

Package: osrm.backend (via r-universe)

May 30, 2026

Title Bindings for 'Open Source Routing Machine'

Version 0.3.1

Description Install and control 'Open Source Routing Machine' ('OSRM') backend executables to prepare routing data and run/stop a local 'OSRM' server. For computations with the running server use the 'osrm' R package (<https://cran.r-project.org/package=osrm>).

License MIT + file LICENSE

URL <https://github.com/e-kotov/osrm.backend>,
<https://www.ekotov.pro/osrm.backend/>

BugReports <https://github.com/e-kotov/osrm.backend/issues>

Imports digest, httr2, jsonlite, lifecycle, processx, ps, viridisLite

Suggests DT, knitr, mapgl, osrm, rmarkdown, sf, shiny, testthat (>= 3.0.0), tibble, withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

SystemRequirements OSRM backend binaries (>= v5.27.0) <https://github.com/Project-OSRM/osrm-backend>; package downloads binaries automatically if not found

Config/pak/sysreqs libssl-dev

Repository <https://e-kotov.r-universe.dev>

Date/Publication 2026-05-30 16:50:08 UTC

RemoteUrl <https://github.com/e-kotov/osrm.backend>

RemoteRef HEAD

RemoteSha f6ed0a0e9ec11170bfba77349e2cb7a27087c2ba

Contents

osrm_check_available_versions	2
osrm_check_latest_version	3
osrm_cleanup	3
osrm_clear_path	5
osrm_contract	6
osrm_customize	8
osrm_extract	10
osrm_find_profile	12
osrm_get_server_profile	13
osrm_gui	14
osrm_install	16
osrm_partition	18
osrm_prepare_graph	20
osrm_servers	23
osrm_start	24
osrm_start_server	26
osrm_stop	29
osrm_stop_all	31
osrm_uninstall	32
osrm_which	33
Index	35

osrm_check_available_versions

Check for Available OSRM Versions

Description

[Stable]

Queries the GitHub API to get a list of all available version tags for the OSRM backend that have binaries for the current platform.

Usage

```
osrm_check_available_versions(prereleases = FALSE)
```

Arguments

`prereleases` A logical value. If TRUE, include pre-release versions in the returned list. Defaults to FALSE.

Value

A character vector of available version tags.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {  
  # Get all stable versions with binaries for this platform  
  osrm_check_available_versions()  
}
```

osrm_check_latest_version

Check for the Latest Stable OSRM Version

Description

[Stable]

Queries the GitHub API to find the most recent stable (non-pre-release) version tag for the OSRM backend that has binaries available for the current platform.

Usage

```
osrm_check_latest_version()
```

Value

A string containing the latest version tag (e.g., "v26.4.1").

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {  
  # Get the latest stable version number of OSRM backend  
  osrm_check_latest_version()  
}
```

osrm_cleanup

Clean Up OSRM Files in a Directory

Description

[Stable]

Remove OSRM-generated files from a directory. This is useful when switching between algorithms (CH and MLD) or when you want to start fresh.

Usage

```
osrm_cleanup(path, keep_osm = TRUE, dry_run = FALSE, quiet = FALSE)
```

Arguments

path	A string. Path to an OSRM file or directory containing OSRM files. If a file path is provided (e.g., data.osm.pbf or data.osrm.hsgr), the base name will be extracted and all related .osrm.* files will be removed.
keep_osm	Logical. If TRUE (default), keeps the original .osm.pbf (or .osm, .osm.bz2) file. If FALSE, removes it as well.
dry_run	Logical. If TRUE, shows what would be deleted without actually deleting. Default is FALSE.
quiet	Logical. If TRUE, suppresses messages. Default is FALSE.

Details

OSRM creates many .osrm.* files during the extract, contract, partition, and customize stages. This function helps clean up these files.

Important: The CH and MLD algorithms cannot safely coexist in the same directory because the MLD partition stage modifies some extract-stage files. Use this function to clean up before switching algorithms.

Value

Invisibly returns a character vector of removed file paths.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # Stage a temporary workspace with placeholder OSRM files
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  file.create(
    file.path(osrm_dir, "cur.osrm.timestamp"),
    file.path(osrm_dir, "cur.osrm.hsgr"),
    file.path(osrm_dir, "cur.osrm.mldgr"),
    file.path(osrm_dir, "cur.osrm.partition")
  )

  # Preview what would be deleted
  osrm_cleanup(osrm_dir, dry_run = TRUE)

  # Clean up OSRM artifacts (keep the OSM file)
  osrm_cleanup(osrm_dir)
```

```

# Remove everything including the OSM source
osrm_cleanup(osrm_dir, keep_osm = FALSE)

osrm_uninstall(
  dest_dir = install_dir,
  clear_path = TRUE,
  force = TRUE,
  quiet = TRUE
)
unlink(osrm_dir, recursive = TRUE)
}

```

osrm_clear_path	<i>Clear OSRM Path from Project's .Rprofile</i>
-----------------	---

Description

Scans the .Rprofile file in the current project's root directory and removes any lines that were added by osrm_install() to modify the PATH.

Usage

```
osrm_clear_path(quiet = FALSE)
```

Arguments

quiet A logical value. If TRUE, suppresses messages. Defaults to FALSE.

Value

TRUE if the file was modified, FALSE otherwise.

Examples

```

if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  # Clean up a temporary project's .Rprofile
  old <- setwd(tempdir())
  on.exit(setwd(old), add = TRUE)
  writeLines(
    c(
      "#added-by-r-pkg-osrm.backend",
      'Sys.setenv(PATH = paste("dummy", Sys.getenv("PATH"), sep = .Platform$path.sep))'
    ),
    ".Rprofile"
  )
  osrm_clear_path(quiet = TRUE)
  unlink(".Rprofile")
}

```

 osrm_contract

Contract OSRM Graph for Contraction Hierarchies (CH)

Description

[Stable]

Run the `osrm-contract` tool to contract an OSRM graph for the CH pipeline. After running, a companion `<base>.osrm.hsgr` file must exist to confirm success.

Usage

```
osrm_contract(
  input_osrm,
  threads = 8L,
  verbosity = c("INFO", "NONE", "ERROR", "WARNING", "DEBUG"),
  segment_speed_file = NULL,
  turn_penalty_file = NULL,
  edge_weight_updates_over_factor = 0,
  parse_conditionals_from_now = 0,
  time_zone_file = NULL,
  quiet = FALSE,
  verbose = FALSE,
  spinner = TRUE,
  echo_cmd = FALSE
)
```

Arguments

<code>input_osrm</code>	A string. Path to a <code>.osrm.timestamp</code> file, the base path to the <code>.osrm</code> files (without extension), or a directory containing exactly one <code>.osrm.timestamp</code> file.
<code>threads</code>	An integer. Number of threads to use; default 8.
<code>verbosity</code>	A string. Log verbosity level passed to <code>-l/--verbosity</code> (one of "NONE", "ERROR", "WARNING", "INFO", "DEBUG", "VERBOSE", "QUIET", "NONE"); default "INFO".
<code>segment_speed_file</code>	A string or NULL. Path to <code>nodeA,nodeB,speed</code> CSV; default NULL.
<code>turn_penalty_file</code>	A string or NULL. Path to <code>from_,to_,via_nodes,penalties</code> CSV; default NULL.
<code>edge_weight_updates_over_factor</code>	A numeric. Threshold for logging large weight updates; default 0.
<code>parse_conditionals_from_now</code>	A numeric. UTC timestamp for conditional restrictions; default 0.
<code>time_zone_file</code>	A string or NULL. GeoJSON file for time zone boundaries; default NULL.
<code>quiet</code>	A logical. Master switch that suppresses package messages and process output when TRUE; default FALSE.

verbose	A logical. When TRUE and quiet = FALSE, streams stdout and stderr from the underlying processx::run calls.
spinner	A logical. When TRUE and quiet = FALSE, shows a spinner instead of live logs; default TRUE.
echo_cmd	A logical. When TRUE and quiet = FALSE, prints each command before running; default FALSE.

Value

An object of class `osrm_job` with the following elements:

osrm_job_artifact The path to the contracted `.osrm.hmgr` file.

osrm_working_dir The directory containing all OSRM files.

logs The processx::run result object.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # Prepare a small graph then contract it for the CH pipeline
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  profile <- osrm_find_profile("car.lua")

  extract_job <- osrm_extract(
    input_osm = tmp_pbf,
    profile = profile,
    overwrite = TRUE,
    threads = 1L
  )

  ch_graph <- osrm_contract(extract_job, threads = 1L, verbose = TRUE)
  ch_graph$osrm_job_artifact

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

osrm_customize

Customize OSRM Graph for Multi-Level Dijkstra (MLD)

Description

[Stable]

Run the `osrm-customize` tool to customize a partitioned OSRM graph for the MLD pipeline. After running, a companion `<base>.osrm.mldgr` file must exist to confirm success.

Usage

```
osrm_customize(
  input_osrm,
  threads = 8L,
  verbosity = c("INFO", "NONE", "ERROR", "WARNING", "DEBUG"),
  segment_speed_file = NULL,
  turn_penalty_file = NULL,
  edge_weight_updates_over_factor = 0,
  parse_conditionals_from_now = 0,
  time_zone_file = NULL,
  quiet = FALSE,
  verbose = FALSE,
  spinner = TRUE,
  echo_cmd = FALSE
)
```

Arguments

<code>input_osrm</code>	A string. Path to a <code>.osrm.partition</code> file, the base path to the partitioned <code>.osrm</code> files (without extension), or a directory containing exactly one <code>.osrm.partition</code> file.
<code>threads</code>	An integer. Number of threads to use; default 8 (<code>osrm-customize</code> 's default).
<code>verbosity</code>	A string. Log verbosity level passed to <code>-l/--verbosity</code> (one of "NONE", "ERROR", "WARNING", "INFO", "DEBUG", "NONE"); default "INFO".
<code>segment_speed_file</code>	A string or NULL. Path to <code>nodeA,nodeB,speed</code> CSV; default NULL.
<code>turn_penalty_file</code>	A string or NULL. Path to <code>from_,to_,via_nodes,penalties</code> CSV; default NULL.
<code>edge_weight_updates_over_factor</code>	A numeric. Factor threshold for logging large weight updates; default 0.
<code>parse_conditionals_from_now</code>	A numeric. UTC timestamp from which to evaluate conditional turn restrictions; default 0.

time_zone_file	A string or NULL. GeoJSON file with time zone boundaries; default NULL.
quiet	A logical. Master switch that suppresses package messages and process output when TRUE; default FALSE.
verbose	A logical. When TRUE and quiet = FALSE, streams stdout and stderr from the underlying processx::run calls.
spinner	A logical. When TRUE and quiet = FALSE, shows a spinner instead of live logs; default TRUE.
echo_cmd	A logical. When TRUE and quiet = FALSE, prints each command before running; default FALSE.

Value

An object of class `osrm_job` with the following elements:

osrm_job_artifact The path to the customized `.osrm.mldgr` file.

osrm_working_dir The directory containing all OSRM files.

logs The `processx::run` result object.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # Partition then customize a graph for the MLD pipeline
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  profile <- osrm_find_profile("car.lua")

  extract_job <- osrm_extract(
    input_osm = tmp_pbf,
    profile = profile,
    overwrite = TRUE,
    threads = 1L
  )
  partition_job <- osrm_partition(extract_job, threads = 1L)

  mld_graph <- osrm_customize(partition_job, threads = 1L, verbose = TRUE)
  mld_graph$osrm_job_artifact

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
  )
}
```

```

    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}

```

osrm_extract

Extract OSM into OSRM Graph Files

Description

[Stable]

Run the `osrm-extract` tool to preprocess an OSM file (`.osm`, `.osm.bz2`, or `.osm.pbf`) into the base `.osrm` graph files using a specified Lua profile. After running, a companion `<base>.osrm.timestamp` file must exist to confirm success.

Usage

```

osrm_extract(
  input_osm,
  profile = osrm_find_profile("car.lua"),
  threads = 8L,
  overwrite = FALSE,
  verbosity = c("INFO", "NONE", "ERROR", "WARNING", "DEBUG"),
  data_version = NULL,
  small_component_size = 1000L,
  with_osm_metadata = FALSE,
  parse_conditional_restrictions = FALSE,
  location_dependent_data = NULL,
  disable_location_cache = FALSE,
  dump_nbg_graph = FALSE,
  quiet = FALSE,
  verbose = FALSE,
  spinner = TRUE,
  echo_cmd = FALSE
)

```

Arguments

<code>input_osm</code>	A string. Path to the input OSM file (<code>.osm</code> , <code>.osm.bz2</code> , or <code>.osm.pbf</code>) or a directory containing exactly one OSM file with a supported extension.
<code>profile</code>	A string. Path to the OSRM Lua profile (e.g. returned by <code>osrm_find_profile("car.lua")</code>).
<code>threads</code>	An integer. Number of threads for <code>-t/--threads</code> ; default 8 (OSRM's default).
<code>overwrite</code>	A logical. If <code>FALSE</code> (default), stops when any existing <code>.osrm*</code> files matching the base name are found alongside <code>input_osm</code> . Set to <code>TRUE</code> to proceed regardless.

verbosity	A string. Log verbosity level passed to <code>-l/--verbosity</code> (one of "NONE", "ERROR", "WARNING", "INFO", "DEBUG", "VERBOSE", "QUIET", "SILENT", default "INFO").
data_version	A string or NULL. Passed to <code>-d/--data_version</code> ; default NULL, in which case the option is omitted.
small_component_size	An integer. For <code>--small-component-size</code> ; default 1000 (OSRM's default).
with_osm_metadata	A logical. If TRUE, adds <code>--with-osm-metadata</code> ; default FALSE.
parse_conditional_restrictions	A logical. If TRUE, adds <code>--parse-conditional-restrictions</code> ; default FALSE.
location_dependent_data	A string or NULL. Path to GeoJSON, passed to <code>--location-dependent-data</code> ; default NULL, in which case the option is omitted.
disable_location_cache	A logical. If TRUE, adds <code>--disable-location-cache</code> ; default FALSE.
dump_nbg_graph	A logical. If TRUE, adds <code>--dump-nbg-graph</code> ; default FALSE.
quiet	A logical. Master switch that suppresses package messages and process output when TRUE; default FALSE.
verbose	A logical. When TRUE and quiet = FALSE, streams stdout and stderr from the underlying processx::run calls.
spinner	A logical. When TRUE and quiet = FALSE, shows a spinner instead of live logs; default TRUE.
echo_cmd	A logical. When TRUE and quiet = FALSE, prints each command before running; default FALSE.

Value

An object of class `osrm_job` with the following elements:

osrm_job_artifact The path to the generated `.osrm.timestamp` file.

osrm_working_dir The directory containing all OSRM files.

logs The `processx::run` result object.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  # Install OSRM (temporary, session PATH)
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # copy example OSM PBF into a temporary workspace to avoid polluting pkg data
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
```

```

tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)

# Find the path to the profile first
car_profile <- osrm_find_profile("car.lua")

# extract OSRM graph files
result <- osrm_extract(
  input_osm      = tmp_pbf,
  profile        = car_profile,
  overwrite      = TRUE,
  threads        = 1L
)
# path to generated .osrm files (specifically, the .osrm.timestamp file)
result$osrm_job_artifact

# Clean up binaries and workspace
osrm_uninstall(
  dest_dir = install_dir,
  clear_path = TRUE,
  force = TRUE,
  quiet = TRUE
)
unlink(osrm_dir, recursive = TRUE)
}

```

osrm_find_profile *Locate an OSRM Lua profile (e.g. car.lua) in a host installation*

Description

[Stable]

By default OSRM ships profiles for "car", "bike" and "foot" in a profiles/ directory alongside the binaries. This function will try to locate osrm-routed on the PATH, resolve symlinks, and look first for a profiles/ directory next to the binary (as placed there by osrm_install()). If that fails, it looks for sibling directories share/osrm/profiles and share/osrm-backend/profiles. IF that fails, it will try to fall back on /usr/local/share/osrm/profiles, /usr/local/share/osrm-backend/profiles, /usr/share/osrm/profiles, and /usr/share/osrm-backend/profiles.

Usage

```
osrm_find_profile(profile = "car.lua")
```

Arguments

profile	A single string, the name of the Lua profile file (e.g. "car.lua"). Defaults to "car.lua".
---------	--

Value

The normalized filesystem path to the profile.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )
  osrm_find_profile("car.lua")
  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
}
```

osrm_get_server_profile

Retrieve the OSRM Profile for a Running Server

Description**[Stable]**

Attempts to determine the profile (e.g., "car", "bike", "foot") used by an OSRM server. It follows a priority list:

1. Checks the active server registry for the given port or ID.
2. Checks for a dataset.meta.json file in the directory of the graph file.
3. Checks the graph filename for hints (e.g. berlin-car.osrm).
4. Falls back to getOption("osrm.profile").

Usage

```
osrm_get_server_profile(input_osrm = NULL, port = NULL)
```

Arguments

input_osrm	Optional. Can be an OSRM job process (an osrm_server object inheriting from processx::process), a path string, or NULL.
port	Optional integer. The port of the server.

Value

A character string representing the profile name (default "car").

 osrm_gui

Launch a GUI to View and Debug OSRM Routing

Description**[Experimental]**

Launches a lightweight Shiny application to interactively visualize routing on a local OSRM server. This interface mimics the `r5r_gui` experience, supporting left-click for start, right-click for end, and draggable markers.

Usage

```
osrm_gui(
  input_osrm = NULL,
  port = "auto",
  style = "https://basemaps.cartocdn.com/gl/voyager-gl-style/style.json",
  center = NULL,
  zoom = NULL,
  autozoom = TRUE,
  update_while_drag = FALSE,
  debug = FALSE
)
```

Arguments

<code>input_osrm</code>	Optional. Can be: <ul style="list-style-type: none"> • An OSRM job process (an <code>osrm_server</code> object inheriting from <code>processx::process</code>) returned by <code>osrm_start()</code> or <code>osrm_start_server()</code>. When providing a process, you must also specify port explicitly. • A path string to an <code>.osrm.hmgr</code> or <code>.osrm.mldgr</code> file. • A path string to an <code>.osm.pbf</code> file (will be prepared and started). • <code>NULL</code> (default): Auto-detects a running OSRM server using <code>osrm_servers()</code>. Errors if no servers are running.
<code>port</code>	Integer or "auto". The port the server is running on (or should run on). Defaults to "auto", which attempts to auto-detect a running OSRM server using <code>osrm_servers()</code> . If multiple servers are running, the most recent one is selected with a warning. If no servers are running, an error is raised.
<code>style</code>	Character. Map style for mapgl. Defaults to "https://basemaps.cartocdn.com/gl/voyager-gl-style/style.json".

center	Numeric vector of length 2 (<code>c(lng, lat)</code>), or named list (<code>list(lng = ..., lat = ...)</code>), or NULL (default). Initial map center. If NULL and <code>input_osrm</code> is a <code>.osm.pbf</code> file, attempts to auto-center on the PBF extent. Priority is given to a fast pure R header parser and <code>osmium fileinfo</code> (fast); otherwise estimates the extent by sampling a small number of features via <code>sf::st_read()</code> (for example, reading with a <code>LIMIT 10</code> query).
zoom	Numeric. Initial zoom level. If NULL (default) and center is auto-detected from PBF, defaults to 9. Otherwise uses map default.
autozoom	Logical. Whether to enable auto-zoom by default. Defaults to TRUE.
update_while_drag	Logical. Whether to enable live tracking mode by default (updates route while dragging). Defaults to FALSE.
debug	Logical. Whether to enable debug mode (prints OSRM requests to console). Defaults to FALSE.

Details

The function checks for optional dependencies `shiny`, `mapgl`, `osrm`, `sf`, and `DT`. If missing, it prompts the user to install them.

It attempts to detect an active OSRM server. If an OSRM job process (from `osrm_start()`) is passed, it uses that configuration. If a path is passed, it will start a temporary server for the session.

Value

No return value; launches a Shiny Gadget.

Examples

```
if (interactive()) {
  # 1. Auto-detect running server (errors if none running):
  osrm_gui()

  # 2. Connect to specific port:
  # osrm_gui(port = 5001)

  # 3. Start from a graph file (auto-center on PBF):
  # osrm_gui("berlin.osrm.mldgr")

  # 4. Start from PBF with auto-center:
  # osrm_gui("berlin.osm.pbf")

  # 5. Explicit center and zoom:
  # osrm_gui(port = 5001, center = c(13.4, 52.5), zoom = 12)

  # 6. Use an existing process (must specify port):
  # srv <- osrm_start("graph.osrm.mldgr", port = 6000)
  # osrm_gui(srv, port = 6000)

  # 7. Enable debug mode:
  # osrm_gui(debug = TRUE)
```

```
}
```

```
osrm_install
```

```
Install OSRM Backend Binaries
```

Description

[Stable]

Downloads and installs pre-compiled binaries for the OSRM backend from the official GitHub releases. The function automatically detects the user's operating system and architecture to download the appropriate files. Only the latest v5 release (v5.27.1), v6.0.0, v26.4.0 and v26.4.1 were manually tested and are known to work well; other releases available on GitHub can be installed but are not guranteed to function correctly.

Usage

```
osrm_install(
  version = "latest",
  dest_dir = NULL,
  force = FALSE,
  path_action = c("session", "project", "none"),
  quiet = FALSE,
  check_tested = TRUE
)
```

Arguments

version	A string specifying the OSRM version tag to install. Defaults to "latest". Use "latest" to automatically find the most recent stable version (internally calls <code>osrm_check_latest_version()</code>). Versions other than v5.27.1, v6.0.0, v26.4.0, v26.4.1, and v26.5.0 will trigger a warning but are still attempted if binaries are available.
dest_dir	A string specifying the directory where OSRM binaries should be installed. If NULL (the default), a user-friendly, persistent location is chosen via <code>tools::R_user_dir("osrm.backend", which = "cache")</code> , and the binaries are installed into a subdirectory named after the OSRM version (e.g. <code>.../cache/v26.4.1</code>).
force	A logical value. If TRUE, reinstall OSRM even if it's already found in <code>dest_dir</code> . If FALSE (default), the function will stop if an existing installation is detected.
path_action	A string specifying how to handle the system PATH. One of: <ul style="list-style-type: none"> "session" (default): Adds the OSRM bin directory to the PATH for the current R session only. "project": Modifies the <code>.Rprofile</code> in the current project to set the PATH for all future sessions in that project. "none": Does not modify the PATH.

quiet	A logical value. If TRUE, suppresses installer messages and warnings. Defaults to FALSE.
check_tested	A logical value. If TRUE (default), the function issues a warning if the requested OSRM version has not been explicitly validated by the package maintainers. If FALSE, it issues a status message instead.

Details

The function performs the following steps:

1. Queries the GitHub API to find the specified release of Project-OSRM/osrm-backend.
2. Identifies the correct binary (.tar.gz archive) for the user's OS (Linux, macOS, or Windows) and architecture (x64, arm64).
3. Downloads the archive to a temporary location.
4. Extracts the archive and locates the OSRM executables (e.g., osrm-routed, osrm-extract).
5. Copies these executables to a local directory (defaults to `file.path(tools::R_user_dir("osrm.backend"), which = "source")`).
6. Downloads the matching Lua profiles from the release tarball and installs them alongside the binaries.
7. Optionally modifies the PATH environment variable for the current session or project.

macOS users should note that upstream OSRM v6.x (and newer) binaries are built for macOS 15.0 (Sequoia, Darwin 24.0.0) or newer. `osrm_install()` automatically blocks v6+ installs on older macOS releases and, when `version = "latest"`, selects the most recent v5 build instead while warning about the requirement. Warnings include both the marketing version and Darwin kernel so you'll see messages like `macOS 13 Ventura [Darwin 22.6.0]`.

When installing OSRM v6.x or newer for Windows, the upstream release omits the Intel Threading Building Blocks (TBB) runtime and a compatible bz2 DLL. To keep the executables runnable out of the box, `osrm_install()` fetches TBB from [oneTBB v2022.3.0](#) and the BZip2 runtime from [bzip2-windows v1.0.8.0](#), verifying their SHA-256 checksums before extraction. Without these extra libraries, the OSRM v6+ binaries shipped for Windows cannot start.

On macOS, OSRM v6.x and newer binaries also miss the bundled TBB runtime. The installer reuses the libraries from release v5.27.1 to keep the binaries functional and patches their `libbz2` linkage using `install_name_tool` so that they load the system-provided BZip2 runtime.

Power users (including package authors running cross-platform tests) can override the auto-detected platform by setting the R options `osrm.backend.override_os` and `osrm.backend.override_arch` (e.g., `options(osrm.backend.override_os = "linux", osrm.backend.override_arch = "arm64")`) before calling `osrm_install()`. Overrides allow requesting binaries for any OS and CPU combination that exists on the GitHub releases.

Value

The path to the installation directory.

Examples

```

if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  old <- setwd(tempdir())
  on.exit(setwd(old), add = TRUE)

  # Install the default stable version and set PATH for this session
  install_dir <- osrm_install(path_action = "session", quiet = TRUE)

  # Install for a project non-interactively (e.g., in a script)
  osrm_install(path_action = "project", quiet = TRUE, force = TRUE)

  # Clean up the project's .Rprofile and uninstall binaries
  osrm_clear_path(quiet = TRUE)
  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
}

```

osrm_partition

Partition OSRM Graph for Multi-Level Dijkstra (MLD)

Description**[Stable]**

Run the osrm-partition tool to partition an OSRM graph for the MLD pipeline. After running, a companion `<base>.osrm.partition` file must exist to confirm success.

Usage

```

osrm_partition(
  input_osrm,
  threads = 8L,
  verbosity = c("INFO", "NONE", "ERROR", "WARNING", "DEBUG"),
  balance = 1.2,
  boundary = 0.25,
  optimizing_cuts = 10L,
  small_component_size = 1000L,
  max_cell_sizes = c(128, 4096, 65536, 2097152),
  quiet = FALSE,
  verbose = FALSE,
  spinner = TRUE,
  echo_cmd = FALSE
)

```



```
profile <- osrm_find_profile("car.lua")

extract_job <- osrm_extract(
  input_osm = tmp_pbf,
  profile = profile,
  overwrite = TRUE,
  threads = 1L
)

partition_job <- osrm_partition(extract_job, threads = 1L, verbose = TRUE)
partition_job$osrm_job_artifact

osrm_uninstall(
  dest_dir = install_dir,
  clear_path = TRUE,
  force = TRUE,
  quiet = TRUE
)
unlink(osrm_dir, recursive = TRUE)
}
```

osrm_prepare_graph *Prepare OSRM Graph for Routing (Extract + Partition/Contract)*

Description

[Stable]

High-level wrapper that first runs `osrm-extract` on an OSM file to produce the base `.osrm` graph, then prepares it for routing via either the MLD pipeline (`osrm-partition` + `osrm-customize`) or the CH pipeline (`osrm-contract`).

Usage

```
osrm_prepare_graph(
  input_osm,
  profile = osrm_find_profile("car.lua"),
  threads = 8L,
  overwrite = FALSE,
  verbosity = c("INFO", "NONE", "ERROR", "WARNING", "DEBUG"),
  data_version = NULL,
  small_component_size = 1000L,
  with_osm_metadata = FALSE,
  parse_conditional_restrictions = FALSE,
  location_dependent_data = NULL,
  disable_location_cache = FALSE,
  dump_nbg_graph = FALSE,
```

```

    algorithm = c("mld", "ch"),
    balance = 1.2,
    boundary = 0.25,
    optimizing_cuts = 10L,
    max_cell_sizes = c(128, 4096, 65536, 2097152),
    quiet = FALSE,
    verbose = FALSE,
    spinner = TRUE,
    echo_cmd = FALSE
)

```

Arguments

input_osm	A string. Path to the input OSM file (.osm, .osm.bz2, or .osm.pbf) or a directory containing exactly one OSM file with a supported extension.
profile	A string. Path to the OSRM Lua profile (e.g. returned by <code>osrm_find_profile("car.lua")</code>).
threads	An integer. Number of threads for extract and partition/contract; default 8.
overwrite	A logical. If FALSE, stops if any existing .osrm* files matching the base name are found alongside input_osm. Set to TRUE to overwrite them.
verbosity	A string. Log verbosity for extract/partition/contract (one of "NONE", "ERROR", "WARNING", "INFO", "DEBUG", "VERBOSE", "DETAILED"); default "INFO".
data_version	A string or NULL. Passed to osrm-extract via -d; default NULL.
small_component_size	An integer. For extract & partition; default 1000.
with_osm_metadata	A logical. Adds --with-osm-metadata during extract; default FALSE.
parse_conditional_restrictions	A logical. Adds --parse-conditional-restrictions; default FALSE.
location_dependent_data	A string or NULL. Path to GeoJSON for extract; default NULL.
disable_location_cache	A logical. Adds --disable-location-cache; default FALSE.
dump_nbg_graph	A logical. Adds --dump-nbg-graph; default FALSE.
algorithm	A string. One of "MLD" (default) or "CH" (case-insensitive).
balance	A numeric. Balance for osrm-partition; default 1.2.
boundary	A numeric. Boundary percentage for osrm-partition; default 0.25.
optimizing_cuts	An integer. Optimizing cuts for osrm-partition; default 10.
max_cell_sizes	A numeric vector. Max cell sizes for osrm-partition; default c(128, 4096, 65536, 2097152).
quiet	A logical. Master switch that suppresses package messages and process output when TRUE; default FALSE.
verbose	A logical. When TRUE and quiet = FALSE, streams stdout and stderr from the underlying processx::run calls.

spinner	A logical. When TRUE and quiet = FALSE, shows a spinner instead of live logs; default TRUE.
echo_cmd	A logical. When TRUE and quiet = FALSE, prints each command before running; default FALSE.

Value

An object of class `osrm_job` with the following elements:

osrm_job_artifact The path to the final routing-ready graph file (`.osrm.hmgr` for CH or `.osrm.mldgr` for MLD).

osrm_working_dir The directory containing all OSRM files.

logs A list of processx: :run results for each stage: extract, partition/contract, and customize (if MLD).

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # Prepare a routing-ready graph with the default MLD pipeline
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)

  graph <- osrm_prepare_graph(
    input_osm = tmp_pbf,
    overwrite = TRUE,
    threads = 1L,
    algorithm = "mld"
  )
  graph$osrm_job_artifact

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}
```

osrm_servers

*List OSRM servers***Description****[Stable]**

Lists osrm-routed processes. By default, it returns a snapshot of servers started by the current R session (registered via `osrm_start_server()` or `osrm_start()`). You can optionally list all osrm-routed processes running on the system, including those started by other sessions or manually.

You can stop a server by passing its id, port, or pid to `osrm_stop()`.

Usage

```
osrm_servers(include_all = FALSE, output = c("data.frame", "list"))
```

Arguments

include_all	Logical; if TRUE, scans the system process table for all osrm-routed processes, including those not started by this package in the current session. Default is FALSE.
output	Character string specifying the return format. Either "data.frame" (the default) which returns a tabular summary with a custom print method, or "list" which returns a detailed list of server metadata objects.

Value

If `output = "data.frame"`, returns a `data.frame` (class `osrm_server_list`) of OSRM job processes with columns: `id`, `pid`, `port`, `algorithm`, `started_at`, `alive`, `has_handle`, `log`, `input_osm`, `center_lon`, `center_lat`. External servers will have `id` prefixed with `sys-` and `log` set to `<external>`. If `output = "list"`, returns a named list of server metadata.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # copy example OSM PBF into a temporary workspace to avoid polluting pkg data
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  graph <- osrm_prepare_graph(tmp_pbf, overwrite = TRUE, threads = 1L)
```

```

srv <- osrm_start_server(graph$osrm_job_artifact, port = 6000, threads = 1L)
osrm_servers()
osrm_stop(srv)

osrm_uninstall(
  dest_dir = install_dir,
  clear_path = TRUE,
  force = TRUE,
  quiet = TRUE
)
unlink(osrm_dir, recursive = TRUE)
}

```

osrm_start

Start an OSRM Server with Automatic Setup

Description

[Stable]

A high-level, "one-shot" function to start an OSRM server that automatically handles OSRM installation and graph preparation. This is the recommended function for most users to get a server running quickly with minimal steps.

Usage

```

osrm_start(
  path,
  algorithm = c("mld", "ch"),
  force_rebuild = FALSE,
  show_progress = TRUE,
  quiet = FALSE,
  verbose = FALSE,
  ...
)

```

Arguments

path	A string. Path to the input data. Can be one of: <ul style="list-style-type: none"> • A path to an OSM file (e.g., /path/to/data.osm.pbf). • A path to a directory containing OSRM graph files or an OSM file. • A direct path to a final graph file (.osrm.mldgr or .osrm.hmgr).
algorithm	A string specifying the routing algorithm to use for graph preparation, either "mld" (Multi-Level Dijkstra, default) or "ch" (Contraction Hierarchies). This is only used when osrm_prepare_graph is automatically called.

<code>force_rebuild</code>	A logical value. If TRUE, force the rebuilding of the OSRM graph even if it already exists.
<code>show_progress</code>	A logical value. If TRUE (default), show a progress spinner during graph preparation.
<code>quiet</code>	A logical value. If TRUE, suppresses installer messages and warnings. Defaults to FALSE.
<code>verbose</code>	A logical. If FALSE (default), suppresses detailed console output from backend commands. If TRUE, shows all output, which is useful for debugging.
<code>...</code>	Additional arguments passed on to <code>osrm_prepare_graph()</code> (e.g., <code>overwrite = TRUE</code>) and <code>osrm_start_server()</code> (e.g., <code>port = 5001</code>).

Details

This function is designed for convenience and automates the entire setup process. By default, it is not verbose and only prints high-level status messages.

1. **Check for OSRM Installation:** It first verifies if the `osrm`-routed executable is available in the system's PATH. If not, it automatically calls `osrm_install()` to download and install the latest stable version.
2. **Process Input Path and Prepare Graph:** The function intelligently handles the path argument to find or create the necessary graph files. If the graph files do not exist, it automatically runs `osrm_prepare_graph()` to build them, which may take some time.
3. **Start Server:** Once the graph files are located or prepared, it launches the `osrm`-routed server and prints a confirmation message with the server's PID and port.

For advanced users or for debugging, set `verbose = TRUE` to see the detailed console output from the installation and graph preparation steps. For full manual control, use the lower-level functions like `osrm_prepare_graph()` and `osrm_start_server()` directly.

Value

An OSRM job process (an `osrm_server` object inheriting from `processx::process`) for the running server.

See Also

[osrm_stop\(\)](#), [osrm_start_server\(\)](#) for manual server startup.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # copy example OSM PBF into a temporary workspace to avoid polluting pkg data
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
}
```

```

osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
dir.create(osrm_dir, recursive = TRUE)
local_pbf <- file.path(osrm_dir, "cur.osm.pbf")
file.copy(from = pbf_path, to = local_pbf, overwrite = TRUE)

# Start the server with one command.
# It will quietly install OSRM and prepare the graph if needed.
osrm_process <- osrm_start(local_pbf)

# Stop the server when done.
osrm_stop()

# To see all backend logs during setup, use verbose = TRUE
osrm_process_verbose <- osrm_start(local_pbf, verbose = TRUE)
osrm_stop()

osrm_uninstall(
  dest_dir = install_dir,
  clear_path = TRUE,
  force = TRUE,
  quiet = TRUE
)
unlink(osrm_dir, recursive = TRUE)
}

```

osrm_start_server	<i>Start an OSRM MLD/CH server with osrm-routed</i>
-------------------	---

Description

[Stable]

Launches an osrm-routed process pointing at a localized OSRM graph (either .osrm.mldgr for MLD or .osrm.hsgsr for CH/CoreCH).

Usage

```

osrm_start_server(
  osrm_path,
  version = FALSE,
  help = FALSE,
  verbosity = c("INFO", "ERROR", "WARNING", "NONE", "DEBUG"),
  trial = FALSE,
  ip = "0.0.0.0",
  port = 5001L,
  threads = 8L,
  shared_memory = FALSE,
  memory_file = NULL,
  mmap = FALSE,

```

```

dataset_name = NULL,
algorithm = NULL,
max_viaroute_size = 500L,
max_trip_size = 100L,
max_table_size = 100L,
max_matching_size = 100L,
max_nearest_size = 100L,
max_alternatives = 3L,
max_matching_radius = -1L,
quiet = FALSE,
verbose = FALSE,
echo_cmd = FALSE,
input_osm = NULL
)

```

Arguments

osrm_path	Character(1). Path to the .osrm.mldgr or .osrm.hsgr file
version	Logical; if TRUE, prints version and exits
help	Logical; if TRUE, prints help and exits
verbosity	Character; one of "NONE", "ERROR", "WARNING", "INFO", "DEBUG"
trial	Logical or integer; if TRUE or >0, quits after initialization (default: FALSE)
ip	Character; IP address to bind (default: "0.0.0.0")
port	Integer; TCP port to listen on (default: 5001). The function checks if this port is already in use by another running OSRM server (even from another session) and will stop with an error if a conflict is detected.
threads	Integer; number of worker threads (default: 8)
shared_memory	Logical; load graph from shared memory (default: FALSE)
memory_file	Character or NULL; DEPRECATED (behaves like mmap)
mmap	Logical; memory-map data files (default: FALSE)
dataset_name	Character or NULL; name of shared memory dataset
algorithm	Character or NULL; one of "MLD", "CH", or "CoreCH" (case-insensitive). If NULL (default), auto-selected based on file extension.
max_viaroute_size	Integer (default: 500)
max_trip_size	Integer (default: 100)
max_table_size	Integer (default: 100)
max_matching_size	Integer (default: 100)
max_nearest_size	Integer (default: 100)
max_alternatives	Integer (default: 3)

max_matching_radius	Integer; use -1 for unlimited (default: -1)
quiet	Logical; when TRUE, suppresses package messages.
verbose	Logical; when TRUE, routes server stdout and stderr to the R console for live debugging. Note: This can cause deadlocks in tight loops if R is busy. Defaults to FALSE, which writes logs to a temporary file.
echo_cmd	Logical; echo command line to console before launch (default: FALSE)
input_osm	Character or NULL; path to the original OSM input file (for tracking purposes). This parameter is typically used internally by <code>osrm_start()</code> to record the source data.

Details

The server's standard output and error streams are handled via temporary files by default to prevent deadlocks in R's single-threaded environment. This ensures reliable operation while preserving logs for debugging startup failures.

To customize logging behavior, you can use the following approaches:

- **Default (Temp File):** Logs are written to a temporary file. This prevents deadlocks while keeping logs available for debugging.
- **Verbose Mode:** Set `verbose = TRUE` to display logs directly in the R console. Note: This can cause deadlocks in tight loops if R is busy.
- **Custom Log File:** Set the `osrm.server.log_file` option to redirect output to a specific file: `options(osrm.server.log_file = "path/to/osrm.log")`

Note: List specifications (e.g., ``list(stdout = "...", stderr = "...)``) are deprecated and will fall back to the default temporary file behavior.

You can override the osrm-routed executable via `options(osrm.routed.exec = "/full/path/to/osrm-routed")`.

Value

An OSRM job process (an `osrm_server` object inheriting from `processx::process`) for the running server (also registered internally).

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # Build a graph then launch an OSRM server on a custom port
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
}
```

```
file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)

graph <- osrm_prepare_graph(
  input_osm = tmp_pbf,
  overwrite = TRUE,
  threads = 1L,
  algorithm = "mld"
)

server <- osrm_start_server(
  osrm_path = graph$osrm_job_artifact,
  port = 6000,
  threads = 1L
)

# Later, stop the server again
osrm_stop(server)

osrm_uninstall(
  dest_dir = install_dir,
  clear_path = TRUE,
  force = TRUE,
  quiet = TRUE
)
unlink(osrm_dir, recursive = TRUE)
}
```

osrm_stop

Stop an OSRM Server

Description

[Stable]

Terminates an osrm-routed process launched by `osrm_start()` or `osrm_start_server()`. Can also stop external servers by PID or ID.

Usage

```
osrm_stop(
  server = NULL,
  id = NULL,
  port = NULL,
  pid = NULL,
  wait = 1000L,
  quiet = FALSE
)
```

Arguments

server	Optional OSRM job process (an <code>osrm_server</code> object inheriting from <code>processx::process</code>) returned by <code>osrm_start_server()</code> .
id	Optional character id from <code>osrm_servers()</code> .
port	Optional integer TCP port.
pid	Optional integer process id.
wait	Integer milliseconds to wait for clean shutdown (default 1000).
quiet	Logical; suppress messages (default FALSE).

Details

This function provides a flexible way to stop a running OSRM process. If no arguments are specified, it defaults to stopping the most recently started server that is still alive in the current session.

You can also stop a specific server by providing:

- The OSRM job process (a `processx::process` object) returned by `osrm_start()` or `osrm_start_server()`.
- The server's id, port, or pid (use `osrm_servers()` to find these).

Advanced Use: You can stop an external osrm-routed process (one not started by the current R session) by passing its PID, or by finding it via `osrm_servers(include_all = TRUE)` and passing its id or port. This requires permission to signal the process.

Value

A list with fields id, pid, port, stopped (logical).

See Also

[osrm_start\(\)](#), [osrm_servers\(\)](#), [osrm_stop_all\(\)](#)

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # copy example OSM PBF into a temporary workspace to avoid polluting pkg data
  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  graph <- osrm_prepare_graph(tmp_pbf, overwrite = TRUE, threads = 1L)

  srv <- osrm_start_server(graph$osrm_job_artifact, port = 6000, threads = 1L)
```

```

# Stop by passing the process object
osrm_stop(srv)

# Or stop by port after the process is registered
osrm_stop(port = 6000)

osrm_uninstall(
  dest_dir = install_dir,
  clear_path = TRUE,
  force = TRUE,
  quiet = TRUE
)
unlink(osrm_dir, recursive = TRUE)
}

## Not run:
# Advanced: Stop an external server by PID
# 1. Find the PID of an external server
srvs <- osrm_servers(include_all = TRUE)
# 2. Stop it by PID
if (nrow(srvs) > 0) {
  osrm_stop(pid = srvs$pid[1])
}

# Or stop by its external ID (e.g., "sys-12345")
osrm_stop(id = "sys-12345")

## End(Not run)

```

osrm_stop_all

Stop all running OSRM servers started via this package

Description

[Stable]

Usage

```
osrm_stop_all(include_all = FALSE)
```

Arguments

include_all A logical value. If TRUE, stop all OSRM servers found running on the system, including those not started by the current R session. Requires ps and jsonlite packages. Defaults to FALSE.

Value

The number of servers attempted.

Examples

```

if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  pbf_path <- system.file("extdata/cur.osm.pbf", package = "osrm.backend")
  osrm_dir <- file.path(tempdir(), paste0("osrm-", Sys.getpid()))
  dir.create(osrm_dir, recursive = TRUE)
  tmp_pbf <- file.path(osrm_dir, "cur.osm.pbf")
  file.copy(from = pbf_path, to = tmp_pbf, overwrite = TRUE)
  graph <- osrm_prepare_graph(tmp_pbf, overwrite = TRUE, threads = 1L)

  srv <- osrm_start_server(graph$osrm_job_artifact, port = 6000, threads = 1L)
  stopped <- osrm_stop_all()
  stopped

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
  unlink(osrm_dir, recursive = TRUE)
}

```

osrm_uninstall

Uninstall OSRM Backend Binaries

Description**[Stable]**

Removes the OSRM backend binaries and optionally clears the PATH configuration from the project's .Rprofile.

Usage

```

osrm_uninstall(
  dest_dir = NULL,
  clear_path = TRUE,
  quiet = FALSE,
  all = FALSE,
  force = FALSE
)

```

Arguments

dest_dir	A string specifying the directory from which to remove OSRM binaries. If NULL (the default), the function looks for an installation in the per-version subdirectories inside <code>tools::R_user_dir("osrm.backend", which = "cache")</code> and removes it. When multiple versions are installed, interactive sessions that are not quiet will be prompted (with a numbered menu and 0 to cancel) to choose a directory; otherwise, <code>dest_dir</code> must be supplied. Ignored if <code>all = TRUE</code> .
clear_path	A logical value. If TRUE (default), also removes the PATH configuration from the project's <code>.Rprofile</code> by calling <code>osrm_clear_path()</code> .
quiet	A logical value. If TRUE, suppresses informational messages and confirmation prompts. Defaults to FALSE.
all	A logical value. If TRUE, removes all OSRM installations found in the default cache directory. Will prompt for confirmation unless <code>force = TRUE</code> . Defaults to FALSE. When TRUE, the <code>dest_dir</code> parameter is ignored.
force	A logical value. If TRUE, skips all confirmation prompts, enabling non-interactive usage. Defaults to FALSE.

Value

TRUE if one or more directories were successfully removed, and FALSE otherwise.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  # Install OSRM temporarily
  install_dir <- osrm_install(path_action = "session", quiet = TRUE)

  # Uninstall that specific version and clear PATH changes
  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )

  # If multiple installs exist, remove them all
  osrm_uninstall(all = TRUE, force = TRUE, quiet = TRUE)
}
```

Description

[Stable]

Resolves the osrm-routed executable available on the current PATH (or the override provided via `options(osrm.routed.exec)`). Runs `osrm-routed --version` to verify availability, then prints the directory containing the executable together with the backend version reported by `osrm-routed` so you know what will be used in the current session.

Usage

```
osrm_which(quiet = FALSE)
```

Arguments

<code>quiet</code>	Logical; if FALSE (default), prints information about the located installation. If TRUE, suppresses printed output and only returns the information as a list.
--------------------	--

Value

A list with components `executable` (full path to `osrm-routed`), `directory` (its parent folder), `osrm_version` (character vector of non-empty lines emitted by `osrm-routed --version`), and the raw process `system::run` result in `logs`.

Examples

```
if (identical(Sys.getenv("OSRM_EXAMPLES"), "true")) {
  install_dir <- osrm_install(
    version = "latest",
    path_action = "session",
    quiet = TRUE
  )

  # check which OSRM installation will be used
  osrm_which()

  osrm_uninstall(
    dest_dir = install_dir,
    clear_path = TRUE,
    force = TRUE,
    quiet = TRUE
  )
}
```

Index

osrm_check_available_versions, 2
osrm_check_latest_version, 3
osrm_check_latest_version(), 16
osrm_cleanup, 3
osrm_clear_path, 5
osrm_contract, 6
osrm_customize, 8
osrm_extract, 10
osrm_find_profile, 12
osrm_get_server_profile, 13
osrm_gui, 14
osrm_install, 16
osrm_partition, 18
osrm_prepare_graph, 20
osrm_servers, 23
osrm_servers(), 14, 30
osrm_start, 24
osrm_start(), 23, 28, 30
osrm_start_server, 26
osrm_start_server(), 23, 25
osrm_stop, 29
osrm_stop(), 23, 25
osrm_stop_all, 31
osrm_stop_all(), 30
osrm_uninstall, 32
osrm_which, 33