Package: mobdb (via r-universe)

October 27, 2026

Type Package

Title Access the 'Mobility Database' API to Discover Transit Feeds

Version 1.0.3

Description Search and access transit feed data from the 'Mobility Database' <https://mobilitydatabase.org>. The package wraps the 'Mobility Database' API, allowing users to discover 'GTFS' (General Transit Feed Specification) and 'GBFS' (General Bikeshare Feed Specification) feeds from agencies worldwide. Functions are designed to integrate seamlessly with packages like 'tidytransit' and 'gtfstools' for subsequent feed analysis.

URL https://mobdb.jasonadle.dev, https://github.com/jasonad123/mobdb

BugReports https://github.com/jasonad123/mobdb/issues

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports curl, httr2 (>= 1.0.0), cli (>= 3.0.0), rlang (>= 1.0.0), tibble (>= 3.0.0), dplyr (>= 1.0.0), lifecycle, digest

Suggests testthat (>= 3.0.0), withr (>= 2.0.0), httptest2, knitr, rmarkdown, tidytransit, gtfsio, keyring, sf, gbfs, zip, hms

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libssl-dev

Repository https://jasonad123.r-universe.dev

Date/Publication 2026-08-28 16:29:33 UTC

RemoteUrl https://github.com/jasonad123/mobdb

RemoteRef HEAD

RemoteSha 3924598572c1b7c641ed4b2107d1f58226b8306b

Contents

download_best_feed . . . 2
download_feed . . . 5
feeds . . . 8
feeds_bbox . . . 10
filter_by_validation . . . 11
get_validation_report . . . 13
gtfs_to_spec_format . . . 14
mobdb_browse . . . 16
mobdb_cache_clear . . . 16
mobdb_cache_info . . . 17
mobdb_cache_list . . . 17
mobdb_cache_path . . . 18
mobdb_datasets . . . 19
mobdb_extract_datasets . . . 20
mobdb_extract_locations . . . 21
mobdb_extract_urls . . . 22
mobdb_feed_url . . . 23
mobdb_get_dataset . . . 24
mobdb_get_feed . . . 24
mobdb_has_key . . . 25
mobdb_read_gtfs . . . 25
mobdb_search . . . 26
mobdb_set_key . . . 28
view_validation_report . . . 29

Index **31**

`download_best_feed` *Download the "best" GTFS Schedule feed with smart selection*

Description

[Experimental]

A wrapper around `download_feed()` that *automagically* selects the best GTFS Schedule feed when multiple options exist. This function:

- Searches for feeds using provider name or location
- Ranks feeds by status, official designation, and validation quality
- Prompts for user selection when multiple equally-ranked feeds exist (in interactive mode)
- Falls back to historical datasets when current feed is marked "future" or "inactive"
- Only works with GTFS Schedule feeds (not GTFS-RT or GBFS)

This is designed for use cases where you just want the best, most recent feed without needing to specify exact feed IDs or handle multiple results manually.

Usage

```
download_best_feed(
  provider = NULL,
  country_code = NULL,
  subdivision_name = NULL,
  municipality = NULL,
  feed_name = NULL,
  prefer_official = TRUE,
  prefer_active = TRUE,
  max_validation_errors = NULL,
  interactive = NULL,
  exclude_flex = TRUE,
  use_source_url = FALSE,
  auth_args = NULL,
  export_path = NULL,
  raw = NULL,
  ...
)
```

Arguments

provider	Provider/agency name (partial match).
country_code	ISO 2-letter country code (requires subdivision_name).
subdivision_name	State/province/region name (requires country_code).
municipality	City name.
feed_name	Feed name filter (case-insensitive substring match).
prefer_official	Logical. If TRUE (default), prefer feeds marked as official.
prefer_active	Logical. If TRUE (default), prefer feeds with status "active" over "future", "development", "deprecated", or "inactive".
max_validation_errors	Integer. Maximum number of validation errors allowed. Feeds exceeding this threshold are filtered out. If NULL (default), no filtering.
interactive	Logical. If TRUE, prompt user to select when multiple equally-ranked feeds exist. If FALSE, automatically select the first highest-ranked feed with a warning. If NULL (default), uses getOption("mobdb.interactive") or falls back to interactive() to detect if running in an interactive R session.
exclude_flex	Logical. If TRUE (default), exclude GTFS-Flex feeds.
use_source_url	Logical. Download from agency's source URL (TRUE) or Mobility Database's hosted URL (FALSE, default).
auth_args	Authentication arguments if required (see download_feed()).
export_path	A string. Optional path to save the GTFS feed as a ZIP file (e.g., "data/gtfs/feed.zip"). See download_feed() for details.

raw — A logical. Controls whether the file saved to export_path is the raw download (TRUE) or a parsed-and-re-exported version (FALSE). Defaults to TRUE when export_path is provided, FALSE otherwise.

... — Additional arguments passed to tidytransit::read_gtfs().

Value

If export_path is provided with raw = TRUE (the default when exporting), the file path (invisibly). Otherwise, a gtfs object from tidytransit, or NULL if user cancels selection.

Selection algorithm

When multiple feeds match the search criteria, feeds are ranked by:

1. **Status** (if prefer_active = TRUE): active > future > development > inactive > deprecated
2. **Official designation** (if prefer_official = TRUE): official > unclassified > unofficial
3. **Validation quality**: Feeds with fewer errors score higher
4. **Service date coverage**: Feeds covering today's date score higher
5. **Recency**: More recently added feeds get a tiebreaker boost

If multiple feeds have the same score and interactive = TRUE, you'll be prompted to choose.

Status handling

The function handles different feed statuses as follows:

- **"active"**: Preferred. Feed should be used in public trip planners.
- **"future"** or **"inactive"**: Automatically searches for historical datasets with service dates covering today. "future" feeds are not yet active; "inactive" feeds haven't been recently updated and may provide outdated information.
- **"deprecated"**: Explicitly deprecated and shouldn't be used. Warns user to search for a replacement feed.
- **"development"**: For development purposes only, shouldn't be used in production.

GTFS Schedule Only

Like download_feed(), this function only works with GTFS Schedule feeds. For GTFS-RT or GBFS feeds, use mobdb_read_gtfs() or fetch URLs with mobdb_get_feed().

See Also

download_feed() for precise control, feeds() to explore available feeds before downloading, mobdb_search() for full-text search with validation data

Examples

```
# Simple one-shot download by provider name
bart_feed <- download_best_feed(provider = "Bay Area Rapid Transit")


# Non-interactive mode (for scripts)
options(mobdb.interactive = FALSE)
feed <- download_best_feed(provider = "Bay Area Rapid Transit")
```

---

download_feed *Download GTFS Schedule feeds*

Description

A convenience function for downloading GTFS Schedule feeds from the Mobility Database. This is a "one-stop-shop" that can search for feeds by provider/location and download them in a single call, or download a specific feed by ID.

Note: This function is specifically designed for GTFS Schedule feeds only. GTFS Realtime and GBFS feeds use a different data model and are not supported by this function.

This function was formerly called `mobdb_download_feed()`.

Usage

```
download_feed(
  feed_id = NULL,
  provider = NULL,
  country_code = NULL,
  subdivision_name = NULL,
  municipality = NULL,
  exclude_flex = TRUE,
  feed_name = NULL,
  use_source_url = FALSE,
  dataset_id = NULL,
  latest = TRUE,
  status = "active",
  official = NULL,
  auth_args = NULL,
  export_path = NULL,
  raw = NULL,
  ...
)
```

Arguments

feed_id	A string or data frame. The unique identifier for the feed (e.g., "mdb-2862"), or a single-row data frame from feeds() or mobdb_search(). If a data frame is provided, the feed ID will be extracted automatically. If provided, all other search parameters are ignored.
provider	A string. Filter by provider/agency name (partial match). Use this to search for feeds without knowing the feed_id.
country_code	A string. Two-letter ISO country code (e.g., "US", "CA").
subdivision_name	A string. State, province, or region name.
municipality	A string. City or municipality name.
exclude_flex	A logical. If TRUE (default), automatically exclude feeds with "flex" in the feed name (case-insensitive). GTFS-Flex feeds are an extension of the GTFS Schedule specification and may contain files that have unique schemas that may not work with standard GTFS tools.
feed_name	A string. Optional filter for feed name. If provided, only feeds whose feed_name contains this string (case-insensitive) will be considered. Use NULL (default) to skip this filter.
use_source_url	A logical. If FALSE (default), uses Mobility Database's hosted/archived URL which ensures you get the exact version in their database. If TRUE, uses the provider's direct source URL which may be more current but could differ from the hosted version.
dataset_id	A string. Optional specific dataset ID for historical versions (e.g., "mdb-53-202510250025"). If provided, downloads that specific dataset version instead of the latest. Cannot be used with use_source_url = TRUE. If dataset_id is provided without feed_id, the feed ID will be automatically extracted from the dataset ID format.
latest	A logical. If TRUE (default), download the most recent dataset. If FALSE, returns information about all available datasets for the feed without downloading. Only works when feed_id is provided directly; cannot be used with search parameters like provider or country_code.
status	A string. Feed status filter: "active" (default), "deprecated", "inactive", "development", or "future". Only used when searching by provider/location.
official	A logical. If NULL (default), return all feeds regardless of official status when searching by provider/location. If TRUE, return official feeds and feeds with unknown official status (NA). If FALSE, only return feeds explicitly marked as unofficial.
auth_args	A string. Some agencies require authentication to download feeds directly from their source URLs. Provide your API key/token in one of two formats: • Just the value: "your_api_key_here" • Parameter and value: "apikey=your_api_key_here" Also accepts a value stored in .Renviron (.e.g Sys.getenv("AGENCY_API_KEY") stored in the same formats) Only valid when use_source_url = TRUE. If a feed requires authentication, you'll receive an error message with a link to obtain

credentials. The authentication method (URL parameter or HTTP header) is determined automatically from the feed's metadata.

`export_path` A string. Optional path to save the GTFS feed as a ZIP file (e.g., "data/gtfs/feed.zip"). By default, saves the raw file exactly as downloaded. Set `raw = FALSE` to parse with tidytransit and re-export in GTFS-spec-compliant format (requires `tidytransit` and `gtfsio`). If `NULL` (default), the feed is not saved to disk.

`raw` A logical. Controls whether the file saved to `export_path` is the raw download (`TRUE`) or a parsed-and-re-exported version (`FALSE`). Defaults to `TRUE` when `export_path` is provided, `FALSE` otherwise.

`...` Additional arguments passed to `tidytransit::read_gtfs()`.

Value

If `export_path` is provided with `raw = TRUE` (the default when exporting), the file path (invisibly). If `latest = TRUE`, a `gtfs` object as returned by `tidytransit::read_gtfs()`. If `latest = FALSE`, a tibble of all available datasets with their metadata.

See Also

`mobdb_datasets()` to list all available historical versions, `get_validation_report()` to check feed quality before downloading, `feeds()` to search for feeds, `mobdb_read_gtfs()` for more flexible GTFS reading

Examples

```
# Download by feed ID
gtfs <- download_feed("mdb-2862")

# Download from search results
feeds <- feeds(provider = "TransLink", data_type = "gtfs")
gtfs <- download_feed(feeds[1, ])

# Search and download by provider name
gtfs <- download_feed(provider = "Arlington")

# Download using agency's source URL instead of Mobility Database
gtfs <- download_feed(provider = "TriMet", use_source_url = TRUE)

# See all available versions for a feed
versions <- download_feed("mdb-2862", latest = FALSE)

# Download a specific historical version (feed_id auto-extracted from dataset_id)
historical <- download_feed(dataset_id = "mdb-53-202507240047")

# Filter by location (may return multiple feeds requiring disambiguation,
# in which case refine with `provider` or `feed_name`)
try(download_feed(
  country_code = "US",
  subdivision_name = "California",
  municipality = "San Francisco"
```

```
))

# Save GTFS feed to disk (raw file, no parsing required)
path <- download_feed("mdb-247", export_path = tempfile(fileext = ".zip"))
```

`feeds` *List and filter GTFS Schedule, GTFS-RT, and GBFS feeds*

Description

Query the Mobility Database for transit/bikeshare feeds matching specified criteria. Returns a tibble with feed metadata including download URLs.

This function was formerly called `mobdb_feeds()`.

Usage

```
feeds(
  provider = NULL,
  country_code = NULL,
  subdivision_name = NULL,
  municipality = NULL,
  data_type = NULL,
  status = NULL,
  official = NULL,
  limit = 100,
  offset = 0,
  use_cache = TRUE
)
```

Arguments

`provider`	A string. Filter by provider/agency name (partial match).
`country_code`	A string. Two-letter ISO country code (e.g., "US", "CA"). **Note:** Location filters (`country_code`, `subdivision_name`, `municipality`) require `data_type` to be specified.
`subdivision_name`	A string. State, province, or region name. Requires `data_type` to be specified.
`municipality`	A string. City, municipality, or jurisdiction name. Requires `data_type` to be specified.
`data_type`	A string. Type of feed: "gtfs" (schedule), "gtfs_rt" (realtime), or "gbfs" (bike share). Required when using location filters.
`status`	A string. Feed status: "active", "deprecated", "inactive", "development", or "future".
`official`	A logical. If `TRUE`, only return official feeds. If `FALSE`, only return unofficial feeds. If `NULL` (default), return all feeds regardless of official status.

limit	An integer. Maximum number of results to return (default: 100).
offset	An integer. Number of results to skip for pagination (default: 0).
use_cache	A logical. If TRUE (default), use cached results if available. If FALSE, always fetch fresh data from the API. Cached data expires after 1 hour.

Value

A tibble containing feed information with columns including:

- id - Unique feed identifier
- data_type - Type of feed (gtfs, gtfs_rt, or gbfs)
- created_at - Date and time feed was added to database
- external_ids - External identifier information
- provider - Transit agency/provider name
- feed_contact_email - Contact email for the feed
- source_info - Data frame containing:
  - producer_url - Direct download URL for the feed
  - authentication_type - Type of auth required (0 = none)
  - authentication_info_url - Human-readable page for authentication info
  - api_key_parameter_name - Name of the parameter to pass in the URL to provide the API key
  - license_url - License information
- created_at - Feed creation timestamp
- status - Feed status (active, inactive, deprecated)
- official - Whether feed is official
- official_updated_at - Date and time of last update
- Additional metadata columns

Examples

```
# Get all active GTFS feeds in California
ca_feeds <- feeds(
  country_code = "US",
  subdivision_name = "California",
  data_type = "gtfs",
  status = "active"
)

# Search for a specific provider
sf_muni <- feeds(provider = "San Francisco")

# Get feeds with pagination
first_100 <- feeds(limit = 100, offset = 0)
next_100 <- feeds(limit = 100, offset = 100)
```

`feeds_bbox` *Find GTFS Schedule feeds by bounding box*

Description

Discover GTFS Schedule feeds whose geographic coverage overlaps with or is contained within a specified bounding box. This function is designed for feed discovery based on geographic location.

Important: This function only works with GTFS Schedule feeds because bounding box data is derived from the feed's latest dataset.

Usage

```
feeds_bbox(
  bbox,
  filter_method = "completely_enclosed",
  provider = NULL,
  country_code = NULL,
  subdivision_name = NULL,
  municipality = NULL,
  status = NULL,
  official = NULL,
  limit = 100,
  offset = 0,
  use_cache = TRUE
)
```

Arguments

`bbox`	A numeric vector of length 4 specifying the bounding box as `c(min_lon, min_lat, max_lon, max_lat)` in WGS84 coordinates (EPSG:4326). Alternatively, an `sf` bbox object can be provided if `sf` is installed.
`filter_method`	A string. Method for filtering feeds by bounding box: • `"completely_enclosed"` (default) - Feeds whose coverage is fully inside the specified bounding box • `"partially_enclosed"` - Feeds whose coverage overlaps with the box • `"disjoint"` - Feeds whose coverage is completely outside the box
`provider`	A string. Filter by provider/agency name (partial match).
`country_code`	A string. Two-letter ISO country code (e.g., "US", "CA").
`subdivision_name`	A string. State, province, or region name.
`municipality`	A string. City or municipality name.
`status`	A string. Feed status: "active", "deprecated", "inactive", "development", or "future".

official	A logical. If TRUE, only return official feeds. If FALSE, only return unofficial feeds. If NULL (default), return all feeds regardless of official status.
limit	An integer. Maximum number of results to return (default: 100).
offset	An integer. Number of results to skip for pagination (default: 0).
use_cache	A logical. If TRUE (default), use cached results if available. If FALSE, always fetch fresh data from the API. Cached data expires after 1 hour.

Value

A tibble containing GTFS Schedule feed information with columns including:

- id - Unique feed identifier
- data_type - Always "gtfs" for this function
- provider - Transit agency/provider name
- status - Feed status
- source_info - Data frame containing download URLs and auth info
- latest_dataset - Information about the most recent dataset including bounding box coordinates
- Additional metadata columns

Examples

```
# Find feeds in the San Francisco Bay Area
# Bounding box: c(min_lon, min_lat, max_lon, max_lat)
bay_area_feeds <- feeds_bbox(
  bbox = c(-122.5, 37.2, -121.8, 38.0),
  filter_method = "partially_enclosed"
)

# Find feeds completely within Los Angeles County
la_feeds <- feeds_bbox(
  bbox = c(-118.9, 33.7, -118.0, 34.8),
  filter_method = "completely_enclosed",
  status = "active"
)
```

---

filter_by_validation *Filter feeds or datasets by validation quality*

Description

Filter feed or dataset results by validation quality thresholds. This is a convenience wrapper around get_validation_report() that returns the original data filtered to only include feeds/datasets meeting your quality criteria.

Note: This function does *not* support GBFS validation reports at this time as GBFS validation reports are located at a different endpoint and have a different validation criteria.

Usage

```
filter_by_validation(
  data,
  max_errors = NULL,
  max_warnings = NULL,
  max_info = NULL,
  require_validation = TRUE
)
```

Arguments

data	A tibble from feeds(), mobdb_datasets(), or mobdb_search().
max_errors	Maximum number of validation errors allowed. Use 0 for error-free feeds. If NULL (default), no error filtering is applied.
max_warnings	Maximum number of validation warnings allowed. If NULL (default), no warning filtering is applied.
max_info	Maximum number of informational notices allowed. If NULL (default), no info filtering is applied.
require_validation	Logical. If TRUE (default), exclude feeds/datasets that have no validation data. If FALSE, include them in results.

Value

A filtered version of the input data frame containing only feeds/datasets that meet the specified quality criteria.

See Also

get_validation_report() to inspect validation metrics, view_validation_report() to view full validation reports

Examples

```
# Create sample data with validation information (search results structure)
sample_data <- tibble::tibble(
  id = c("mdb-1", "mdb-2", "mdb-3"),
  provider = c("Agency A", "Agency B", "Agency C"),
  latest_dataset = tibble::tibble(
    id = c("mdb-1-202501", "mdb-2-202501", "mdb-3-202501"),
    validation_report = tibble::tibble(
      total_error = c(0L, 5L, 100L),
      total_warning = c(10L, 50L, 500L),
      total_info = c(5L, 10L, 20L)
    )
  )
)

# Filter to feeds with zero errors
```

```
filter_by_validation(sample_data, max_errors = 0)

# Filter with multiple criteria
filter_by_validation(sample_data, max_errors = 10, max_warnings = 100)


# With real API data:
ca_feeds <- feeds(
  country_code = "US",
  subdivision_name = "California",
  data_type = "gtfs"
)
clean_feeds <- filter_by_validation(ca_feeds, max_errors = 0)
```

`get_validation_report` *Get GTFS-Schedule validation report for feeds or datasets*

Description

Extract validation report summary from feed/dataset results. The Mobility Database runs all GTFS Schedule feeds through the canonical GTFS validator, and this function surfaces that validation data to help assess feed quality before downloading.

Note: This function does *not* support GBFS validation reports at this time as GBFS validation reports are located at a different endpoint and have a different validation criteria.

Usage

```
get_validation_report(data)
```

Arguments

`data`	A tibble from `feeds()`, `mobdb_datasets()`, or `mobdb_search()`.

Value

A tibble with validation summary information:

- `feed_id` or `dataset_id` - Identifier
- `provider` - Provider name (if available)
- `total_error` - Number of validation errors
- `total_warning` - Number of validation warnings
- `total_info` - Number of informational notices
- `html_report` - URL to full HTML validation report
- `json_report` - URL to JSON validation report

See Also

filter_by_validation() to filter by quality thresholds, view_validation_report() to open full HTML/JSON reports in browser, mobdb_datasets() to get dataset information with validation data, mobdb_extract_datasets() to extract validation from search results

Examples

```
# Create sample dataset data with validation_report
sample_datasets <- tibble::tibble(
  id = "mdb-1-202501010000",
  feed_id = "mdb-1",
  validation_report = tibble::tibble(
    total_error = 0L,
    total_warning = 5L,
    total_info = 10L,
    unique_error_count = 0L,
    unique_warning_count = 3L,
    unique_info_count = 5L,
    url_html = "https://example.com/report.html",
    url_json = "https://example.com/report.json",
    validated_at = "2025-01-01T00:00:00Z",
    validator_version = "5.0.0"
  )
)

# Extract validation report
get_validation_report(sample_datasets)


# With real API data:
bart_feeds <- feeds(provider = "Bay Area Rapid Transit", data_type = "gtfs")
datasets <- mobdb_datasets(bart_feeds$id[1])
validation <- get_validation_report(datasets)
```

---

gtfs_to_spec_format *Convert tidygtfs object to GTFS-spec-compliant format*

Description

Converts a tidygtfs object (as returned by tidytransit::read_gtfs()) back to GTFS-spec-compliant string formats. This reverses tidytransit's automatic type conversions:

- **Date columns** (R Date objects) are converted back to YYYYMMDD strings (e.g., as.Date("2024-01-15") becomes "20240115")
- **Time columns** (hms/difftime objects) are converted back to HH:MM:SS strings, preserving values >= 24:00:00 for trips past midnight (e.g., hms::hms(hours = 25, minutes = 30) becomes "25:30:00")

Columns that are already in the correct format (character or integer) are left unchanged. Returns a modified copy; the original object is not modified.

Usage

```
gtfs_to_spec_format(gtfs)
```

Arguments

gtfs	A gtfs/tidygtfs object, typically from tidytransit::read_gtfs() or download_feed().

Value

A modified copy of the gtfs object with date and time columns converted to GTFS-spec-compliant strings.

Affected tables and columns

Date columns (YYYYMMDD):

- calendar: start_date, end_date
- calendar_dates: date
- feed_info: feed_start_date, feed_end_date

Time columns (HH:MM:SS):

- stop_times: arrival_time, departure_time
- frequencies: start_time, end_time

Examples

```
gtfs <- download_feed("mdb-247")

# Dates are R Date objects from tidytransit
class(gtfs$calendar$start_date)
# [1] "Date"

# Convert to GTFS-spec format
spec <- gtfs_to_spec_format(gtfs)
spec$calendar$start_date
# [1] "20240101"

spec$stop_times$arrival_time[1]
# [1] "08:30:00"
```

mobdb_browse *Load the Mobility Database in browser*

Description

Opens the Mobility Database in your default web browser. You'll need to log in or sign up on the website to get an API key to use this package.

Usage

```
mobdb_browse()
```

Value

Invisibly returns the URL that was opened.

Examples

```
## Not run:
mobdb_browse()

## End(Not run)
```

mobdb_cache_clear *Clear mobdb cache*

Description

Removes cached files from the cache directory. Can remove all files or only those older than a specified number of days.

Usage

```
mobdb_cache_clear(older_than = NULL)
```

Arguments

older_than	Optional. Remove only files older than this many days. If NULL (default), removes all cached files.

Value

Invisibly returns NULL. Called for its side effect of removing cached files (and, if requested via older_than, only those older than the given number of days)

Examples

```
# Clear all cache
mobdb_cache_clear()

# Clear only files older than 7 days
mobdb_cache_clear(older_than = 7)
```

`mobdb_cache_info` *Show cache information*

Description

Displays information about the mobdb cache including location, number of files, and total size.

Usage

```
mobdb_cache_info()
```

Value

List with cache information (invisibly):

`path`	Cache directory path
`files`	Number of cached files
`size_mb`	Total size in megabytes
`exists`	Whether cache directory exists

Examples

```
# Show cache info
mobdb_cache_info()
```

`mobdb_cache_list` *List cached files*

Description

Returns a tibble with information about all cached files, including file name, size, modification time, and age.

Usage

```
mobdb_cache_list()
```

Value

Tibble with columns:

`file`	File name
`size_mb`	File size in megabytes
`modified`	Last modification time
`age_hours`	Age in hours

Examples

```
# List all cached files
mobdb_cache_list()
```

`mobdb_cache_path` *Set or show mobdb cache directory*

Description

Configure the directory where mobdb caches API responses. By default, mobdb uses `tools::R_user_dir("mobdb", "cache")`.

Usage

```
mobdb_cache_path(path = NULL, install = FALSE, overwrite = FALSE)
```

Arguments

`path`	Optional. Directory path for cache. If NULL (default), shows current cache path without changing it.
`install`	Logical. If TRUE, adds MOBDB_CACHE_PATH to .Renviron for persistence across R sessions. Default: FALSE
`overwrite`	Logical. If TRUE, overwrites existing MOBDB_CACHE_PATH in .Renviron. Default: FALSE

Value

Character string with cache path (invisibly)

Examples

```
# Show current cache path
mobdb_cache_path()


# Set for current session only
mobdb_cache_path(file.path(tempdir(), "mobdb_cache_example"))
```

```
## Not run:
# Set permanently in .Renviron
mobdb_cache_path("~/my_mobdb_cache", install = TRUE)

## End(Not run)
```

`mobdb_datasets` *Get datasets for a feed*

Description

Retrieve information about available datasets (historical versions) for a specific feed. Each dataset represents a snapshot of the feed at a particular point in time.

Usage

```
mobdb_datasets(feed_id, latest = TRUE, use_cache = TRUE)
```

Arguments

`feed_id`	A string. The unique identifier for the feed.
`latest`	A logical. If `TRUE` (default), return only the most recent dataset. If `FALSE`, return all available datasets.
`use_cache`	A logical. If `TRUE` (default), use cached results if available. If `FALSE`, always fetch fresh data from the API. Cached data expires after 24 hours (datasets are immutable).

Value

A tibble containing dataset information including:

- `id` - Dataset identifier
- `feed_id` - Associated feed ID
- `downloaded_at` - Timestamp when dataset was captured
- `hash` - Hash of the dataset file
- `download_url` - URL to download this specific dataset version
- Additional metadata columns

See Also

`download_feed()` to download specific historical versions, `get_validation_report()` to extract validation data from datasets, `mobdb_get_dataset()` to get details for a specific dataset

Examples

```
# Get latest dataset for a feed (GTFS schedule feeds only)
latest <- mobdb_datasets("mdb-53")

# Get all historical datasets
all_versions <- mobdb_datasets("mdb-53", latest = FALSE)
```

`mobdb_extract_datasets` *Extract latest dataset information from search results*

Description

Helper function to extract dataset details from search results. The search endpoint includes a `latest_dataset` field with comprehensive information about the most recent dataset, including validation results.

Usage

```
mobdb_extract_datasets(results)
```

Arguments

`results`	A tibble returned by `mobdb_search()`.

Value

A tibble with one row per feed, containing key dataset information:

- `id` - Feed ID
- `dataset_id` - Latest dataset ID
- `hosted_url` - URL to download the latest validated dataset
- `downloaded_at` - When the dataset was captured
- `hash` - Dataset file hash
- `service_date_range_start` - Start of service dates
- `service_date_range_end` - End of service dates
- `total_error` - Number of validation errors (if available)
- `total_warning` - Number of validation warnings (if available)
- Note: Report URLs (html_report, json_report) are only available when using `mobdb_datasets()`, not from search results

See Also

`get_validation_report()` to get full validation details with report URLs, `mobdb_search()` to search for feeds, `mobdb_datasets()` to get dataset information directly

Examples

```
# Create sample data matching mobdb_search() output with latest_dataset
sample_results <- tibble::tibble(
  id = "mdb-1",
  provider = "Sample Agency",
  latest_dataset = tibble::tibble(
    id = "mdb-1-202501010000",
    hosted_url = "https://example.com/dataset.zip",
    downloaded_at = "2025-01-01T00:00:00Z",
    hash = "abc123",
    service_date_range_start = "2025-01-01",
    service_date_range_end = "2025-12-31",
    agency_timezone = "America/Los_Angeles",
    validation_report = tibble::tibble(
      total_error = 0L,
      total_warning = 5L,
      total_info = 10L,
      url_html = "https://example.com/report.html",
      url_json = "https://example.com/report.json"
    )
  )
)

# Extract dataset information
mobdb_extract_datasets(sample_results)


# With real API data:
results <- mobdb_search("transit")
datasets <- mobdb_extract_datasets(results)
```

`mobdb_extract_locations` *Extract location information from search results*

Description

Helper function to extract and unnest location information from search results. The `locations` field in search results is a list of data frames; this function flattens it into a more usable format.

Usage

```
mobdb_extract_locations(results, unnest = TRUE)
```

Arguments

`results`	A tibble returned by `mobdb_search()`.
`unnest`	Logical. If `TRUE` (default), unnest locations so each feed-location combination gets its own row. If `FALSE`, return a summary of locations per feed.

Value

A tibble with location information. If `unnest = TRUE`, each row represents a feed-location pair. If `unnest = FALSE`, returns one row per feed with concatenated location strings.

Examples

```
# Create sample data matching mobdb_search() output structure
sample_results <- tibble::tibble(
  id = c("mdb-1", "mdb-2"),
  provider = c("Agency A", "Agency B"),
  locations = list(
    data.frame(
      country_code = "US",
      country = "United States",
      subdivision_name = "California",
      municipality = "San Francisco"
    ),
    data.frame(
      country_code = "CA",
      country = "Canada",
      subdivision_name = "British Columbia",
      municipality = "Vancouver"
    )
  )
)

# Extract and unnest locations
mobdb_extract_locations(sample_results)

# Get summary without unnesting
mobdb_extract_locations(sample_results, unnest = FALSE)


# With real API data:
results <- mobdb_search("California")
locations <- mobdb_extract_locations(results)
```

---

mobdb_extract_urls *Extract download URLs from feed results*

Description

Helper function to extract producer URLs from a tibble of feeds returned by `feeds()` or `mobdb_search()`. This is useful when you want to get all the source URLs from a set of search results.

Usage

```
mobdb_extract_urls(feeds)
```

Arguments

feeds	A tibble returned by feeds() or mobdb_search().

Value

A character vector of download URLs, with the same length as the input tibble. Returns NA for feeds without a URL.

Examples

```
# Create sample data matching feeds() output structure
sample_feeds <- tibble::tibble(
  id = c("mdb-1", "mdb-2"),
  provider = c("Agency A", "Agency B"),
  source_info = tibble::tibble(
    producer_url = c("https://example.com/feed1.zip", "https://example.com/feed2.zip"),
    authentication_type = c(0L, 0L)
  )
)

# Extract URLs from sample data
mobdb_extract_urls(sample_feeds)


# With real API data:
ca_gtfs <- feeds(subdivision_name = "California", data_type = "gtfs")
ca_urls <- mobdb_extract_urls(ca_gtfs)
```

---

mobdb_feed_url *Get download URL for a feed*

Description

Convenience function to quickly get the direct download or source URL for a feed. This is useful for passing to tidytransit::read_gtfs() or similar functions.

Usage

```
mobdb_feed_url(feed_id)
```

Arguments

feed_id	A string. The unique identifier for the feed.

Value

A string. The direct download URL, or NULL if not available.

Examples

```
# Get download URL
url <- mobdb_feed_url("mdb-53")


# Use with tidytransit
library(tidytransit)
gtfs <- read_gtfs(url)
```

`mobdb_get_dataset` *Get details for a specific dataset*

Description

Retrieve detailed information about a single dataset by its ID.

Usage

```
mobdb_get_dataset(dataset_id)
```

Arguments

`dataset_id`	A string. The unique identifier for the dataset.

Value

A list containing detailed dataset information.

Examples

```
# Get details for a specific dataset
dataset_info <- mobdb_get_dataset("mdb-53-202510250025")
```

`mobdb_get_feed` *Get details for a specific feed*

Description

Retrieve detailed information about a single feed by its ID.

Usage

```
mobdb_get_feed(feed_id)
```

Arguments

feed_id	A string. The unique identifier for the feed.

Value

A list containing detailed feed information.

Examples

```
# Get details for a specific feed
feed_details <- mobdb_get_feed("mdb-53")
```

---

mobdb_has_key *Check if Mobility Database API token is configured*

Description

Check whether a refresh token has been set for the current session or is available in the environment.

Usage

```
mobdb_has_key()
```

Value

Logical. TRUE if a token is configured, FALSE otherwise.

Examples

```
# Check if API token is configured
mobdb_has_key()
```

---

mobdb_read_gtfs *Read GTFS feed directly from Mobility Database*

Description

[Superseded]

Note: This function is superseded by download_feed(), which provides the same functionality plus integrated search, Flex filtering, and more control over data sources. New code should use download_feed() instead.

Convenience wrapper that fetches a feed's download URL from the Mobility Database and passes it to tidytransit::read_gtfs(). Requires the tidytransit package.

Usage

```
mobdb_read_gtfs(feed_id, dataset_id = NULL, ...)
```

Arguments

feed_id	A string. The unique identifier for the feed, or a data frame with a single row from feeds() or mobdb_search().
dataset_id	A string. Optional specific dataset ID. If NULL (default), uses the current/latest feed URL.
...	Additional arguments passed to tidytransit::read_gtfs().

Value

A gtfs object as returned by tidytransit::read_gtfs().

Examples

```
# Read latest feed by ID (Bay Area Rapid Transit)
gtfs <- mobdb_read_gtfs("mdb-53")

# Read from search results
feeds <- feeds(provider = "TransLink", data_type = "gtfs")
gtfs <- mobdb_read_gtfs(feeds[1, ])

# Read specific historical dataset
gtfs_historical <- mobdb_read_gtfs("mdb-53", dataset_id = "mdb-53-202510250025")
```

---

mobdb_search *Search for feeds across the Mobility Database*

Description

[Experimental]

Perform a text search across feed names, providers, and locations.

Note: Search is performed on English words and is case insensitive. Word order is not relevant for matching. For example `New York City Transit` will be parsed as `new` & `york` & `city` & `transit`

The endpoint used has known issues with relevance ranking. For better results when searching by provider name, consider using feeds() with the `provider` parameter.

Usage

```
mobdb_search(
  query,
  feed_id = NULL,
  data_type = NULL,
  official = NULL,
  status = NULL,
  gtfs_feature = NULL,
  gbfs_version = NULL,
  limit = 50,
  offset = 0,
  use_cache = TRUE
)
```

Arguments

query	A string. Search query string. Searches across provider names, feed names, and locations.
feed_id	A string. The unique identifier for the feed (e.g. "mdb-696", "mdb-1707", "gbfs-lime_vancouver_bc"). When provided, searches only for this specific feed and all other filter parameters must be omitted.
data_type	A string. Optional filter by data type: "gtfs", "gtfs_rt", or "gbfs".
official	A logical. If TRUE, only return official feeds when searching by provider/location. If FALSE, only return unofficial feeds. If NULL (default), return all feeds regardless of official status.
status	A string. Feed status filter: "active", "deprecated", "inactive", "development", or "future".
gtfs_feature	A character vector. Filter feeds by their GTFS features. Only valid for GTFS Schedule feeds. GTFS features definitions are defined here.
gbfs_version	A character vector. Comma-separated list of GBFS versions to filter by. Only valid for GBFS feeds. GBFS version notes are defined here
limit	An integer. Maximum number of results (default: 50).
offset	An integer. Number of results to skip for pagination (default: 0).
use_cache	A logical. If TRUE (default), use cached results if available. If FALSE, always fetch fresh data from the API. Cached data expires after 1 hour.

Value

A tibble of matching feeds. Note that search results include additional fields compared to feeds():

- locations - List of data frames with geographical information
- latest_dataset - Data frame with most recent dataset details and validation
- Core fields (id, provider, data_type, status, source_info) are the same

Examples

```
# Search for transit agencies (Note: results may not be well-ranked)
results <- mobdb_search("transit")

# Better approach: use feeds() with provider filter
bart <- feeds(provider = "BART")
mta <- feeds(provider = "MTA New York")

# Search with filters
gtfs_feeds <- mobdb_search(
  "transit",
  data_type = "gtfs",
  official = TRUE
)

# Search with pagination
first_50 <- mobdb_search("train", limit = 50, offset = 0)
next_50 <- mobdb_search("train", limit = 50, offset = 50)

# Search for official GTFS feeds only
official_feeds <- mobdb_search("metro", official = TRUE, data_type = "gtfs")

# Note: For location-specific searches (state/province/city), use feeds() instead:
ontario_transit <- feeds(
  provider = "transit",
  country_code = "CA",
  subdivision_name = "Ontario",
  data_type = "gtfs"
)
```

---

mobdb_set_key *Set Mobility Database API refresh token*

Description

Store your Mobility Database API refresh token for use in subsequent API calls. The refresh token is used to generate short-lived access tokens automatically.

Usage

```
mobdb_set_key(refresh_token, install = FALSE)
```

Arguments

refresh_token	A string. Your Mobility Database API refresh token. Obtain this by signing up at https://mobilitydatabase.org and navigating to your account details page.
install	A logical. If TRUE, will set the token in .Renviron for use across sessions. If FALSE (default), token is only set for the current session.

Value

Invisibly returns TRUE if successful.

Examples

```
# Set token for current session
mobdb_set_key("your_refresh_token_here")


## Not run:
# Set token permanently in .Renviron
mobdb_set_key("your_refresh_token_here", install = TRUE)

## End(Not run)
```

---

view_validation_report

View GTFS-Schedule validation report in browser

Description

Opens the Mobility Database validation report for a feed or dataset in your default web browser. The report shows detailed validation results from the canonical GTFS validator.

Note: This function does *not* support GBFS validation reports at this time as GBFS validation reports are located at a different endpoint and have a different validation criteria.

Usage

```
view_validation_report(data, format = "html")
```

Arguments

data	One of: • A single-row tibble from mobdb_datasets() or mobdb_search() • A character string feed_id (e.g., "mdb-482") • A character string dataset_id (e.g., "mdb-482-202511010126")
format	Character. Report format: "html" (default) or "json".

Value

Invisibly returns the URL that was opened.

See Also

get_validation_report() to extract validation data as a tibble, filter_by_validation() to filter by quality thresholds, mobdb_datasets() to get dataset information with validation reports

Examples

```
## Not run:
# View validation report for Alexandria DASH
view_validation_report("mdb-482")

# View report from dataset results
datasets <- mobdb_datasets("mdb-482")
view_validation_report(datasets)

# View JSON report instead
view_validation_report("mdb-482", format = "json")

## End(Not run)
```

Index

∗ **authentication**
    mobdb_has_key, 25
    mobdb_set_key, 28
∗ **datasets**
    mobdb_datasets, 19
    mobdb_get_dataset, 24
∗ **validation**
    filter_by_validation, 11
    get_validation_report, 13
    view_validation_report, 29

download_best_feed, 2
download_feed, 5
download_feed(), *2–4*, *15*, *19*, *25*

feeds, 8
feeds(), *4*, *6*, *7*, *12*, *13*, *22*, *23*, *26*, *27*
feeds_bbox, 10
filter_by_validation, 11
filter_by_validation(), *14*, *29*

get_validation_report, 13
get_validation_report(), *7*, *11*, *12*, *19*, *20*, *29*
gtfs_to_spec_format, 14

mobdb_browse, 16
mobdb_cache_clear, 16
mobdb_cache_info, 17
mobdb_cache_list, 17
mobdb_cache_path, 18
mobdb_datasets, 19
mobdb_datasets(), *7*, *12–14*, *20*, *29*
mobdb_extract_datasets, 20
mobdb_extract_datasets(), *14*
mobdb_extract_locations, 21
mobdb_extract_urls, 22
mobdb_feed_url, 23
mobdb_get_dataset, 24
mobdb_get_dataset(), *19*
mobdb_get_feed, 24
mobdb_get_feed(), *4*
mobdb_has_key, 25
mobdb_read_gtfs, 25
mobdb_read_gtfs(), *4*, *7*
mobdb_search, 26
mobdb_search(), *4*, *6*, *12*, *13*, *20–23*, *26*, *29*
mobdb_set_key, 28

tidytransit::read_gtfs(), *4*, *7*, *14*, *15*, *25*, *26*

view_validation_report, 29
view_validation_report(), *12*, *14*