Package 'openmpp'

February 20, 2026

Title Programmatic Interface to 'OpenM++'

Version 0.0.2

Description A programmatic interface to the 'OpenM++' microsimulation platform (<https://openmpp.org>).
The primary goal of this package is to wrap the 'OpenM++' Web Service (OMS) to provide 'OpenM++' users a programmatic interface for the R language.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3.9000

Depends R (>= 4.1.0)

Imports cli, curl, glue, httr2, jsonlite, purrr, R6, readr, rlang, tibble, tidyr, tidyselect

URL https://github.com/mattwarkentin/openmpp, https://mattwarkentin.github.io/openmpp/

BugReports https://github.com/mattwarkentin/openmpp/issues

Collate 'API-CPAC.R' 'API-Custom.R' 'API-Local.R' 'Utils.R' 'ClassModel.R' 'ClassModelRun.R' 'ClassModelRunSet.R' 'ClassWorkset.R' 'RunOptions.R' 'Scenario.R' 'oms-Administrative.R' 'oms-DownloadModels.R' 'oms-ModelExtras.R' 'oms-ModelMetadata.R' 'oms-ModelRunResults.R' 'oms-ModelWorksetMetadata.R' 'oms-ModelingTaskMetadata.R' 'oms-ReadMicrodata.R' 'oms-ReadOutputTables.R' 'oms-ReadParameters.R' 'oms-RunModels.R' 'oms-Services.R' 'oms-UpdateModelProfile.R' 'oms-UpdateModelRuns.R' 'oms-UpdateModelWorkset.R' 'oms-UpdateModelingTasks.R' 'oms-UploadModels.R' 'oms-UserFiles.R' 'oms-UserSettings.R' 'openmpp-package.R' 'zzz.R'

Suggests gh, pingr, sys, testthat (>= 3.0.0), withr

Config/testthat/edition 3

NeedsCompilation no

Author Matthew T. Warkentin [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-8730-3511>),
John M. Hutchinson [ctb]

Maintainer Matthew T. Warkentin <matthew.warkentin@ucalgary.ca>

Repository CRAN

Date/Publication 2026-02-20 21:30:02 UTC

Contents

admin_models_refresh *OpenM++ Administrative Tasks*

Description

Functions for performing administrative tasks. More information about the administrative API endpoints can be found at here.

Usage

```
admin_models_refresh()

admin_models_close()

admin_database_close(model)

admin_model_delete(model)

admin_database_open(path)

admin_database_cleanup(path, name = NULL, digest = NULL)

admin_cleanup_logs()

admin_cleanup_log(name)

admin_jobs_pause(pause)

admin_jobs_pause_all(pause)

admin_service_shutdown()
```

Arguments

`model`	Model digest or model name.
`path`	Path to model database file relative to the `models/bin` folder. For example, the name of the model with a `".sqlite"` extension.
`name`	Model name. Optional for `admin_database_cleanup()`.
`digest`	Model digest. Optional for `admin_database_cleanup()`.
`pause`	Logical. Whether to pause or resume model runs queue processing.

Details

To find the relative `path` to a database file for cleanup with `admin_database_cleanup(path)` or opening a database file with `admin_database_open(path)`, users can run the `get_models_list()` function to retrieve the list of model information and find the `DbPath` list item. `DbPath` is the relative path to the database file. You must replace the forward-slashes in the relative path with asterisks.

Value

A `list` or nothing, invisibly.

Examples

```
## Not run:
use_OpenMpp_local()
admin_models_refresh()

## End(Not run)
```

`create_scenario` *Manage Model Scenarios*

Description

Manage Model Scenarios

Usage

```
create_scenario(model, name, base = NULL)

create_workset(model, name, base = NULL)
```

Arguments

`model`	Model name or digest.
`name`	New scenario name.
`base`	Base run digest (optional).

Value

Nothing, invisibly.

Examples

```
## Not run:
use_OpenMpp_local()
create_scenario("RiskPaths", "NewScenario")

## End(Not run)
```

`create_task` *Update Modeling Tasks*

Description

Functions for updating modeling tasks. More information about these API endpoints can be found at here.

Usage

```
create_task(data)

update_task(data)

delete_task(model, task)
```

Arguments

`data`	Data used for the body of the request.
`model`	Model digest or model name.
`task`	Modeling task.

Value

A `list`, `tibble`, or nothing (invisibly).

Examples

```
## Not run:
use_OpenMpp_local()
create_task(list(
  ModelName = "RiskPaths",
  ModelDigest = "d976aa2fb999f097468bb2ea098c4daf",
  Name = "NewTask",
  Set = list("Default")
))
delete_task("RiskPaths", "NewTask")

## End(Not run)
```

`delete_model_run` *Update Model Runs*

Description

Functions to update model runs. More information about these API endpoints can be found at here.

Usage

```
delete_model_run(model, run)

delete_run(model, run)

delete_model_runs(model)

delete_runs(model)
```

Arguments

`model`	Model digest or model name.
`run`	Model run digest, run stamp or run name, modeling task run stamp or task run name.

Value

A `list`, `tibble`, or nothing (invisibly).

Examples

```
## Not run:
use_OpenMpp_local()
delete_model_run("RiskPaths", "53300e8b56eabdf5e5fb112059e8c137")

## End(Not run)
```

`get_model` *Model Metadata*

Description

Functions to get the list of models and load a specific model definition. More information about these API endpoints can be found at here.

Usage

```
get_model(model)

get_models()

get_models_list()
```

Arguments

`model`	Model digest or model name.

Value

A `list` or `tibble`.

Examples

```
## Not run:
use_OpenMpp_local()
get_models()
get_models_list()
get_model("RiskPaths")

## End(Not run)
```

`get_model_lang_list` *Model Extras*

Description

Functions for retrieving extra model information, including language lists, word lists, profiles, and profile lists. More information about these API endpoints can be found at here.

Usage

```
get_model_lang_list(model)

get_model_word_list(model)

get_model_profile(model, profile)

get_model_profile_list(model)
```

Arguments

`model`	Model digest or model name.
`profile`	Profile name.

Value

A `list` from a JSON response object.

Examples

```
## Not run:
use_OpenMpp_local()
get_model_lang_list('RiskPaths')

## End(Not run)
```

`get_model_run` *Model Run Results Metadata*

Description

Functions to retrieve model run metadata and delete model runs. More information about these API endpoints can be found at here.

Usage

```
get_model_run(model, run)

get_run(model, run)

get_model_runs_list(model)

get_model_runs(model)

get_runs(model)

get_model_run_status(model, run)

get_run_status(model, run)

get_model_run_list_status(model, run)

get_model_run_status_first(model)

get_model_run_status_last(model)

get_model_run_status_compl(model)
```

Arguments

`model`	Model digest or model name.
`run`	Model run digest, run stamp or run name, modeling task run stamp or task run name.

Value

A `list`, `tibble`, or nothing (invisibly).

Examples

```
## Not run:
use_OpenMpp_local()
get_model_run("RiskPaths", "53300e8b56eabdf5e5fb112059e8c137")
get_run("RiskPaths", "53300e8b56eabdf5e5fb112059e8c137")
get_model_runs_list("RiskPaths")
get_model_runs("RiskPaths")

## End(Not run)
```

`get_model_task` *Modeling Task Metadata*

Description

Functions for getting creating, updating, retrieving, and deleting model tasks. More information about these API endpoints can be found at here.

Usage

```
get_model_task(model, task)

get_model_tasks_list(model)

get_model_tasks(model)

get_model_task_worksets(model, task)

get_model_task_hist(model, task)

get_model_task_status(model, task, run)

get_model_task_run_list_status(model, task, run)

get_model_task_run_first(model, task)

get_model_task_run_last(model, task)

get_model_task_run_compl(model, task)
```

Arguments

`model`	Model digest or model name.
`task`	Modeling task.
`run`	Model run digest, run stamp or run name, modeling task run stamp or task run name.

Value

A `list`, `tibble`, or nothing (invisibly).

Examples

```
## Not run:
use_OpenMpp_local()
get_model_tasks_list("RiskPaths")
get_model_tasks("RiskPaths")

## End(Not run)
```

`get_run_microdata` *Read Run Microdata*

Description

Functions for retrieving microdata from model runs. More information about these API endpoints can be found at here.

Usage

```
get_run_microdata(model, run, name)

get_run_microdata_csv(model, run, name)
```

Arguments

`model`	Model digest or model name.
`run`	Model run digest, run stamp or run name, modeling task run stamp, or task run name.
`name`	Microdata entity name.

Value

A `list` or `tibble`.

Examples

```
## Not run:
use_OpenMpp_local()
get_run_microdata_csv("RiskPaths", "53300e8b56eabdf5e5fb112059e8c137", "Person")

## End(Not run)
```

`get_run_table` *Read Output Tables*

Description

Functions for retrieving output tables from model runs. More information about these API endpoints can be found at here.

Usage

```
get_run_table(model, run, name)

get_run_table_csv(model, run, name)

get_run_table_acc_csv(model, run, name)

get_run_table_calc_csv(model, run, name, calc)

get_run_table_comparison_csv(model, run, name, compare, variant)
```

Arguments

`model`	Model digest or model name.
`run`	Model run digest, run stamp or run name, modeling task run stamp or task run name.
`name`	Output table name.
`calc`	Name of calculation. One of `"avg"`, `"sum"`, `"count"`, `"max"`, `"min"`, `"var"`, `"sd"`, `"se"`, or `"cv"`.
`compare`	Comparison to calculate. One of `"diff"`, `"ratio"`, or `"percent"`. Comparisons are for the base run relative to the variant run (i.e., for `"diff"` it is the difference of values represented as Variant - Base).
`variant`	Run digest, name, or stamp for the variant model run.

Value

A `list` or `tibble`.

Examples

```
## Not run:
use_OpenMpp_local()
get_run_table_csv("RiskPaths", "53300e8b56eabdf5e5fb112059e8c137", "T01_LifeExpectancy")

## End(Not run)
```

`get_service_config` *OpenM++ Model Run Jobs and Service State*

Description

Functions for retrieving or deleting service information. More information about these API endpoints can be found at here.

Usage

```
get_service_config()

get_service_state()

get_disk_use()

refresh_disk_use()

get_active_job_state(job)

get_queue_job_state(job)

get_hist_job_state(job)

set_queue_job_pos(pos, job)

delete_job_hist(job)
```

Arguments

`job`	Model run submission time stamp.
`pos`	Position.

Value

A `list` from a JSON response object, or nothing (invisibly).

Examples

```
## Not run:
use_OpenMpp_local()
get_service_config()
get_service_state()
get_disk_use()

## End(Not run)
```

`get_user_files` *Manage OpenM++ User Files*

Description

Functions for getting, setting, and deleting user file. More information about these API endpoints can be found at here.

Usage

```
get_user_files(ext = "*", path = "")

upload_user_files(path)

create_user_files_folder(path)

delete_user_files(path)

delete_user_files_all()
```

Arguments

`ext`	Comma-separated string of file extensions. Default is `"*"` which returns all files.
`path`	Optional file path. Default is `""` (empty) and returns the entire tree of user files.

Value

A `list` from JSON response object or nothing (invisibly).

Examples

```
## Not run:
use_OpenMpp_local()
get_user_files()

## End(Not run)
```

`get_user_views` *Manage OpenM++ User Settings*

Description

Functions for getting, setting, and deleting user settings. More information about these API endpoints can be found at here.

Usage

```
get_user_views(model)

set_user_views(model, data)

delete_user_views(model)
```

Arguments

`model`	Model digest or model name.
`data`	Data used for the body of the request.

Value

A `list` from JSON response object or nothing (invisibly).

Examples

```
## Not run:
use_OpenMpp_local()
get_user_views("RiskPaths")

## End(Not run)
```

`get_workset` *Model Workset Metadata*

Description

Functions for creating, copying, merging, retrieving, and deleting worksets. More information about these API endpoints can be found at here.

Usage

```
get_workset(model, set)

get_worksets_list(model)

get_worksets(model)

get_scenarios(model)

get_workset_status(model, set)

get_workset_status_default(model)
```

Arguments

`model`	Model digest or model name.
`set`	Name of workset (input set of model parameters).

Value

A `list`, `tibble`, or nothing (invisibly).

Examples

```
## Not run:
use_OpenMpp_local()
get_worksets("RiskPaths")
get_scenarios("RiskPaths")
get_workset("RiskPaths", "Default")

## End(Not run)
```

`get_workset_param` *Read Parameters*

Description

Functions for retrieving parameters from worksets or model runs. More information about these API endpoints can be found at here.

Usage

```
get_workset_param(model, set, name)

get_workset_param_csv(model, set, name)

get_run_param(model, run, name)

get_run_param_csv(model, run, name)
```

Arguments

`model`	Model digest or model name.
`set`	Name of workset (input set of model parameters).
`name`	Output table name.
`run`	Model run digest, run stamp or run name, modeling task run stamp or task run name.

Value

A `list` or `tibble`.

Examples

```
## Not run:
use_OpenMpp_local()
get_workset_param("RiskPaths", "Default", "AgeBaselinePreg1")

## End(Not run)
```

`initiate_model_download` *Download Model, Model Run Results, or Input Parameters*

Description

Functions to download model, model run results, or input parameters. More information about these API endpoints can be found at here.

Usage

```
initiate_model_download(model)

initiate_run_download(model, run)

initiate_workset_download(model, set)
```

```
delete_download_files(folder)

delete_download_files_async(folder)

get_download_log(name)

get_download_logs_model(model)

get_download_logs_all()

get_download_filetree(folder)
```

Arguments

`model`	Model digest or model name.
`run`	Model run digest, run stamp or run name, modeling task run stamp or task run name.
`set`	Name of workset (input set of model parameters).
`folder`	Download folder file name.
`name`	Output table name.

Value

Nothing, invisibly.

Examples

```
## Not run:
use_OpenMpp_local()
get_download_logs_model('RiskPaths')

## End(Not run)
```

`initiate_run_upload` *Upload Model Runs or Worksets*

Description

Functions to upload model run results or input parameters (worksets). More information about these API endpoints can be found at here.

Usage

```
initiate_run_upload(model, run, data)

initiate_workset_upload(model, set, data)

delete_upload_files(folder)

delete_upload_files_async(folder)

get_upload_log(name)

get_upload_logs_model(model)

get_upload_logs_all()

get_upload_filetree(folder)
```

Arguments

`model`	Model digest or model name.
`run`	Model run digest, run stamp or run name, modeling task run stamp or task run name.
`data`	Data used for the body of the request.
`set`	Name of workset (input set of model parameters).
`folder`	Upload folder file name.
`name`	Output table name.

Value

Nothing, invisibly.

Examples

```
## Not run:
use_OpenMpp_local()
get_upload_logs_all()

## End(Not run)
```

`is_compatible` *Are data frames compatible?*

Description

Are data frames compatible?

Usage

```
is_compatible(x, y)
```

Arguments

`x`	New parameters.
`y`	Old parameters

`load_model` *OpenM++ Model Class*

Description

OpenM++ Model Class

Usage

```
load_model(model)
```

Arguments

`model`	Model digest or model name.

Value

An `OpenMppModel` instance.

Public fields

`OpenMppType` OpenM++ object type (used for `print()` method).

`ModelDigest` Model digest.

`ModelName` Model name.

`ModelVersion` Model version.

`ModelMetadata` Model metadata.

`ParamsInfo` Input parameter information.

`TablesInfo` Output table information.

Active bindings

`ModelWorksets` Data frame of worksets.

`ModelScenarios` Data frame of scenarios.

`ModelRuns` Data frame of model runs.

`ModelTasks` Data frame of model tasks.

Methods

Public methods:

- `OpenMppModel$new()`
- `OpenMppModel$print()`

Method `new()`**:** Create a new OpenMppModel object.

Usage:

```
OpenMppModel$new(model)
```

Arguments:

`model` Model digest or name.

Returns: A new `OpenMppModel` object.

Method `print()`**:** Print a OpenMppModel object.

Usage:

```
OpenMppModel$print(...)
```

Arguments:

`...` Not currently used.

Returns: Self, invisibly.

Examples

```
## Not run:
use_OpenMpp_local()
load_model("RiskPaths")

## End(Not run)
```

`load_model_run` *OpenM++ ModelRun Class*

Description

OpenM++ ModelRun Class

Usage

```
load_model_run(model, run)

load_run(model, run)
```

Arguments

`model`	Model digest or model name.
`run`	Model run digest, run stamp or run name, modeling task run stamp or task run name.

Details

`load_run()` is an alias for `load_model_run()`.

Value

An `OpenMppModelRun` instance.

Super class

`openmpp::OpenMppModel` -> `OpenMppModelRun`

Public fields

`RunName` Run name.

`RunDigest` Run digest.

`RunStamp` Run stamp.

`RunMetadata` Run metadata.

`Parameters` Model run parameters.

`Tables` Model run output tables

`OpenMppType` OpenM++ object type (used for `print()`).

Active bindings

`RunStatusInfo` Run status information.

`RunStatus` Run status.

Methods

Public methods:

- `OpenMppModelRun$new()`
- `OpenMppModelRun$print()`
- `OpenMppModelRun$get_table()`
- `OpenMppModelRun$get_table_acc()`
- `OpenMppModelRun$get_table_calc()`
- `OpenMppModelRun$get_table_comparison()`
- `OpenMppModelRun$write_table()`
- `OpenMppModelRun$write_tables()`
- `OpenMppModelRun$get_log()`
- `OpenMppModelRun$write_log()`

Method `new()`: Create a new OpenMppModelRun object.

Usage:

```
OpenMppModelRun$new(model, run)
```

Arguments:

`model` Model digest or name.

`run` Run digest, run stamp, or run name.

Returns: A new `OpenMppModelRun` object.

Method `print()`**:** Print a `OpenMppModelRun` object.

Usage:

```
OpenMppModelRun$print(...)
```

Arguments:

`...` Not currently used.

Returns: Self, invisibly.

Method `get_table()`**:** Retrieve a table.

Usage:

```
OpenMppModelRun$get_table(name)
```

Arguments:

`name` Table name.

Returns: A `tibble`.

Method `get_table_acc()`**:** Retrieve a table with all accumulator values.

Usage:

```
OpenMppModelRun$get_table_acc(name)
```

Arguments:

`name` Table name.

Returns: A `tibble`.

Method `get_table_calc()`**:** Retrieve a table calculation.

Usage:

```
OpenMppModelRun$get_table_calc(name, calc)
```

Arguments:

`name` Table name.

`calc` Name of calculation. One of `"avg"`, `"sum"`, `"count"`, `"max"`, `"min"`, `"var"`, `"sd"`, `"se"`, or `"cv"`.

Returns: A `tibble`.

Method `get_table_comparison()`**:** Retrieve a table comparison.

Usage:

```
OpenMppModelRun$get_table_comparison(name, compare, variant)
```

Arguments:

`name` Table name.

`compare` Comparison to calculate. One of `"diff"`, `"ratio"`, or `"percent"`.

`variant` Run digest, name, or stamp for the variant model run.

Returns: A `tibble`.

Method `write_table()`**:** Write an output table to disk (CSV).

Usage:

```
OpenMppModelRun$write_table(name, file)
```

Arguments:

`name` Table name.

`file` File path.

Returns: Self, invisibly.

Method `write_tables()`**:** Write all output tables to disk (CSV).

Usage:

```
OpenMppModelRun$write_tables(dir)
```

Arguments:

`dir` Directory path.

Returns: Self, invisibly.

Method `get_log()`**:** Get console log for model run.

Usage:

```
OpenMppModelRun$get_log()
```

Returns: Self, invisibly.

Method `write_log()`**:** Write console log for model run to disk.

Usage:

```
OpenMppModelRun$write_log(dir)
```

Arguments:

`dir` Directory to save log file.

Returns: Self, invisibly.

Examples

```
## Not run:
use_OpenMpp_local()
load_model_run("RiskPaths", "53300e8b56eabdf5e5fb112059e8c137")
load_run("RiskPaths", "53300e8b56eabdf5e5fb112059e8c137")

## End(Not run)
```

`load_model_runs` *OpenM++ ModelRunSet Class*

Description

OpenM++ ModelRunSet Class

Usage

```
load_model_runs(model, runs)

load_runs(model, runs)
```

Arguments

`model`	Model name or digest.
`runs`	Character vector of model run names, digests, or stamps.

Details

`load_runs()` is an alias for `load_model_runs()`.

Value

An `OpenMppModelRunSet` instance.

Public fields

`ModelDigest` Model digest.

`ModelName` Model name.

`ModelVersion` Model version.

`OpenMppType` OpenM++ object type (used for `print()`).

`Tables` Model run output tables

Active bindings

`RunNames` Run names.

`RunDigests` Run digests.

`RunStamps` Run stamps.

`RunStatuses` Run statuses.

`RunMetadatas` Run metadatas.

Methods

Public methods:

- `OpenMppModelRunSet$new()`
- `OpenMppModelRunSet$print()`
- `OpenMppModelRunSet$get_table()`
- `OpenMppModelRunSet$get_table_calc()`
- `OpenMppModelRunSet$write_table()`
- `OpenMppModelRunSet$write_tables()`

Method `new()`: Create a new OpenMppModelRunSet object.

Usage:

```
OpenMppModelRunSet$new(model, runs)
```

Arguments:

`model` Model digest or name.

`runs` Run digests, run stamps, or run names.

Returns: A new `OpenMppModelRunSet` object.

Method `print()`: Print a `OpenMppModelRunSet` object.

Usage:

```
OpenMppModelRunSet$print(...)
```

Arguments:

`...` Not currently used.

Returns: Self, invisibly.

Method `get_table()`: Retrieve a table.

Usage:

```
OpenMppModelRunSet$get_table(name)
```

Arguments:

`name` Table name.

Returns: A `tibble`.

Method `get_table_calc()`: Retrieve a table calculation.

Usage:

```
OpenMppModelRunSet$get_table_calc(name, calc)
```

Arguments:

`name` Table name.

`calc` Name of calculation. One of `"avg"`, `"sum"`, `"count"`, `"max"`, `"min"`, `"var"`, `"sd"`, `"se"`, or `"cv"`.

Returns: A `tibble`.

Method `write_table()`: Write an output table to disk (CSV).

Usage:

`OpenMppModelRunSet$write_table(name, file)`

Arguments:

`name` Table name.

`file` File path.

Returns: Self, invisibly.

Method `write_tables()`**:** Write all output tables to disk (CSV).

Usage:

`OpenMppModelRunSet$write_tables(dir)`

Arguments:

`dir` Directory path.

Returns: Self, invisibly.

Examples

```
## Not run:
use_OpenMpp_local()
load_model_runs("RiskPaths", rep("53300e8b56eabdf5e5fb112059e8c137", 2))
load_runs("RiskPaths", rep("53300e8b56eabdf5e5fb112059e8c137", 2))

## End(Not run)
```

`load_workset` *OpenM++ Workset Class*

Description

OpenM++ Workset Class

Usage

```
load_workset(model, set)

load_scenario(model, set)
```

Arguments

`model`	Model digest or model name.
`set`	Name of workset (input set of model parameters).

Details

`load_scenario()` is an alias for `load_workset()`.

Value

An `OpenMppWorkset` instance.

Super class

`openmpp::OpenMppModel` -> `OpenMppWorkset`

Public fields

`WorksetName` Workset name.

`WorksetMetadata` Workset metadata.

`OpenMppType` OpenM++ object type (used for `print()`).

`Parameters` Workset parameters.

Active bindings

`ReadOnly` Workset read-only status.

`BaseRunDigest` Base run digest for input parameters.

Methods

Public methods:

- `OpenMppWorkset$new()`
- `OpenMppWorkset$print()`
- `OpenMppWorkset$set_base_digest()`
- `OpenMppWorkset$delete_base_digest()`
- `OpenMppWorkset$copy_params()`
- `OpenMppWorkset$delete_params()`
- `OpenMppWorkset$get_param()`
- `OpenMppWorkset$set_param()`
- `OpenMppWorkset$run()`

Method `new()`: Create a new OpenMppWorkset object.

Usage:

```
OpenMppWorkset$new(model, set)
```

Arguments:

`model` Model digest or name.

`set` Workset name.

Returns: A new `OpenMppWorkset` object.

Method `print()`: Print a `OpenMppWorkset` object.

Usage:

```
OpenMppWorkset$print(...)
```

Arguments:

`...` Not currently used.

Returns: Self, invisibly.

Method `set_base_digest()`**:** Set the base run digest.

Usage:

```
OpenMppWorkset$set_base_digest(base)
```

Arguments:

`base` Base run digest.

Returns: Self, invisibly.

Method `delete_base_digest()`**:** Delete the base run digest.

Usage:

```
OpenMppWorkset$delete_base_digest()
```

Returns: Self, invisibly.

Method `copy_params()`**:** Copy parameters from a base scenario.

Usage:

```
OpenMppWorkset$copy_params(names)
```

Arguments:

`names` Character vector of parameter names.

Returns: Self, invisibly.

Method `delete_params()`**:** Delete parameters from scenario.

Usage:

```
OpenMppWorkset$delete_params(names)
```

Arguments:

`names` Character vector of parameter names.

Returns: Self, invisibly.

Method `get_param()`**:** Retrieve a parameter.

Usage:

```
OpenMppWorkset$get_param(name)
```

Arguments:

`name` Parameter name.

Returns: A `tibble`.

Method `set_param()`**:** Set a parameter.

Usage:

```
OpenMppWorkset$set_param(name, data)
```

Arguments:

`name` Parameter name.

`data` New parameter data.

Returns: Self, invisibly.

Method `run()`**:** Initiate a model run for the model workset/scenario.

Usage:

```
OpenMppWorkset$run(
  name,
  opts = opts_run(),
  wait = FALSE,
  wait_time = 0.2,
  progress = TRUE
)
```

Arguments:

`name` Run name.

`opts` Run options. See `opts_run()` for more details.

`wait` Logical. Should we wait until the model run is done?

`wait_time` Number of seconds to wait between status checks.

`progress` Logical. Should a progress bar be shown?

Returns: Self, invisibly.

Examples

```
## Not run:
use_OpenMpp_local()
load_workset("RiskPaths", "Default")
load_scenario("RiskPaths", "Default")

## End(Not run)
```

`opts_run` *Run Configuration Options*

Description

Run Configuration Options

Usage

```
opts_run(
  SimulationCases = 5000,
  SubValues = 12,
  RunStamp = TimeStamp(),
  Opts = list(),
  ...
)

## S3 method for class 'OpenMppRunOpts'
print(x, ...)
```

Arguments

`SimulationCases`	Number of cases to simulate. Default is 5000.
`SubValues`	Number of sub values. Default is 12.
`RunStamp`	Run stamp. Default is generated based on the current time.
`Opts`	Additional options to pass to the `Opts` list component.
`...`	Any other run options.
`x`	Object to print.

Details

The default number of `SimulationCases` is low to enable rapid iteration but should be increased when running a model where the results are expected to be robust.

Examples

```
opts <- opts_run()
print(opts)
```

`run_model` *Run Models and Monitor Progress*

Description

More information about these API endpoints can be found at here.

Usage

```
run_model(data)

get_model_run_state(model, stamp)

get_run_state(model, stamp)

stop_model_run(model, stamp)

stop_run(model, stamp)
```

Arguments

`data`	Data used for the body of the POST request.
`model`	Model digest or name.
`stamp`	Model run stamp.

Value

A `list` or nothing, invisibly.

Examples

```
## Not run:
use_OpenMpp_local()
get_model_run_state("RiskPaths", "2025_01_28_21_00_48_385")

## End(Not run)
```

`set_workset_readonly` *Update Workset Metadata*

Description

Functions for creating, copying, merging, retrieving, and deleting worksets. More information about these API endpoints can be found at here.

Usage

```
set_workset_readonly(model, set, readonly)

new_workset(data)

merge_workset(workset)

update_workset_param_csv(workset, csv)

replace_workset(workset)

delete_workset(model, set)

delete_scenario(model, set)

delete_workset_param(model, set, name)

update_workset_param(model, set, name, data)

copy_param_run_to_workset(model, set, name, run)
```

```
merge_param_run_to_workset(model, set, name, run)

copy_param_workset_to_workset(model, set, name, from)

merge_param_workset_to_workset(model, set, name, from)

upload_workset_params(model, set, data)
```

Arguments

`model`	Model digest or model name.
`set`	Name of workset (input set of model parameters).
`readonly`	Boolean. Should workset be read-only?
`data`	Data used for the body of the request.
`workset`	Workset metadata.
`csv`	CSV file path.
`name`	Output table name.
`run`	Model run digest, run stamp or run name, modeling task run stamp or task run name.
`from`	Source workset name.

Value

A `list`, `tibble`, or nothing (invisibly).

Examples

```
## Not run:
use_OpenMpp_local()
set_workset_readonly("RiskPaths", "Default", TRUE)

## End(Not run)
```

`touch_model_profile` *Update Model Profile*

Description

Functions for creating, modifying, and deleting profiles and profile options. More information about these API endpoints can be found at here.

Usage

```
touch_model_profile(model, data)

delete_model_profile(model, profile)

set_model_profile_opt(model, profile, key, value)

delete_model_profile_opt(model, profile, key)
```

Arguments

`model`	Model digest or model name.
`data`	Data used for the body of the request.
`profile`	Profile name.
`key`	Option key.
`value`	Option value.

Value

A `list` from a JSON response object or nothing (invisibly).

Examples

```
## Not run:
use_OpenMpp_local()
touch_model_profile(
  "RiskPaths",
  list(Name = "profile1", Opts = list(Parameter.StartingSeed = "192"))
)
delete_model_profile("RiskPaths", "profile1")

## End(Not run)
```

`use_OpenMpp_CPAC` *OpenM++ CPAC API Connection*

Description

Register a connection to the OpenM++ Web Services (OMS) Application Programming Interface (API) running on the CPAC cloud-based server.

Usage

```
use_OpenMpp_CPAC(
  url = Sys.getenv("OPENMPP_CPAC_URL"),
  key = Sys.getenv("OPENMPP_CPAC_KEY"),
  ...
)
```

Arguments

`url`	URL for making API requests. See `Details` for more instructions.
`key`	API Key for making requests.
`...`	Not currently used.

Details

An API key is sensitive information and should not be shared. To avoid hard-coding your user name and password into your R scripts, we recommend declaring this information in your global or project-specific `.Renviron` file. The same approach may be used to declare your URL for making API requests. While the URL us typically not sensitive information, keeping all of this information in one place makes sense for consistency. For remote API connections to CPAC, set the following environment variables in your `.Renviron` files:

- `OPENMPP_CPAC_URL`: URL for making remote API requests.
- `OPENMPP_CPAC_KEY`: API key for making remote API requests.

Value

Nothing, invisibly. Behind-the-scenes, an instance of the `OpenMppCPAC` R6 class is created. This objects should not be accessed directly by the user, instead, the package internally uses these connections to communicate with the OpenM++ API.

Examples

```
## Not run:
use_OpenMpp_CPAC()

## End(Not run)
```

`use_OpenMpp_custom` *OpenM++ Custom API Connection*

Description

Register a custom connection to the OpenM++ Web Services (OMS) Application Programming Interface (API). A custom conneciton is typically comprised of two main tasks: (1) authentication and (2) building the request. To create your own custom connection, you simply need to create a function (`req`) that performs authentication (if needed) and builds the initial portion the API request using the authentication information.

Usage

```
use_OpenMpp_custom(req, ...)
```

Arguments

`req`	A function that that performs any required user authentication with the API and builds the initial part of the API request including handling the authentication. Calling `req()` should return a `httr2_request` object.
`...`	Not currently used.

Value

Nothing, invisibly. Behind-the-scenes, an instance of the `OpenMppCustom` R6 class is created. These objects should not be accessed directly by the user, instead, the package internally uses these connections to communicate with the OpenM++ OMS API.

`use_OpenMpp_local` *OpenM++ Local API Connection*

Description

Register a local connection to the OpenM++ Web Services (OMS) Application Programming Interface (API). Currently, two different API connections "local" and "remote" are available. Note that "local" and "remote" describe where the API is running relative to the machine running the R session. Those who use a machine running on their local network may use a remote connection to connect with a cloud-based API, for example. Users running OpenM++ locally or who are logged into a remote virtual machine running OpenM++ will use the local API connection.

Usage

```
use_OpenMpp_local(url = Sys.getenv("OPENMPP_LOCAL_URL"), ...)
```

Arguments

`url`	URL for making API requests. See `Details` for more instructions.
`...`	Not currently used.

Details

We recommend declaring the API URL in your global or project-specific `.Renviron` file. For local and remote API connections, set the following environment variable in your `.Renviron` files:

- `OPENMPP_LOCAL_URL`: URL for a local API connection. Default is to use http://localhost:4040.

Value

Nothing, invisibly. Behind-the-scenes, an instance of the `OpenMppLocal` R6 class is created. These objects should not be accessed directly by the user, instead, the package internally uses these connections to communicate with the OpenM++ API.

Examples

```
## Not run:
use_OpenMpp_local()

## End(Not run)
```

Index