Package: mighty.metadata (via r-universe)

October 29, 2026

Title Manage 'CDISC' 'ADaM' Dataset Specifications in 'YAML' Format

Version 0.2.0

Description Load, validate, and manipulate Clinical Data Interchange Standards Consortium ('CDISC') Analysis Data Model ('ADaM') dataset metadata stored as 'YAML' files. Metadata files are validated against a JSON schema. Provides functions to inspect and modify columns, parameters, and row-level operations within and across 'ADaM' domains. Designed for use with the 'mighty' framework.

License Apache License (>= 2)

URL https://novonordisk-opensource.github.io/mighty.metadata/, https://github.com/NovoNordisk-OpenSource/mighty.metadata

BugReports https://github.com/NovoNordisk-OpenSource/mighty.metadata/issues

Depends R (>= 4.1)

Imports cli, dplyr, glue, purrr, rlang, S7, S7schema (>= 0.2.1), stats, tibble, zephyr

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

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Aksel Thomsen [aut, cre], Ari Siggaard Knoph [aut], Matthew Phelps [aut], Giulia Pais [aut], Katarzyna Graczyk [aut], Michael Hedegaard Thomsen [aut], Novo Nordisk A/S [cph]

Maintainer Aksel Thomsen <oath@novonordisk.com>

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

Date/Publication 2026-10-29 16:18:32 UTC

RemoteUrl https://github.com/cran/mighty.metadata

RemoteRef HEAD

RemoteSha 7e2af1ddf93f9e98cb92dec908e3ef298976c914

Contents

codelists . . . 2
columns . . . 6
create_md . . . 7
create_md_col . . . 8
create_md_param . . . 9
create_md_table . . . 10
create_md_values . . . 11
documents . . . 12
filter_domains . . . 15
mighty.metadata-options . . . 16
mighty_config . . . 16
mighty_domain . . . 17
mighty_study . . . 19
parameters . . . 21
populate_core . . . 22
populate_sparse . . . 23
resolve_includes . . . 24
resolve_subsets . . . 25
rows . . . 26
study_config . . . 27
write_config . . . 29

Index **30**

`codelists` *Mighty Codelists*

Description

`mighty_codelists()` creates an S7 object for storing codelist metadata. The class represents the contents of `_codelists.yml` as a list of codelist entries. Each codelist is referenced from column metadata via its `id` (the `codelist` field of a column).

A codelist can contain the following operation groups:

- `values`: values of a sponsor-defined (non-standard) codelist,
- `subset`: values kept from a standard controlled terminology codelist,
- `restore`: SDTM controlled terminology values included in the ADaM codelist even when they are not used in the SDTM define.xml,
- `extend`: values added to an extensible codelist.

`values` is meant for codelists that are not part of standard controlled terminology, while `subset`, `restore` and `extend` modify a standard codelist retrieved from an external source.

The object is validated on creation and after every modification. Validation includes:

- schema compliance with `inst/schema/codelists.json`,

- uniqueness of codelist identifiers (`id`),
- uniqueness of `code` within a codelist across all operation groups,
- `values` being mutually exclusive with `subset` and `restore`,
- consistent use of `decode` within an operation group (all or none).

Codelists do not have to be referenced by a column. Use `check_codelist_references()` to check that every codelist in a `mighty_study()` is used.

Usage

```
mighty_codelists(file, .data)

list_codelists(x)

select_codelist(x, id)

add_codelist(x, id, label, description, datatype, code, decode = NULL)

remove_codelist(x, id)

update_codelist(x, id, ...)

add_codelist_values(x, id, code, decode = NULL)

extend_codelist_values(x, id, code, decode = NULL)

restore_codelist_values(x, id, code)

subset_codelist_values(x, id, code)

remove_codelist_value(x, id, code)

move_codelist_value(x, id, code, .pos)

update_codelist_value(x, id, code, decode)

check_codelist_references(study)
```

Arguments

`file`	`character(1)` path to `_codelists.yml`. Optional; mutually exclusive with `.data`.
`.data`	`list()` of codelist entries. Optional; mutually exclusive with `file`. Supplying neither returns an empty `mighty_codelists`.
`x`	A `mighty_codelists()` or `mighty_study()` object.
`id`	`character(1)` codelist id. `remove_codelist()` accepts `character()` to remove several codelists.

label	character(1) codelist label.
description	character(1) codelist description.
datatype	character(1) one of "text", "integer", "float".
code	character() or numeric() coded value(s).
decode	character() decode(s), same length as code. Optional (NULL) except in update_codelist_value().
...	Codelist-level fields to update (label, description, datatype).
.pos	integer(1) new position of the value within its group.
study	A mighty_study() object.

Details

Functions to list, select, add, remove, and update codelists and their values in your mighty_codelists() object or directly on a mighty_study() object.

Codelist level:

- add_codelist() adds a new sponsor-defined codelist with values.
- remove_codelist() removes codelist(s) and all their contents.
- update_codelist() updates label, description and datatype.
- select_codelist() returns a single codelist entry as a list.

Value level:

- add_codelist_values() adds values to an existing codelist.
- remove_codelist_value() removes value(s) from whichever operation group they belong to. Empty groups are dropped, and a codelist left without any values is removed.
- move_codelist_value() moves a value within its operation group.
- update_codelist_value() updates the decode of a values or extend entry.

Reuse of external codelists:

Most codelists come from an external source of controlled terminology (e.g. CDISC CT), retrieved by downstream tooling such as mighty.toolbox. Instead of redefining these codelists, the functions below record how the study changes them. Support for reusing external codelists is still maturing. Each function creates the codelist entry if it does not exist.

- subset_codelist_values() keeps only the given codes from the external codelist (subset).
- restore_codelist_values() includes SDTM controlled terminology codes that would otherwise be excluded from the ADaM define.xml (restore).
- extend_codelist_values() adds sponsor-defined codes to an extensible external codelist (extend).

subset and restore cannot be combined with values on the same codelist, as values defines a sponsor-defined codelist.

Study level:

- check_codelist_references() checks that every codelist in a mighty_study() is referenced by at least one column (codelist field), including columns in parameters. It returns study invisibly or errors. The check is not run automatically, so codelists can be added before they are used by any column.

Value

- `mighty_codelists()`: an object of class `mighty_codelists`.
- `list_codelists()`: `character()` vector with codelist ids.
- `select_codelist()`: selected codelist entry as a list.
- All other functions: the modified object.

Examples

```
cl <- mighty_codelists(
  .data = list(
    list(
      id = "REGION",
      label = "Region",
      description = "Geographical regions",
      datatype = "text",
      values = list(list(code = "EUROPE", decode = "Europe"))
    ),
    list(
      id = "AGEU",
      subset = list(list(code = "YEARS")),
      restore = list(list(code = "MONTHS"))
    )
  )
)

print(cl)

cl <- mighty_codelists() |>
  add_codelist(
    id = "REGION",
    label = "Region",
    description = "Geographical regions used in the study",
    datatype = "text",
    code = c("EUROPE", "NORTH AMERICA"),
    decode = c("Europe", "North America")
  ) |>
  subset_codelist_values(id = "AGEU", code = c("MONTHS", "YEARS")) |>
  restore_codelist_values(id = "AGEU", code = "DAYS") |>
  extend_codelist_values(
    id = "LOC",
    code = "MY CUSTOM LOCATION",
    decode = "My Custom Location"
  )

list_codelists(cl)
select_codelist(cl, id = "AGEU")

cl <- cl |>
  update_codelist(id = "REGION", description = "Regions used in the trial") |>
  update_codelist_value(id = "REGION", code = "EUROPE", decode = "EU") |>
  move_codelist_value(id = "AGEU", code = "YEARS", .pos = 1) |>
```

```
  remove_codelist_value(id = "AGEU", code = "MONTHS") |>
  remove_codelist(id = "LOC")


# Check that all codelists in a study are referenced by a column
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
) |>
  extend_codelist_values("UNUSED", code = "A")

try(check_codelist_references(study))
```

`columns` *Update columns in your metadata*

Description

Functions to list, select, remove, add, update, and move columns in your `mighty_domain()` objects.

Usage

```
list_columns(x)

remove_columns(x, id)

add_column(x, id, ..., .pos = length(x[["columns"]]) + 1L)

move_column(x, id, .pos = length(x[["columns"]]))

select_column(x, id)

update_column(x, id, ...)
```

Arguments

`x`	`mighty_domain()` Object to manipulate.
`id`	`character()` Id of the column(s) to remove, add, or move.
`...`	Additional properties to add/update for the column, e.g. a `label = "my label"`.
`.pos`	`integer(1)` Position to put or move the column to. Default places the column as the last.

Value

`invisible(x)`

Examples

```
# Load example configuration
x <- mighty_domain(
  file = system.file("examples", "advs.yml", package = "mighty.metadata")
)

# List all columns defined
list_columns(x)

# Remove the STUDYID and USUBJID columns
x |>
  remove_columns(c("STUDYID", "USUBJID")) |>
  list_columns()

# Add a new column
x |>
  add_column(id = "NEW") |>
  list_columns()

# Add new column with label and as the first column
y <- x |>
  add_column(id = "NEW", label = "My label", .pos = 1)

list_columns(y)

y[["columns"]][[1]] |>
  str()

# Move the STUDYID column to the 3rd position
x |>
  move_column(id = "STUDYID", .pos = 3) |>
  list_columns()

# Update an existing column
x |>
  update_column(id = "STUDYID", label = "Updated Label") |>
  select_column(id = "STUDYID") |>
  str()

# Select a specific column
select_column(x, id = "STUDYID") |>
  str()
```

`create_md` *Create All Metadata Data Sets*

Description

Converts a mighty_study or mighty_domain object into all four metadata data sets at once, as a convenience wrapper around `create_md_table()`, `create_md_col()`, `create_md_param()` and

create_md_values().

Usage

```
create_md(x)
```

Arguments

x	A mighty_study or mighty_domain object.

Value

A named list of tibbles:

mdtable Table definitions, see create_md_table()

mdcol Column definitions, see create_md_col()

mdparam BDS parameter definitions, see create_md_param()

mdvalues Value level definitions, see create_md_values()

See Also

mighty_study, create_md_table(), create_md_col(), create_md_param(), create_md_values()

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)
create_md(study)
```

---

create_md_col *Create Metadata Column Table*

Description

Converts a mighty_study or mighty_domain object into a flat dataframe of column definitions.

Usage

```
create_md_col(x)
```

Arguments

x	A mighty_study or mighty_domain object.

Value

A tibble with one row per column containing:

table_id Table identifier

table_label Table label/description

order Column order within table

id Column name

label Column label

origin Origin type (e.g., "Predecessor", "Derived")

key Logical, whether column is a key

is_core Logical, whether column is a core variable

core String, whether a column is Req, Cond or Perm

method Derivation method

codelist Codelist reference

format_type Data type ("C" or "N")

format_length Maximum length

format_display Display format

comment Comment

See Also

mighty_study, `populate_sparse()`, `populate_core()`, `create_md()`, `create_md_table()`, `create_md_param()`, `create_md_values()`

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)
create_md_col(study)
```

`create_md_param` *Create Metadata Parameter Table*

Description

Converts a mighty_study or mighty_domain object into a flat dataframe of BDS parameter definitions.

Only the parameters themselves are returned. Use `create_md_values()` to get the column definitions nested inside each parameter.

Usage

```
create_md_param(x)
```

Arguments

x	A mighty_study or mighty_domain object.

Value

A tibble with one row per parameter containing:

table_id Table identifier

table_label Table label/description

order Parameter order within table

id Parameter code (PARAMCD)

label Parameter label (PARAM)

See Also

mighty_study, create_md(), create_md_table(), create_md_col(), create_md_values()

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)
create_md_param(study)
```

---

create_md_table *Create Metadata Dataset Table*

Description

Converts a mighty_study or mighty_domain object into a flat dataframe of table (dataset) definitions.

Usage

```
create_md_table(x)
```

Arguments

x	A mighty_study or mighty_domain object.

Value

A tibble with one row per table containing:

order Table order within the study, NA for a single domain

id Table identifier

label Table label/description

class CDISC class of the dataset

subclass CDISC subclass of the dataset

structure Description of the structure of the dataset

keys List column of key variables

comment Comment

See Also

mighty_study, create_md(), create_md_col(), create_md_param(), create_md_values()

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)
create_md_table(study)
```

---

create_md_values *Create Metadata Value Table*

Description

Converts a mighty_study or mighty_domain object into a flat dataframe of value level definitions, i.e. the columns defined inside each BDS parameter.

Parameters without any columns entry contribute no rows. Use create_md_param() for the full list of parameters.

Usage

```
create_md_values(x)
```

Arguments

x	A mighty_study or mighty_domain object.

Value

A tibble with one row per parameter column containing:

table_id Table identifier
table_label Table label/description
param_order Parameter order within table
param_id Parameter code (PARAMCD)
param_label Parameter label (PARAM)
order Column order within parameter
id Column name
label Column label
origin Origin type (e.g., "Predecessor", "Derived")
method Derivation method
codelist Codelist reference
format_type Data type
format_length Maximum length
format_display Display format
comment Comment

See Also

mighty_study, create_md(), create_md_table(), create_md_col(), create_md_param()

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)
create_md_values(study)
```

---

documents *Mighty Documents*

Description

mighty_documents() creates an S7 object for storing document metadata. The class inherits from S7::class_list and represents the contents of _documents.yml as a list of document entries.

The object is validated on creation and when validate() is called. Validation includes:

- schema compliance with inst/schema/documents.json,
- uniqueness of document identifiers (id).

Writing to YAML is done via write_config() on a mighty_study() object, where documents are saved to _documents.yml.

Functions to list, select, remove, add, and update documents in your mighty_documents() object (or through mighty_study@documents).

Usage

```
mighty_documents(file, .data)

list_documents(x)

select_document(x, id)

remove_documents(x, id)

add_document(
  x,
  id,
  title,
  doctype,
  href,
  .pos = length(list_documents(x)) + 1L
)

update_document(x, id, ...)

check_document_references(study, ...)
```

Arguments

file	character(1) path to _documents.yml. Optional; mutually exclusive with .data.
.data	list() of document entries. Optional; mutually exclusive with file. Supplying neither returns an empty mighty_documents.
x	A mighty_documents() or mighty_study() object.
id	character() id of document(s) to select, remove, or update.
title	character(1) document title.
doctype	character(1) document type.
href	character(1) document path/URL.
.pos	integer(1) insertion position for a new document.
...	Additional document properties to update.
study	A mighty_study() object.

Value

An object of class mighty_documents.

- list_documents(): character() vector with document ids.
- select_document(): selected document entry as a list.
- add_document(), update_document(), remove_documents(): modified object (invisible(x)).
- check_document_references(): the input study (invisible(study)). Errors if a domain, column, or parameter references an unknown document id or a METHOD document on a non-Derived column, and warns if a COMMENT reference has missing comment text.

Examples

```
docs <- mighty_documents(
  .data = list(
    list(
      id = "DOC001",
      title = "Statistical Analysis Plan",
      doctype = "suppdoc",
      href = "./docs/sap.pdf"
    )
  )
)

# Custom print method gives a small overview
print(docs)

# Write _documents.yml through mighty_study
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)
study@documents <- docs

# Load study config
s <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)

# Add a document
s <- s |>
  add_document(
    id = "SAP",
    title = "Statistical Analysis Plan",
    doctype = "suppdoc",
    href = "./docs/sap.pdf"
  )

# List and select documents
list_documents(s)
select_document(s, id = "SAP")

# Update existing document
s <- s |>
  update_document(
    id = "SAP",
    title = "Statistical Analysis Plan v2"
  )

# Remove one or more documents
s <- s |>
  remove_documents(id = "SAP")

# Work directly on mighty_documents
docs <- mighty_documents()
```

```
docs <- docs |>
  add_document(
    id = "CSR",
    title = "Clinical Study Report",
    doctype = "suppdoc",
    href = "./docs/csr.pdf"
  )

list_documents(docs)


# Check that all document references in a study are valid
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)
check_document_references(study)
```

`filter_domains` *Filter domains in a study*

Description

Returns a `mighty_study` object containing only domains whose names start with the given prefix(es).

Usage

```
filter_domains(x, prefix = "AD")
```

Arguments

`x`	`mighty_study()` Object to filter.
`prefix`	`character()` one or more prefixes to keep (case-insensitive). Defaults to `"AD"` to keep only ADaM domains.

Value

A `mighty_study` object.

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)

# Keep only ADaM domains (default)
study |>
  filter_domains()
```

mighty.metadata-options

Options for mighty.metadata

Description

verbosity_level:

Verbosity level for functions in mighty.metadata. See zephyr::verbosity_level for details.

- Default: `NA_character_`
- Option: `mighty.metadata.verbosity_level`
- Environment: `R_MIGHTY.METADATA_VERBOSITY_LEVEL`

mighty_config *Mighty Config*

Description

`mighty_config()` provides a robust way of working with the `_mighty.yml` configuration file in the {mighty} framework.

A new object is initialized by supplying either the path to a `_mighty.yml` file or an in-memory `list` of the same content. Both are automatically validated against the `mighty.json` schema.

`mighty_config()` inherits from `S7schema::S7schema()`. You can validate an object at any time by calling `validate()` and use `write_config()` to save it back as a yaml file.

Usage

```
mighty_config(file, .data)
```

Arguments

`file`	`character(1)` path to a `_mighty.yml` file. Mutually exclusive with `.data`.
`.data`	`list` holding a `_mighty.yml` configuration already in memory. Mutually exclusive with `file`.

Details

The `_mighty.yml` file is validated against the `mighty.json` schema on load. The file must contain an `external_data` array declaring the primary keys of any datasets external to the ADaM study (e.g. SDTM or reference datasets) that ADaM domain specifications may depend on.

The required `repos` field specifies where `mighty.component` should look for shared components. Each entry is either a local path (e.g. `"."`) or a GitHub reference in `owner/repo/subdir@ref` format (e.g. `"NovoNordisk-OpenSource/mighty.standards/components@main"`).

Value

A `mighty_config` S7 object extending S7schema::S7schema.

`external_data` A list of external data source specifications, each with an `id` and `keys` field.

`repos` A character vector of component repository locations.

Write Config

Use `write_config()` to serialize a `mighty_config()` object back to a `_mighty.yml` file. Supply `path` to write to a specific file; defaults to the file the object was loaded from.

See Also

mighty_study, study_config, mighty_domain, `write_config()`

Examples

```
x <- mighty_config(
  file = system.file("examples", "_mighty.yml", package = "mighty.metadata")
)

# Custom print method gives a small overview
print(x)

# Underlying object is a `list`
str(x)

# Write back to a file
tmp <- tempfile(fileext = ".yml")
write_config(x, path = tmp)

# Or build one in memory
mighty_config(
  .data = list(
    external_data = list(list(id = "DM", keys = "USUBJID")),
    repos = "."
  )
)
```

`mighty_domain` *Mighty Domain*

Description

`mighty_domain()` provides a robust way of working with ADaM metadata in the {`mighty`} framework.

A new object is initialized by supplying either an existing yaml metadata file or an in-memory `list` of the same content. This package provides helpers to update column, parameter, and row entries. See the references below for help:

- help("columns")
- help("parameters")
- help("rows")

mighty_domain() inherits from S7schema::S7schema() and the yaml file is automatically validated when loaded. The helper functions above also always validates the new configuration before returning.

You can at anytime validate an object by calling validate() and use write_config() to save it as a yaml file again.

Usage

```
mighty_domain(file, .data)
```

Arguments

file	character(1) path to a yaml file defining an ADaM dataset. Mutually exclusive with .data.
.data	list holding an ADaM dataset specification already in memory. Mutually exclusive with file.

Value

A mighty_domain S7 object extending S7schema::S7schema. The underlying list contains the parsed and validated YAML metadata including id, label, class, keys, columns, parameters, and rows.

Examples

```
x <- mighty_domain(
  file = system.file("examples", "advs.yml", package = "mighty.metadata")
)

# Custom print method gives a small overview
print(x)

# Underlying object is a `list`
str(x)

# Or build one in memory
mighty_domain(
  .data = list(
    id = "ADVS",
    label = "Vital Signs Analysis Dataset",
    class = "BASIC DATA STRUCTURE",
    structure = "One record per parameter, per visit, per subject",
    keys = c("USUBJID", "PARAMCD"),
    columns = list(
      list(id = "USUBJID", label = "Unique Subject Identifier")
    )
```

```
  )
)
```

`mighty_study` *Mighty Study*

Description

Creates a `mighty_study` object by loading all YAML metadata files from a directory. Each YAML file (except `_mighty.yml`, `_study.yml`, `_documents.yml` and `_codelists.yml`) is parsed as a mighty_domain object. The optional `_study.yml` file provides study-level properties, the optional `_mighty.yml` file provides mighty framework configuration, the optional `_documents.yml` provides study-level documents metadata, and the optional `_codelists.yml` provides study-level codelists metadata.

Usage

```
mighty_study(path, populate = FALSE)
```

Arguments

`path`	`character(1)` path to a directory containing YAML metadata files.
`populate`	`logical(1)` if `TRUE`, calls `populate_core()` then `populate_sparse()` before returning. Default is `FALSE`.

Details

The function scans the directory for files matching `*.yaml` or `*.yml`:

- Files named `_study.yml` or `_study.yaml` are treated as study properties
- Files named `_mighty.yml` or `_mighty.yaml` are treated as mighty framework config
- File named `_documents.yml` is treated as study documents metadata
- File named `_codelists.yml` is treated as study codelists metadata
- All other YAML files must follow ADaM naming conventions (starting with `ad` or `md`) and are loaded as mighty_domain objects
- Only one `_mighty.yml`, one `_study.yml`, one `_documents.yml` and one `_codelists.yml` file is allowed per directory

Value

A `mighty_study` S7 object extending `list`:

List elements mighty_domain objects, named by their `id` field. Access via e.g. `study$adsl`.

`@study` A study_config object loaded from `_study.yml`, or `NULL` if no properties file exists.

`@mighty` A mighty_config object loaded from `_mighty.yml`, or `NULL` if no configuration file exists.

`@documents` Study-level document metadata from `_documents.yml`, or an empty `mighty_documents` if no documents file exists.

`@codelists` A mighty_codelists object loaded from `_codelists.yml`, or an empty `mighty_codelists` if no codelists file exists.

`@path` The source directory path as `character(1)`.

Write Study Metadata

Use `write_config()` to serialize a `mighty_study()` object back to YAML files. Each domain is written as a separate file, plus `_mighty.yml` and `_study.yml` when `@mighty` and `@study` are not `NULL`, and `_documents.yml` and `_codelists.yml` when `@documents` and `@codelists` are not empty. If `path` is `NULL`, files are written to `x@path`.

See Also

mighty_domain, mighty_config, study_config, `write_config()`, `populate_sparse()`, `populate_core()`, `create_md_table()`, `create_md_col()`, `create_md_param()`, `create_md_values()`

Examples

```
# Load example study
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)

# List tables with metadata
names(study)

# Access ADVS
study$ADVS

# Access study-level properties
study@study

# Access mighty framework configuration
study@mighty

# Access study-level documents metadata
study@documents

# Load and populate in one step
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata"),
  populate = TRUE
)

# Write study back to YAML
tmp <- tempdir()
write_config(study, path = tmp)
```

parameters *Update parameters in your metadata*

Description

Functions to list, select, remove, add, update, and move parameters in your `mighty_domain()` objects.

Usage

```
list_parameters(x)

remove_parameters(x, id)

add_parameter(
  x,
  id,
  label,
  columns,
  ...,
  .pos = length(x[["parameters"]]) + 1L
)

move_parameter(x, id, .pos = length(x[["parameters"]]))

update_parameter(x, id, ...)

select_parameter(x, id)
```

Arguments

`x`	`mighty_domain()` Object to manipulate.
`id`	`character()` Id of the parameter(s) to remove, add, or move.
`label`	`character(1)` Parameter label.
`columns`	`list()` Columns to set for the parameter.
`...`	Additional properties to add for the parameter, e.g. a component reference.
`.pos`	`integer(1)` Position to put or move the parameter to. Default places the parameter as the last.

Value

`invisible(x)`

Examples

```
# Load example configuration
x <- mighty_domain(
  file = system.file("examples", "advs.yml", package = "mighty.metadata")
)

# List all parameters defined
list_parameters(x)

# Remove the BMIGRP parameter
x |>
  remove_parameters("BMIGRP") |>
  list_parameters()

# Add a new parameter
x |>
  add_parameter(
    id = "NEW",
    label = "My new parameter",
    columns = list(list(id = "AVAL"))
  ) |>
  list_parameters()

# Move the BMIGRP parameter to the 1st position
x |>
  move_parameter(id = "BMIGRP", .pos = 1) |>
  list_parameters()

# Update an existing parameter
x |>
  update_parameter(id = "BMI", label = "Updated BMI Label") |>
  select_parameter(id = "BMI") |>
  str()

# Select a specific parameter
select_parameter(x, id = "BMI") |>
  str()
```

`populate_core` *Populate Core Variables*

Description

Adds core variables from supplier datasets as predecessor columns to datasets that use them (marked with `usecore = TRUE`).

Note: Currently only accepts core variables from ADSL.

Usage

```
populate_core(x, ...)
```

Arguments

`x`	A mighty_study or mighty_domain object.
`...`	Additional arguments passed to methods.

Value

A modified mighty_study or mighty_domain with core variables added as predecessor columns.

See Also

mighty_study, `populate_sparse()`, `create_md_col()`

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)
study <- populate_core(study)
```

`populate_sparse` *Populate Predecessor Metadata*

Description

Populates column metadata from predecessor references. Columns with a `method` in the format `domain.column` (e.g., `ADSL.USUBJID`) inherit metadata from the referenced predecessor.

Usage

```
populate_sparse(x, ...)
```

Arguments

`x`	A mighty_study or mighty_domain object.
`...`	Additional arguments passed to methods.

Value

A modified mighty_study or mighty_domain with predecessor column metadata populated.

See Also

mighty_study, `populate_core()`, `create_md_col()`

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)
study <- populate_sparse(study)
```

`resolve_includes` *Resolve conditional metadata items*

Description

Evaluates `include` fields on metadata items (domains, columns, rows, parameters) and removes items where the condition evaluates to `FALSE`.

Usage

```
resolve_includes(x, info = list())
```

Arguments

`x`	A mighty_study or mighty_domain object.
`info`	`list()` Named list of values used to evaluate `include` expressions. For mighty_study, merged into the study_config held in `x@study`.

Details

Include conditions are R expressions with `{glue}` syntax for variable substitution and evaluation. The `info` list provides the values used to do this.

Value

The input object with conditional items resolved.

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)

# Add a conditional column
study$ADVS <- update_column(
  study$ADVS,
  id = "STUDYID",
  include = "{study_id == 'my_study'}"
)

# Condition TRUE: column kept (study_id is "my_study" in @study)
study |>
```

```
  resolve_includes() |>
  getElement("ADVS") |>
  list_columns()

# Condition FALSE: column removed
study |>
  resolve_includes(info = list(study_id = "other")) |>
  getElement("ADVS") |>
  list_columns()
```

`resolve_subsets` *Resolve row subsets*

Description

Rewrites row actions that carry a `subset` field into equivalent `component` actions that operate on a filtered subset of the source domain. The `subset` field is removed after resolution.

Usage

```
resolve_subsets(x)
```

Arguments

`x`	A mighty_study or mighty_domain object.

Details

The row's `component.with.domain` is rewritten to `.mighty_subset(<domain>, "<subset>")`, a marker call that `mighty.component::mighty_component$render()` recognizes and expands into code that applies the action only to rows matching the R expression given in `subset`. A `component` with a `with.domain` entry is required.

Value

The input object with row subsets resolved.

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)

# Add a row action restricted to a subset of source rows
study$ADAE <- add_row(
  study$ADAE,
  id = "TRTEMFL_STUDY1",
  component = list(
```

```
    id = "STUDY1_COMPONENT",
    with = list(domain = "ADAE")
  ),
  subset = "STUDYID == 'STUDY1'"
)

# `subset` is folded into `component.with.domain` as a `.mighty_subset()` marker call
study |>
  resolve_subsets() |>
  getElement("ADAE") |>
  select_row("TRTEMFL_STUDY1") |>
  str()
```

`rows` *Update rows in your metadata*

Description

Functions to list, select, remove, add, update, and move row operations in your `mighty_domain()` objects.

Usage

```
list_rows(x)

remove_rows(x, id)

add_row(x, id, ..., .pos = length(x[["rows"]]) + 1L)

move_row(x, id, .pos = length(x[["rows"]]))

update_row(x, id, ...)

select_row(x, id)
```

Arguments

`x`	`mighty_domain()` Object to manipulate.
`id`	`character()` Id of the row(s) to remove, add, or move.
`...`	Additional properties to add for the row, e.g. a `label = "my row"`.
`.pos`	`integer(1)` Position to put or move the row to. Default places the row as the last.

Value

`invisible(x)`

Examples

```
# Load example configuration
x <- mighty_domain(
  file = system.file("examples", "advs.yml", package = "mighty.metadata")
)

# List all rows defined
list_rows(x)

# Add a new row
y <- x |>
  add_row(id = "NEW")

list_rows(y)

# Remove the new row again
y |>
  remove_rows("NEW") |>
  list_rows()

# Update an existing row
x |>
  update_row(id = "BASELINE", method = "Updated method") |>
  select_row(id = "BASELINE") |>
  str()

# Select a specific row
select_row(x, id = "BASELINE") |>
  str()
```

`study_config` *Study Config*

Description

`study_config()` provides a robust way of working with the `_study.yml` configuration file in the {mighty} framework.

A new object is initialized by supplying either the path to a `_study.yml` file or an in-memory `list` of the same content. Both are automatically validated against the `study.json` schema.

`study_config()` inherits from `S7schema::S7schema()`. You can validate an object at any time by calling `validate()` and use `write_config()` to save it back as a yaml file.

Usage

```
study_config(file, .data)
```

Arguments

file	character(1) path to a _study.yml file. Mutually exclusive with .data.
.data	list holding a _study.yml configuration already in memory. Mutually exclusive with file.

Details

The _study.yml file is validated against the study.json schema on load. The file must contain a study_id field. Additional study-level properties are allowed and are kept as-is.

The optional standards and terminology fields describe the standards and controlled terminologies applied in the study; see vignette("study-schema").

Study-level properties are used by resolve_includes() to evaluate the include conditions of domains, columns, parameters, and rows.

Value

A study_config S7 object extending S7schema::S7schema.

study_id Unique identifier of the study.

study_description Optional description of the study.

standards Optional list of standards, each with id and version.

terminology Optional list of controlled terminologies, each with id and version.

Write Config

Use write_config() to serialize a study_config() object back to a _study.yml file. Supply path to write to a specific file; defaults to the file the object was loaded from.

See Also

mighty_study, mighty_config, mighty_domain, write_config()

Examples

```
x <- study_config(
  file = system.file("examples", "_study.yml", package = "mighty.metadata")
)

# Custom print method gives a small overview
print(x)

# Underlying object is a `list`
str(x)

# Write back to a file
tmp <- tempfile(fileext = ".yml")
write_config(x, path = tmp)

# Or build one in memory
```

```
study_config(
  .data = list(study_id = "example_study", study_description = "A study")
)
```

`write_config` *Write a mighty object to YAML*

Description

Serializes mighty_config, study_config, and mighty_study objects back to YAML files. mighty_config and study_config inherit the `S7schema::write_config()` method of their parent class; mighty_study adds a method that writes every domain plus `_mighty.yml` and `_study.yml`.

Usage

```
write_config(x, path = NULL)

## S7 method for class <mighty.metadata::mighty_study>
write_config(x, path = NULL)
```

Arguments

`x`	A mighty_config, study_config, or mighty_study object.
`path`	Destination to write to. A file for mighty_config and study_config, a directory for mighty_study. If `NULL`, defaults to the source the object was loaded from. Objects built from `.data` have `@file` set to `NULL`, so `path` must be supplied explicitly for those.

Value

Invisibly returns `x`.

See Also

mighty_config, study_config, mighty_study

Index

add_codelist *(codelists)*, 2
add_codelist_values *(codelists)*, 2
add_column *(columns)*, 6
add_document *(documents)*, 12
add_parameter *(parameters)*, 21
add_row *(rows)*, 26

check_codelist_references *(codelists)*, 2
check_document_references *(documents)*, 12
codelists, 2
columns, 6
create_md, 7
create_md(), *9–12*
create_md_col, 8
create_md_col(), *7*, *8*, *10–12*, *20*, *23*
create_md_param, 9
create_md_param(), *7–9*, *11*, *12*, *20*
create_md_table, 10
create_md_table(), *7–10*, *12*, *20*
create_md_values, 11
create_md_values(), *8–11*, *20*

documents, 12

extend_codelist_values *(codelists)*, 2

filter_domains, 15

list_codelists *(codelists)*, 2
list_columns *(columns)*, 6
list_documents *(documents)*, 12
list_parameters *(parameters)*, 21
list_rows *(rows)*, 26

mighty.metadata-options, 16
mighty_codelists, *20*
mighty_codelists *(codelists)*, 2
mighty_config, 16, *19*, *20*, *28*, *29*
mighty_documents *(documents)*, 12
mighty_domain, *7–11*, 17, *17*, *19*, *20*, *23–25*, *28*
mighty_study, *7–12*, *17*, 19, *23–25*, *28*, *29*
mighty_study(), *3*
move_codelist_value *(codelists)*, 2
move_column *(columns)*, 6
move_parameter *(parameters)*, 21
move_row *(rows)*, 26

parameters, 21
populate_core, 22
populate_core(), *9*, *19*, *20*, *23*
populate_sparse, 23
populate_sparse(), *9*, *19*, *20*, *23*

remove_codelist *(codelists)*, 2
remove_codelist_value *(codelists)*, 2
remove_columns *(columns)*, 6
remove_documents *(documents)*, 12
remove_parameters *(parameters)*, 21
remove_rows *(rows)*, 26
resolve_includes, 24
resolve_includes(), *28*
resolve_subsets, 25
restore_codelist_values *(codelists)*, 2
rows, 26

S7schema::S7schema, *17*, *18*, *28*
S7schema::write_config(), *29*
select_codelist *(codelists)*, 2
select_column *(columns)*, 6
select_document *(documents)*, 12
select_parameter *(parameters)*, 21
select_row *(rows)*, 26
study_config, *17*, *19*, *20*, *24*, 27, *29*
subset_codelist_values *(codelists)*, 2

update_codelist *(codelists)*, 2
update_codelist_value *(codelists)*, 2
update_column *(columns)*, 6

update_document (documents), 12
update_parameter (parameters), 21
update_row (rows), 26

write_config, 29
write_config(), *17*, *20*, *28*
write_config,mighty.metadata::mighty_study-method
(write_config), 29

zephyr::verbosity_level, *16*