selection.

`M2MDownloadOptions$request()`: Place a download request for the current selection. Files the distribution system can serve immediately come back with URLs; anything needing preparation is collected later via ‘M2MDownloadQueue\$refresh()’.

Usage:

```
M2MDownloadOptions$request(label)
```

Arguments:

label A label identifying this order in the download queue. Required, and used to find the order again later.

Returns: An [M2MDownloadQueue] object.

`M2MDownloadOptions$print()`: Print a summary of the available and selected downloads.

Usage:

```
M2MDownloadOptions$print(...)
```

Arguments:

... Ignored.

M2MDownloadOptions\$clone(): The objects of this class are cloneable with this method.

Usage:

M2MDownloadOptions\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

M2MDownloadQueue

A download order in the M2M queue

Description

Returned by ‘M2MDownloadOptions\$request()’, or reconnected to with ‘M2MSession\$download_queue(label)’. Tracks which files are ready to download and which the distribution system is still preparing.

Public fields

label The label identifying this order.

Methods

Public methods:

- [M2MDownloadQueue\\$new\(\)](#)
- [M2MDownloadQueue\\$refresh\(\)](#)
- [M2MDownloadQueue\\$ready\(\)](#)
- [M2MDownloadQueue\\$pending\(\)](#)
- [M2MDownloadQueue\\$is_ready\(\)](#)
- [M2MDownloadQueue\\$retrieve\(\)](#)
- [M2MDownloadQueue\\$summary\(\)](#)
- [M2MDownloadQueue\\$prepare\(\)](#)
- [M2MDownloadQueue\\$remove_items\(\)](#)
- [M2MDownloadQueue\\$cancel\(\)](#)
- [M2MDownloadQueue\\$print\(\)](#)
- [M2MDownloadQueue\\$clone\(\)](#)

M2MDownloadQueue\$new(): Attach to a download order. Use ‘M2MDownloadOptions\$request()’ or ‘M2MSession\$download_queue()’.

Usage:

M2MDownloadQueue\$new(session, label, available = NULL, n_preparing = NULL)

Arguments:

session The parent [M2MSession].

label The order’s label.

available Optionally seed the ready-to-download files.

n_preparing Number of records the API accepted for preparation.

`M2MDownloadQueue$refresh()`: Re-poll the M2M queue for this label, picking up files that have finished preparing since the last check.

Unlike the other methods here, this updates the object in place rather than returning a new one, because it reflects changing server-side queue state.

Usage:

```
M2MDownloadQueue$refresh()
```

Returns: The queue, invisibly.

`M2MDownloadQueue$ready()`: The files that can be downloaded now. Readiness is based on having a URL, regardless of which internal API bucket supplied it.

Usage:

```
M2MDownloadQueue$ready()
```

Returns: A tibble of downloadable files.

`M2MDownloadQueue$pending()`: The files still being prepared, which have no URL yet.

Usage:

```
M2MDownloadQueue$pending()
```

Returns: A tibble of pending files.

`M2MDownloadQueue$is_ready()`: Whether every file in the order can be downloaded now.

Usage:

```
M2MDownloadQueue$is_ready()
```

Returns: 'TRUE' if nothing is still being prepared.

`M2MDownloadQueue$retrieve()`: Download every file that has a URL to disk. Call '`$refresh()`' first if '`$is_ready()`' is 'FALSE'.

Files are named as the server names them - the '.tar' of a product bundle, the '.TIF' of a band - rather than after the 'entityId' the queue lists them under, which for a scene-level product is the scene id and carries no extension.

A file already in 'out_dir' under that name is left alone unless 'overwrite' is 'TRUE', so re-running this after an interrupted order only fetches what is missing. Since the name usually only becomes known once the server has answered, the request is still made for each file, but none of the body is transferred for one that is already there.

Proxied downloads are reported back to the API afterwards, since it does not serve them itself and would otherwise leave them in the queue indefinitely. Files skipped as already present are reported too - they are on disk, so the order is fulfilled.

Usage:

```
M2MDownloadQueue$retrieve(out_dir, report_proxied = TRUE, overwrite = FALSE)
```

Arguments:

`out_dir` Directory to write files into. Created if missing.

`report_proxied` Whether to mark proxied downloads complete.

`overwrite` Whether to download files that are already in 'out_dir', replacing them.

Returns: A tibble with one row per file: 'entityId', 'downloadId', 'url', 'path', 'size' and 'status'. A 'status' of "expired" means the signed URL is no longer valid - '\$refresh()' and retry; "skipped" means the file was already in 'out_dir'.

M2MDownloadQueue\$summary(): Summarize this order by dataset, via the 'download-summary' endpoint.

Usage:

```
M2MDownloadQueue$summary(download_application = "M2M", send_email = FALSE)
```

Arguments:

download_application The application the downloads were requested under. Required by the API; the counts come back as zero if it does not match the one used at request time.

send_email Whether the API should also email the summary.

Returns: A list with 'label', 'download_count', 'scene_count', 'total_estimated_size' and a 'collections' tibble.

M2MDownloadQueue\$prepare(): Move this order's scenes into the queue for processing, via the 'download-order-load' endpoint.

Unlike the other methods here this changes server-side state: it is what starts a staged order being prepared. Follow it with '\$refresh()' to pick up URLs as they become ready.

Usage:

```
M2MDownloadQueue$prepare(download_application = NULL)
```

Arguments:

download_application Optional application name to scope the order to.

Returns: The queue, invisibly.

M2MDownloadQueue\$remove_items(): Remove individual files from this order, leaving the rest of it in place. '\$cancel()' drops the whole order instead.

Ids come from the 'downloadId' column of '\$ready()' or '\$pending()'.

Usage:

```
M2MDownloadQueue$remove_items(download_id, quiet = FALSE)
```

Arguments:

download_id A vector of download ids.

quiet Suppress the message shown before a large batch.

Returns: The queue, invisibly.

M2MDownloadQueue\$cancel(): Cancel this order, removing it from the M2M queue.

Usage:

```
M2MDownloadQueue$cancel()
```

Returns: The queue, invisibly.

M2MDownloadQueue\$print(): Print a summary of the order.

Usage:

```
M2MDownloadQueue$print(...)
```

Arguments:

... Ignored.

M2MDownloadQueue\$clone(): The objects of this class are cloneable with this method.

Usage:

```
M2MDownloadQueue$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

M2MSceneList

A scene list registered on the M2M server

Description

Scene lists are the M2M API's way of naming a set of scenes so that other endpoints can act on them in bulk: 'download-options' and 'scene-metadata-list' will not take scene ids inline, only a 'listId'.

'M2MSceneSearch\$products()' creates and uses one for you, so the usual pipeline never mentions scene lists. Reach for this class when the list itself is what you want - '\$metadata()' fetches meta-data for every scene in one call rather than one call per scene, and '\$summary()' reports the set's combined extent.

Reattach to an existing list with 'M2MSession\$scene_list()'. Searches create the lists required for '\$products()' internally. Not created directly.

Public fields

list_id The scene list's identifier.

dataset_name The dataset alias the list belongs to, or 'NA' if not yet known. Reattaching to a list by id leaves this 'NA' until '\$dataset()' resolves it, since the API does not report a list's dataset except through its summary.

removed Whether '\$remove()' has deleted this list. Lists also expire on their own, which this does not track.

Methods**Public methods:**

- [M2MSceneList\\$new\(\)](#)
- [M2MSceneList\\$dataset\(\)](#)
- [M2MSceneList\\$scenes\(\)](#)
- [M2MSceneList\\$metadata\(\)](#)
- [M2MSceneList\\$summary\(\)](#)
- [M2MSceneList\\$products\(\)](#)
- [M2MSceneList\\$remove\(\)](#)

- `M2MSceneList$print()`
- `M2MSceneList$clone()`

`M2MSceneList$new()`: Attach to a scene list. Use `'M2MSession$scene_list()'` instead.

Usage:

```
M2MSceneList$new(session, list_id, dataset_name = NA_character_)
```

Arguments:

`session` The parent [M2MSession].

`list_id` The scene list identifier.

`dataset_name` The dataset alias the list belongs to, or 'NA' if unknown.

`M2MSceneList$dataset()`: The dataset alias this list belongs to, looked up from the list's summary if it is not already known and cached thereafter.

A scene list can span several datasets, in which case there is no single answer and this errors - pass `'dataset_name'` to `'M2MSession$scene_list()'` to say which one you mean.

Usage:

```
M2MSceneList$dataset()
```

Returns: The dataset alias, as a string.

`M2MSceneList$scenes()`: The entity ids currently in the list.

Usage:

```
M2MSceneList$scenes()
```

Returns: A tibble of scene entity ids.

`M2MSceneList$metadata()`: Full metadata for every scene in the list.

Usage:

```
M2MSceneList$metadata()
```

Returns: A tibble of scene metadata.

`M2MSceneList$summary()`: Summarize the list's spatial and temporal extent, and the datasets it spans.

Usage:

```
M2MSceneList$summary()
```

Returns: A tibble, with overall bounds in the `'spatialBounds'` attribute.

`M2MSceneList$products()`: Discover the products available for the scenes in this list.

Usage:

```
M2MSceneList$products(band_group = TRUE)
```

Arguments:

`band_group` Whether to include secondary file groups (bands).

Returns: An [M2MDownloadOptions] object.

`M2MSceneList$remove()`: Delete the scene list from the M2M server. Lists also expire on their own after a period of inactivity.

Usage:

`M2MSceneList$remove()`

Returns: The scene list, invisibly.

`M2MSceneList$print()`: Print a summary of the scene list.

Usage:

`M2MSceneList$print(...)`

Arguments:

... Ignored.

`M2MSceneList$clone()`: The objects of this class are cloneable with this method.

Usage:

`M2MSceneList$clone(deep = FALSE)`

Arguments:

deep Whether to make a deep clone.

M2MSceneSearch

Results of a scene search

Description

Returned by `'M2MDataset$search()'`. Holds the matching scenes and is the bridge to discovering downloadable products. Not created directly.

Public fields

`scenes` A tibble of the matching scenes.

`total_hits` The total number of scenes the API reported as matching, which exceeds `'nrow(scenes)'` when `'max_results'` was set.

Methods

Public methods:

- [M2MSceneSearch\\$new\(\)](#)
- [M2MSceneSearch\\$dataset\(\)](#)
- [M2MSceneSearch\\$filter\(\)](#)
- [M2MSceneSearch\\$combine\(\)](#)
- [M2MSceneSearch\\$products\(\)](#)
- [M2MSceneSearch\\$print\(\)](#)
- [M2MSceneSearch\\$clone\(\)](#)

M2MSceneSearch\$new(): Wrap search results. Use ‘M2MDataset\$search()’ instead.

Usage:

```
M2MSceneSearch$new(session, dataset, results, total_hits)
```

Arguments:

session The parent [M2MSession].

dataset The [M2MDataset] that was searched.

results A tibble of matching scenes.

total_hits Total matches reported by the API.

M2MSceneSearch\$dataset(): The dataset these scenes came from.

Usage:

```
M2MSceneSearch$dataset()
```

Returns: An [M2MDataset] object.

M2MSceneSearch\$filter(): Subset the scenes before requesting products, using ‘dplyr::filter()’ semantics against the ‘scenes’ tibble.

Usage:

```
M2MSceneSearch$filter(...)
```

Arguments:

... Expressions passed to ‘dplyr::filter()’.

Returns: A new [M2MSceneSearch] with the subset applied.

M2MSceneSearch\$combine(): Combine these scenes with those of other searches, so one ‘\$products()’ call and one download order covers all of them.

Scenes appearing in more than one search are kept once, so searches that overlap do not double-count. All the searches must be of the same dataset, since a set of products is discovered per dataset.

Usage:

```
M2MSceneSearch$combine(...)
```

Arguments:

... Other [M2MSceneSearch] objects.

Returns: A new [M2MSceneSearch] holding every scene.

M2MSceneSearch\$products(): Discover the products (bands and related files) available for these scenes. This registers a scene list server-side first, which the M2M API requires before download options can be listed.

Usage:

```
M2MSceneSearch$products(band_group = TRUE)
```

Arguments:

band_group Whether to include secondary file groups, i.e. the individual bands. ‘TRUE’ by default.

Returns: An [M2MDownloadOptions] object.

M2MSceneSearch\$print(): Print a summary of the search results.

Usage:

M2MSceneSearch\$print(...)

Arguments:

... Ignored.

M2MSceneSearch\$clone(): The objects of this class are cloneable with this method.

Usage:

M2MSceneSearch\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

M2MSession

M2M API session

Description

Represents an authenticated connection to the USGS/EROS M2M API. Create one with [m2m_session()] rather than calling 'M2MSession\$new()' directly.

Methods

Public methods:

- M2MSession\$new()
- M2MSession\$dataset()
- M2MSession\$find_datasets()
- M2MSession\$download_queue()
- M2MSession\$downloads()
- M2MSession\$remove_items()
- M2MSession\$download_labels()
- M2MSession\$eula()
- M2MSession\$scene_list()
- M2MSession\$ers_session()
- M2MSession\$logout()
- M2MSession\$print()
- M2MSession\$clone()

M2MSession\$new(): Authenticate and open a session. Both arguments are required; [m2m_session()] is the documented way in and is what supplies the 'M2M_USERNAME' / 'M2M_TOKEN' defaults.

Usage:

M2MSession\$new(username, token)

Arguments:

username ERS username.

token Earth Explorer M2M application token.

M2MSession\$dataset(): Look up a dataset by alias or id. This is the usual starting point for a search.

Usage:

```
M2MSession$dataset(name = NULL, id = NULL)
```

Arguments:

name The system-friendly dataset alias, e.g. "landsat_ot_c2_l2". Use this or 'id', not both.

id The dataset identifier. Use this or 'name', not both.

Returns: An [M2MDataset] object.

M2MSession\$find_datasets(): Search the catalog for datasets matching a name pattern. Use this when you don't yet know a dataset's alias; wildcards are applied automatically around 'pattern'.

Usage:

```
M2MSession$find_datasets(pattern, spatial = NULL, temporal = NULL)
```

Arguments:

pattern Search pattern for the dataset name.

spatial Optional spatial filter, see [filter_spatial()].

temporal Optional temporal filter, see [filter_temporal()].

Returns: A tibble of matching datasets. Pass a 'datasetAlias' value from the result to '\$dataset()' to continue.

M2MSession\$download_queue(): Reconnect to an existing download order by its label, for example to resume collecting a large order in a later R session.

Usage:

```
M2MSession$download_queue(label)
```

Arguments:

label The label the downloads were requested under.

Returns: An [M2MDownloadQueue] object.

M2MSession\$downloads(): List every download in the queue, regardless of status or label.

Usage:

```
M2MSession$downloads()
```

Returns: A tibble of queued downloads.

M2MSession\$remove_items(): Remove individual downloads from the queue, across any label. Use this to clear out completed or unwanted entries without dropping a whole order the way 'M2MDownloadQueue\$cancel()' does.

Ids come from the 'downloadId' column of '\$downloads()':

```
"" done <- dplyr::filter(sess$downloads(), statusText == "Complete") sess$remove_items(done$downloadId)
""
```

The API removes one item per request and answers the same way for an id that does not exist, so a successful call is not evidence that anything was removed - check '\$downloads()' afterwards.

Usage:

```
M2MSession$remove_items(download_id, quiet = FALSE)
```

Arguments:

`download_id` A vector of download ids.

`quiet` Suppress the message shown before a large batch.

Returns: The number of ids submitted, invisibly.

M2MSession\$download_labels(): List the distinct order labels in the download queue, one row each. Use this to find an order to reconnect to with '`$download_queue()`' when you no longer remember its label.

Usage:

```
M2MSession$download_labels(download_application = NULL)
```

Arguments:

`download_application` Optional application name to scope the listing to.

Returns: A tibble with 'label', 'downloadCount', 'totalComplete', 'downloadSize' and 'dateEntered' (epoch milliseconds).

M2MSession\$eula(): Retrieve the text of one or more End User License Agreements. Some datasets require a EULA to be accepted (once, through the EarthExplorer website) before downloads will succeed.

Usage:

```
M2MSession$eula(code = NULL, codes = NULL)
```

Arguments:

`code` A single EULA code. Use this or 'codes', not both.

`codes` A character vector of EULA codes. Use this or 'code'.

Returns: A tibble with 'eulaCode' and 'agreementContent'.

M2MSession\$scene_list(): Attach to a scene list that already exists server-side.

Usage:

```
M2MSession$scene_list(list_id, dataset_name = NULL)
```

Arguments:

`list_id` The scene list identifier.

`dataset_name` The dataset alias the list belongs to. Optional - it is looked up from the list's summary when needed. Supply it to save that lookup, or to pick one when the list spans several datasets.

Returns: An [M2MSceneList] object.

M2MSession\$ers_session(): Return the small session value used by the package's internal HTTP functions. This method is intended for package internals.

Usage:

```
M2MSession$ers_session()
```

Returns: An internal 'ers_session' object.

M2MSession\$logout(): End the session, invalidating its API key. M2M sessions also expire on their own after a period of inactivity.

Usage:

`M2MSession$logout()`

Returns: The session, invisibly.

M2MSession\$print(): Print a summary of the session.

Usage:

`M2MSession$print(...)`

Arguments:

... Ignored.

M2MSession\$clone(): The objects of this class are cloneable with this method.

Usage:

`M2MSession$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

Index

`filter_cloud`, [2](#)
`filter_metadata_and`, [3](#)
`filter_metadata_between`, [3](#)
`filter_metadata_or`
 (`filter_metadata_and`), [3](#)
`filter_metadata_value`, [4](#)
`filter_spatial`, [4](#)
`filter_temporal`, [5](#)

`m2m_session`, [6](#)
`M2MDataset`, [7](#)
`M2MDownloadOptions`, [9](#)
`M2MDownloadQueue`, [12](#)
`M2MSceneList`, [15](#)
`M2MSceneSearch`, [17](#)
`M2MSession`, [19](#)