

Package: DynMSR (via r-universe)

October 29, 2026

Title Connect Systems Biology and Pharmacology Formats to Simulation Backends

Version 0.1.2

Author Evgeny Metelkin [aut, cre]

Maintainer Evgeny Metelkin <evgeny.metelkin@gmail.com>

Description Provides a convenient R interface for systems-biology and systems-pharmacology models in Heta, SBML (Systems Biology Markup Language), and DynMS (Dynamic Model Specification) formats. DynMS JSON files are loaded directly; Heta projects and SBML files are converted to DynMS by an optional compatible installation of 'heta-compiler'. Models can be validated, selected from a platform, and generated or built for simulation with 'mrgsolve' as a backend.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1)

Imports jsonlite, jsonvalidate, whisker

Suggests mrgsolve, testthat (>= 3.0.0), withr

Config/testthat/edition 3

URL <https://github.com/hetalang/DynMSR>

BugReports <https://github.com/hetalang/DynMSR/issues>

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libssl-dev libnode-dev

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

Date/Publication 2026-10-29 17:01:00 UTC

RemoteUrl <https://github.com/hetalang/dynmsr>

RemoteRef HEAD

RemoteSha ef3efcfe26c08dff43a6cdd648b06d4a6fc929b4

Contents

build_mrgsolve	2
dynms_lib	3
dynms_load	3
dynms_read	4
dynms_validate_schema	5
dynms_validate_semantic	5
get_model	6
heta_build	7
heta_check	8
heta_load	8
print.model	10
print.platform	10
sbml_load	11
write_mrgsolve	12

Index	14
--------------	-----------

build_mrgsolve	<i>Compile a DynMS model with mrgsolve</i>
----------------	--

Description

Generates an mrgsolve model source file and compiles it with mrgsolve. The mrgsolve package is optional and is required only when this function is called.

Usage

```
build_mrgsolve(model, filepath = NULL, observables = NULL, ...)
```

Arguments

model	A model list, such as one element of platform\$models.
filepath	Optional path for the intermediate mrgsolve model source. If omitted, it is written to a temporary file.
observables	Optional character vector of additional model symbols to return from mrgsolve. Dynamic states and symbols already listed in the DynMS model's observables are skipped. Every requested symbol must exist as a DynMS constant, dynamic state, static state, or assignment. Symbols renamed for mrgsolve are returned under their generated backend names.
...	Additional arguments passed to <code>mrgsolve::mread()</code> .

Value

A compiled mrgsolve model object.

Examples

```
if (requireNamespace("mrgsolve", quietly = TRUE)) {  
  model <- get_model(dynms_load(dynms_lib("repeated-dose")), 1)  
  compiled <- build_mrgsolve(model, observables = "ke", quiet = TRUE)  
  simulation <- mrgsolve::mrgsim_df(compiled, end = 48, delta = 0.1)  
}
```

dynms_lib*Locate bundled DynMS examples*

Description

Returns the available example names or the path to one bundled DynMS JSON file. Use [dynms_load\(\)](#) to load the selected example.

Usage

```
dynms_lib(name = NULL)
```

Arguments

name Name of a bundled example. If NULL, returns all available names.

Value

If name is NULL, a character vector of example names. Otherwise, the path to the selected DynMS JSON file.

Examples

```
dynms_lib()  
platform <- dynms_load(dynms_lib("first-order-decay"))
```

dynms_load*Load a DynMS platform*

Description

Reads a DynMS JSON file, validates it against the bundled schema for its dynms version, runs backend-independent semantic validation, and returns a platform list.

Usage

```
dynms_load(path)
```

Arguments

path Path to a DynMS JSON file.

Details

Schema and semantic validation are both attempted before loading fails. Validation messages are reported with `message()`, followed by one final error when any validation issue is found.

DynMSR currently supports only expressions with format: "math-json".

Value

A platform list. Use `platform$models[[i]]` to select one model for backend-specific operations.

Examples

```
platform <- dynms_load(dynms_lib("two-state-conversion"))
platform
```

dynms_read	<i>Read a DynMS JSON file</i>
------------	-------------------------------

Description

Reads a DynMS JSON file into an R list. DynMS documents are parsed with `jsonlite::fromJSON(..., simplifyVector = FALSE)` to preserve the document structure.

Usage

```
dynms_read(path)
```

Arguments

path Path to a DynMS JSON file.

Value

An R list parsed from JSON.

Examples

```
path <- dynms_lib("first-order-decay")
dynms_read(path)
```

`dynms_validate_schema` *Validate a DynMS document against the JSON Schema*

Description

Validates a DynMS document against a JSON Schema. The document can be passed as a file path or as the raw R list returned by `dynms_read()`. Validation selects a bundled local schema using the document's dynms version. It uses the ajv engine because the bundled schemas use modern JSON Schema features.

Usage

```
dynms_validate_schema(x, error = FALSE)
```

Arguments

<code>x</code>	Path to a DynMS JSON file, or the raw R list returned by <code>dynms_read()</code> .
<code>error</code>	Whether schema validation errors should be raised as one R error after all available issues are collected.

Value

A list with valid, errors, and warnings fields.

Examples

```
validation <- dynms_validate_schema(dynms_lib("first-order-decay"))
validation$valid
```

`dynms_validate_semantic`

Validate DynMS semantic consistency

Description

Validates backend-independent semantic consistency of a schema-valid raw DynMS platform object. Unlike schema validation, this function collects all currently known semantic issues before returning. It does not validate the document structure; use `dynms_validate_schema()` first when the input has not already passed schema validation.

Usage

```
dynms_validate_semantic(raw_platform, error = FALSE)
```

Arguments

- `raw_platform` Raw DynMS platform object, typically returned by `dynms_read()`.
- `error` Whether semantic validation errors should be raised as one R error after all issues are collected.

Value

A list with `valid`, `errors`, and `warnings` fields.

Examples

```
raw_platform <- dynms_read(dynms_lib("first-order-decay"))
validation <- dynms_validate_semantic(raw_platform)
validation$valid
```

get_model	<i>Get one model from a platform</i>
-----------	--------------------------------------

Description

Selects one model from a DynMS platform by numeric index or by name.

Usage

```
get_model(platform, model)
```

Arguments

- `platform` A platform object returned by `dynms_load()`, `heta_load()`, or `sbml_load()`.
- `model` A single model index or model name.

Value

A model list.

Examples

```
dynms_path <- dynms_lib("repeated-dose")
platform <- dynms_load(dynms_path)
model <- get_model(platform, 1)
```

heta_build	<i>Build a Heta project</i>
------------	-----------------------------

Description

Runs `heta build` for a Heta project directory. The arguments map directly to Heta compiler CLI options, for example `units_check = TRUE` adds `--units-check`.

Usage

```
heta_build(
  dir,
  output_dir,
  source = NULL,
  type = NULL,
  debug = FALSE,
  units_check = FALSE,
  log_mode = NULL,
  declaration = NULL,
  log_level = NULL,
  skip_updates = FALSE,
  export = NULL
)
```

Arguments

<code>dir</code>	Project working directory passed as the final CLI argument.
<code>output_dir</code>	Directory for all compiler output. Required. Use <code>output_dir = dir</code> to place output under the project directory.
<code>source</code>	Path to the main source file, passed as <code>--source</code> .
<code>type</code>	Source file type, passed as <code>--type</code> .
<code>debug</code>	If TRUE, add <code>--debug</code> .
<code>units_check</code>	If TRUE, add <code>--units-check</code> .
<code>log_mode</code>	Log file saving mode, passed as <code>--log-mode</code> . Use "always" to write <code>build.log</code> after a successful build.
<code>declaration</code>	Declaration file path without extension, passed as <code>--declaration</code> .
<code>log_level</code>	Log level, passed as <code>--log-level</code> .
<code>skip_updates</code>	If TRUE, add <code>--skip-updates</code> .
<code>export</code>	Export format specification, passed as <code>--export</code> .

Details

Requires a `heta` command that satisfies DynMSR's bundled compatibility rule. Use [heta_check\(\)](#) to inspect availability and the supported version range before building. Compiler output paths are placed under `output_dir`: `dist/`, `meta/`, and, when selected by `log_mode`, `build.log`. The output directory must be supplied explicitly and remains available after the build.

Value

Invisibly returns a list with the command, arguments, exit status, captured output, and output_dir.

Examples

```
## Not run:
project_dir <- "path/to/your/heta-project"
output_dir <- tempfile("dynmsr-heta-output-")
heta_build(project_dir, output_dir, export = "SBML",
            log_mode = "always")

## End(Not run)
```

heta_check	<i>Check Heta compiler availability</i>
------------	---

Description

Checks whether the heta command is available, can report its version with heta --version, and satisfies the bundled DynMSR compatibility rule.

Usage

```
heta_check()
```

Value

A list with available, supported, command, version, supported_version, installation_url, output, and message fields.

Examples

```
heta_check()
```

heta_load	<i>Load a Heta project as a DynMS platform</i>
-----------	--

Description

Builds a Heta project with the DynMS export enabled, reads the generated DynMS JSON file, and returns a DynMS platform object.

Usage

```
heta_load(
  dir,
  output_dir = NULL,
  source = NULL,
  type = NULL,
  debug = FALSE,
  units_check = FALSE,
  log_mode = NULL,
  declaration = NULL,
  log_level = NULL,
  skip_updates = FALSE
)
```

Arguments

<code>dir</code>	Heta project working directory.
<code>output_dir</code>	Optional directory for compiler output. If omitted, a temporary directory is used and removed after loading.
<code>source</code>	Path to the main source file, passed to heta_build() .
<code>type</code>	Source file type, passed to heta_build() .
<code>debug</code>	If TRUE, passed to heta_build() .
<code>units_check</code>	If TRUE, passed to heta_build() .
<code>log_mode</code>	Log file saving mode, passed to heta_build() . Use "always" to write build.log after a successful build.
<code>declaration</code>	Declaration file path without extension, passed to heta_build() .
<code>log_level</code>	Log level, passed to heta_build() .
<code>skip_updates</code>	If TRUE, passed to heta_build() .

Details

Requires a compatible heta-compiler installation. Compiler output is placed under `output_dir` in `dist/`, `meta/`, and, when selected by `log_mode`, `build.log`. When `output_dir` is omitted, a temporary directory is removed after loading.

Value

A platform list. Use [get_model\(\)](#) or ordinary list access such as `platform$models[[1]]` to select one model.

Examples

```
## Not run:
platform <- heta_load("path/to/your/heta-project")
model <- get_model(platform, 1)
compiled <- build_mrgsolve(model, quiet = TRUE)
```

```
simulation <- mrgsolve::mrgsim_df(compiled, end = 24, delta = 0.1)

## End(Not run)
```

print.model	<i>Print a DynMS model</i>
-------------	----------------------------

Description

Displays a compact inventory of model components and their identifiers.

Usage

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

Arguments

x	A DynMS model object.
...	Unused.

Value

x, invisibly.

Examples

```
model <- get_model(dynms_load(dynms_lib("first-order-decay")), 1)
print(model)
```

print.platform	<i>Print a DynMS platform</i>
----------------	-------------------------------

Description

Displays the DynMS version and a compact summary of each model in a platform.

Usage

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

Arguments

x	A DynMS platform object.
...	Unused.

Value

x, invisibly.

Examples

```
platform <- dynms_load(dynms_lib("first-order-decay"))
print(platform)
```

sbml_load

Load an SBML file as a DynMS platform

Description

Builds one SBML file through Heta compiler with the DynMS export enabled, reads the generated DynMS JSON file, and returns a DynMS platform object.

Usage

```
sbml_load(
  path,
  output_dir = NULL,
  debug = FALSE,
  units_check = FALSE,
  log_mode = NULL,
  declaration = NULL,
  log_level = NULL,
  skip_updates = FALSE
)
```

Arguments

path	Path to an SBML file.
output_dir	Optional directory for compiler output, passed to heta_load() .
debug	If TRUE, passed to heta_load() .
units_check	If TRUE, passed to heta_load() .
log_mode	Log file saving mode, passed to heta_load() .
declaration	Declaration file path without extension, passed to heta_load() .
log_level	Log level, passed to heta_load() .
skip_updates	If TRUE, passed to heta_load() .

Details

Requires a compatible heta-compiler installation. The generated DynMS export is written to a temporary directory and removed after loading unless output_dir is supplied.

Value

A platform list. Use `get_model()` or ordinary list access such as `platform$models[[1]]` to select one model.

Examples

```
## Not run:
sbml_path <- "path/to/your/model.xml"
platform <- sbml_load(sbml_path)
model <- get_model(platform, 1)
compiled <- build_mrgsolve(model, quiet = TRUE)
simulation <- mrgsolve::mrgsim_df(compiled, end = 24, delta = 0.1)

## End(Not run)
```

write_mrgsolve

Write an mrgsolve model file

Description

Generates mrgsolve source from one DynMS model.

Usage

```
write_mrgsolve(model, filepath)
```

Arguments

model	A model list, such as one element of <code>platform\$models</code> .
filepath	Path where the generated model source should be written.

Details

The exporter warns and ignores unsupported DynMS features: algebraic dynamic states, `stopSimulation`, and state-event trigger detection. Algebraic states are emitted as ordinary ODE states; state-event conditions are evaluated in generated code without root finding. The current exporter evaluates state-event conditions in \$ODE, which mrgsolve may call at internal trial times. The resulting state update can therefore occur at an output record before or after the true threshold crossing. Smaller delta or hmax values can sometimes make the approximation closer, but cannot reproduce the exact root-crossing time or guarantee which side of the threshold an event will be placed.

Identifiers reserved by mrgsolve are renamed with an `_rnm_` suffix. Further numeric suffixes are added when needed to avoid collisions. The exporter emits one warning and records the complete mapping in the generated model. The DynMS time symbol `t` is emitted as `SOLVERTIME` in \$ODE and as `TIME` in other blocks.

Value

The path to the generated model file.

Examples

```
model <- get_model(dynms_load(dynms_lib("first-order-decay")), 1)
path <- tempfile(fileext = ".mod")
write_mrgsolve(model, path)
unlink(path)
```

Index

build_mrgsolve, [2](#)

dynms_lib, [3](#)
dynms_load, [3](#)
dynms_load(), [3](#), [6](#)
dynms_read, [4](#)
dynms_read(), [5](#), [6](#)
dynms_validate_schema, [5](#)
dynms_validate_schema(), [5](#)
dynms_validate_semantic, [5](#)

get_model, [6](#)
get_model(), [9](#), [12](#)

heta_build, [7](#)
heta_build(), [9](#)
heta_check, [8](#)
heta_check(), [7](#)
heta_load, [8](#)
heta_load(), [6](#), [11](#)

message(), [4](#)
mrgsolve::mread(), [2](#)

print.model, [10](#)
print.platform, [10](#)

sbml_load, [11](#)
sbml_load(), [6](#)

write_mrgsolve, [12](#)