Package: vmxr (via r-universe)

October 27, 2026

Title VeloMetrix R Client

Version 0.3.0

Author Eric Novik [aut, cre], Juho Timonen [ctb], Generable [cph, fnd]

Maintainer Eric Novik <eric@generable.com>

Description A native R client for the VeloMetrix REST API (vmx-api). Wraps the treatment-to-simulation analysis workflow in ergonomic, pipe-friendly verbs that block-and-poll for asynchronous server jobs and return native R objects (tibbles and typed S3 objects). The package orchestrates server-side jobs; it holds no business logic of its own. Targets API 0.2.x and 0.3 / CLI 0.6.x.

License Apache License (>= 2)

URL https://github.com/generable/vmxr

BugReports https://github.com/generable/vmxr/issues

Depends R (>= 4.1.0)

Imports cli, curl, httr2, jsonlite, rlang, tibble, vctrs

Suggests arrow, dplyr, ggplot2, httptest2, knitr, nlmixr2est, rmarkdown, rxode2, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Config/pak/sysreqs libssl-dev

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

Date/Publication 2026-10-27 04:09:45 UTC

RemoteUrl https://github.com/generable/vmxr

RemoteRef HEAD

RemoteSha ee9981139c5a69ce64717dce5158f6fc88906f55

Contents

vmx_all_specimens . . . 3
vmx_analysis_log . . . 4
vmx_client . . . 5
vmx_data_version . . . 6
vmx_data_version_archive . . . 7
vmx_data_version_create . . . 7
vmx_data_version_export . . . 8
vmx_data_version_table . . . 8
vmx_data_version_unarchive . . . 9
vmx_data_versions . . . 10
vmx_dataset . . . 11
vmx_dataset_cancel . . . 11
vmx_dataset_download . . . 12
vmx_dataset_files . . . 12
vmx_dataset_tags . . . 13
vmx_datasets . . . 13
vmx_dosing . . . 14
vmx_dosing_input . . . 14
vmx_dosing_input_status . . . 15
vmx_fit_global_estimates . . . 15
vmx_fit_obs_vs_pred . . . 16
vmx_fit_subject_estimates . . . 16
vmx_fit_vpc . . . 17
vmx_health . . . 17
vmx_login . . . 18
vmx_marker_selection . . . 19
vmx_model_build . . . 20
vmx_model_build_artifacts . . . 21
vmx_model_build_cancel . . . 21
vmx_model_build_events . . . 22
vmx_model_build_export . . . 22
vmx_model_build_logs . . . 23
vmx_model_build_report . . . 23
vmx_model_build_report_create . . . 24
vmx_model_build_results . . . 24
vmx_model_build_runs . . . 25
vmx_model_build_status . . . 25
vmx_model_catalog . . . 26
vmx_model_data . . . 26
vmx_model_describe . . . 27
vmx_model_fit . . . 27
vmx_model_fit_postprocessor_status . . . 28
vmx_model_fits . . . 28
vmx_modeling_options . . . 29
vmx_nca . . . 30
vmx_nca_analyses . . . 31

vmx_nca_get . . . 32
vmx_nca_result . . . 32
vmx_nlmixr_data . . . 33
vmx_nlmixr_default_cmt . . . 35
vmx_pd . . . 36
vmx_pd_marker . . . 36
vmx_pk . . . 37
vmx_prep_answer . . . 38
vmx_prep_questions . . . 38
vmx_prep_status . . . 39
vmx_sim_cancel . . . 39
vmx_sim_existing_subject . . . 40
vmx_sim_existing_subject_from_text . . . 41
vmx_sim_hypothetical_subject . . . 42
vmx_sim_hypothetical_subject_from_text . . . 43
vmx_sim_jobs . . . 44
vmx_sim_population . . . 44
vmx_sim_population_from_text . . . 45
vmx_sim_result . . . 46
vmx_sim_status . . . 46
vmx_studies . . . 47
vmx_study . . . 48
vmx_study_create . . . 48
vmx_study_update . . . 49
vmx_subjects . . . 49
vmx_torsten_data . . . 50
vmx_treatment . . . 50
vmx_treatment_create . . . 51
vmx_treatment_update . . . 51
vmx_treatments . . . 52
vmx_upload . . . 52
vmx_upload_ignore . . . 53
vmx_upload_unignore . . . 54
vmx_wait . . . 54
vmx_whoami . . . 55

Index **56**

`vmx_all_specimens` *Select all specimens*

Description

A sentinel for an explicit "All specimens" selection. Pass it as `pk_specimen` to `vmx_nca()`, `vmx_modeling_options()` or `vmx_model_build()` to pool PK observations across every specimen, or as the `specimen` of `vmx_pd_marker()` to pool one PD marker across its specimens.

Usage

```
vmx_all_specimens()
```

Details

All specimens is never a default; a call that needs a specimen and gets none fails rather than choosing one.

Value

A `vmx_all_specimens` object.

See Also

`vmx_pd_marker()`

Examples

```
vmx_all_specimens()
```

---

vmx_analysis_log *Audit / analysis log for a study*

Description

`GET /studies/{std_id}/analysis-log` — the complete unified newest-first event feed, with a `kind` discriminator per row.

Usage

```
vmx_analysis_log(
  study,
  kind = NULL,
  event_type = NULL,
  outcome = NULL,
  severity = NULL,
  since = NULL,
  resource = NULL,
  client = vmx_client(),
  event_code = NULL,
  requires_staff_review = NULL
)
```

Arguments

study	A study id (std_...) or vmx_study.
kind	Optional kind filter.
event_type	Optional event-type filter.
outcome	Optional outcome filter.
severity	Optional severity filter.
since	Optional lower time bound: a POSIXct/Date (formatted to ISO-8601 UTC) or an ISO-8601 string.
resource	Optional resource id (or object) to scope to.
client	A vmx_client.
event_code	Optional event-code filter.
requires_staff_review	Optional staff-review filter.

Value

A tibble containing all matching events, newest first.

---

vmx_client *Create a vmxr client*

Description

Resolves connection config with the precedence: explicit args -> VMX_API_BASE_URL / VMX_API_TOKEN environment variables -> ~/.Renviron -> a classed error. The token is stored on the object but redacted in print() and never logged.

Usage

```
vmx_client(base_url = NULL, token = NULL, ...)
```

Arguments

base_url	API base URL. Defaults to Sys.getenv("VMX_API_BASE_URL").
token	Authentik personal access token (PAT). Defaults to Sys.getenv("VMX_API_TOKEN"), then to an OIDC access token (see vmx_login()). Never hard-code a PAT in source.
...	Reserved for future options (timeouts, retries, user agent).

Details

When no token is supplied (neither `token=` nor `VMX_API_TOKEN`), the client **auto-authenticates** via OIDC: it reads the token cached by `vmx_login()` (`~/.config/vmx/oidc-token.json`), refreshing it silently when expired, and – in an interactive session – running `vmx_login()` if there is no usable cached token. This requires `VMX_OIDC_ISSUER` + `VMX_OIDC_CLIENT_ID` to be set; otherwise a `vmx_auth_error` names both auth methods.

The bearer token is resolved **per request**, not frozen at construction: a PAT is constant, but an OIDC access token is short-lived (~5-10 min), so a long-lived `con <- vmx_client()` re-resolves each request and silently refreshes from the cached refresh token when the access token is near expiry. This lets a reused client keep working across a full analysis session without re-login (GEN-2344).

Value

An object of class `vmx_client`.

`vmx_data_version` *Fetch one data version*

Description

Fetch one data version

Usage

```
vmx_data_version(id, client = vmx_client())
```

Arguments

`id`	A data-version id (`dv_...`) or `vmx_data_version`.
`client`	A `vmx_client`.

Value

A `vmx_data_version`.

vmx_data_version_archive

Archive a data version

Description

Archive a data version

Usage

```
vmx_data_version_archive(dv, reason = NULL, client = vmx_client())
```

Arguments

dv	A data-version id or vmx_data_version.
reason	Optional free-text reason.
client	A vmx_client.

Value

The updated vmx_data_version.

vmx_data_version_create

Create a data version

Description

Starts a format job over an explicit upload composition (POST /datasets/{ds_id}/data-versions). Returns the in-flight prep-status; poll it with vmx_wait().

Usage

```
vmx_data_version_create(
  dataset,
  uploads,
  prior_config = NULL,
  client = vmx_client()
)
```

Arguments

dataset	A dataset id (ds_...) or vmx_dataset.
uploads	Character vector of upload ids (upl_...) to format over.
prior_config	Optional prior data-version (dv_...) whose config seeds this job (the config-update lineage pointer).
client	A vmx_client.

Value

A vmx_prep_status for the new job.

---

vmx_data_version_export

Export a data version

Description

Fetches the export envelope (GET /data-versions/{id}/export). Pass dest to obtain the bundle: it is streamed to that path via the server's signed download URL. That URL is short-lived — its life is measured in seconds — and is not returned. Without dest, the parsed envelope is returned with its download_url removed, leaving the export's details (its size, its file list, its data version).

Usage

```
vmx_data_version_export(dv, dest = NULL, client = vmx_client())
```

Arguments

dv	A data-version id or vmx_data_version.
dest	Optional local file path to stream the bundle to. Pass it to obtain the bundle; without it no download link is returned.
client	A vmx_client.

Value

Without dest, the export envelope (list) minus its short-lived download_url. With dest, dest invisibly once the bundle is streamed.

---

vmx_data_version_table

A prepared data-version table

Description

GET /data-versions/{id}/tables/{domain} — returns the formatter's prepared domain table as a tibble (columns typed per the server's column metadata, which is attached as the "columns" attribute). gen_subject_uuid is the canonical subject join key.

Usage

```
vmx_data_version_table(
  dv,
  domain = c("subjects", "pk", "dosing", "pd", "labs", "covariates"),
  time_basis = NULL,
  client = vmx_client()
)
```

Arguments

dv	A data-version id (dv_...) or vmx_data_version.
domain	One of "subjects", "pk", "dosing", "pd", "labs", "covariates".
time_basis	Optional time basis name (e.g. "observed"). A basis the DataVersion does not list as available is refused client-side. Passing it with a bare id skips the DataVersion fetch, so the availability check and the requirement that a v0.3 server echo the basis then rely on the server.
client	A vmx_client.

Details

API 0.3 projects every analytical table on a time basis and requires the time_basis query parameter. When time_basis is NULL, the DataVersion's recommended basis is used (fetching the DataVersion first if only an id was given). A v0.3 DataVersion with no recommended basis (no basis has PK-eligible subjects) needs an explicit time_basis; only a DataVersion without a time_bases map (an API 0.2 server) is fetched without the parameter.

Value

A tibble. The selected basis is attached as the "time_basis" attribute when one was requested or echoed.

---

vmx_data_version_unarchive

Unarchive a data version

Description

Unarchive a data version

Usage

```
vmx_data_version_unarchive(dv, client = vmx_client())
```

Arguments

dv	A data-version id or vmx_data_version.
client	A vmx_client.

Value

The updated vmx_data_version.

---

vmx_data_versions *List data versions*

Description

The modeling-eligibility filter maps onto whichever query param the served API contract accepts. The 0.2.2 API filters on a single basis-agnostic eligible_for_modeling boolean; the 0.3 API replaced it with the basis-scoped pk_eligible_for_modeling_after_qc filter, which requires a companion time_basis (api-contract §5.4). Because a 0.3 server *silently drops* the unknown 0.2.2 param and returns the full unfiltered list, the served shape is detected first — from /health's api_contract_version (§5.13) — and the filter is mapped accordingly; a 0.3 server asked to filter without a time_basis errors rather than returning an unfiltered list.

Usage

```
vmx_data_versions(
  treatment = NULL,
  study = NULL,
  include_archived = FALSE,
  eligible_for_modeling = NULL,
  client = vmx_client(),
  time_basis = NULL
)
```

Arguments

treatment	Optional treatment filter.
study	Optional study filter.
include_archived	Include archived versions.
eligible_for_modeling	Optional modeling-eligibility filter (TRUE / FALSE). Against a 0.3 server it selects DataVersions eligible for modeling after QC *on* time_basis — a basis-scoped question; against 0.2.2 it is the basis-agnostic flag. The two servers therefore answer subtly different questions for the same argument (0.3 narrows to one basis), which is inherent to the contract change.
time_basis	Optional time basis: one of "observed", "nominal", "nominal_from_observed_dose", or "observed_from_predose_pk". Required alongside eligible_for_modeling when the server serves the 0.3 contract; ignored by (and not sent to) a 0.2.2 server.
client	A vmx_client.

Value

A tibble containing all matching data versions.

`vmx_dataset` *Fetch one dataset*

Description

Fetch one dataset

Usage

```
vmx_dataset(id, client = vmx_client())
```

Arguments

`id`	A dataset id (`ds_...`) or `vmx_dataset`.
`client`	A `vmx_client`.

Value

A `vmx_dataset`.

`vmx_dataset_cancel` *Cancel a dataset's in-flight format job*

Description

`POST /datasets/{ds_id}/cancel` — terminates the format job and mints a cancelled DataVersion.

Usage

```
vmx_dataset_cancel(dataset, client = vmx_client())
```

Arguments

`dataset`	A dataset id or `vmx_dataset`.
`client`	A `vmx_client`.

Value

The updated `vmx_prep_status`.

vmx_dataset_download *Download a dataset's files*

Description

Not implemented: the API's dataset-files listing does not expose per-file download URLs. Use `vmx_data_version_export()` to pull a curated bundle.

Usage

```
vmx_dataset_download(dataset, dest = ".", client = vmx_client())
```

Arguments

dataset	A dataset id or vmx_dataset.
dest	Destination directory.
client	A vmx_client.

Value

Not yet implemented.

vmx_dataset_files *List the files in a dataset*

Description

List the files in a dataset

Usage

```
vmx_dataset_files(dataset, client = vmx_client())
```

Arguments

dataset	A dataset id or vmx_dataset.
client	A vmx_client.

Value

A tibble containing all files in the dataset.

vmx_dataset_tags *The tags on a dataset*

Description

Returns the dataset's allowlisted tag map as a two-column (`key`, `value`) tibble. Reads the tags off a `vmx_dataset` object when given one, else fetches the dataset.

Usage

```
vmx_dataset_tags(dataset, client = vmx_client())
```

Arguments

`dataset`	A dataset id or `vmx_dataset`.
`client`	A `vmx_client`.

Value

A tibble with `key` and `value` columns.

vmx_datasets *List datasets*

Description

List datasets

Usage

```
vmx_datasets(study = NULL, treatment = NULL, client = vmx_client())
```

Arguments

`study`	Optional study filter.
`treatment`	Optional treatment filter.
`client`	A `vmx_client`.

Value

A tibble containing all matching datasets.

vmx_dosing *Dosing events table*

Description

Dosing events table

Usage

```
vmx_dosing(dv, time_basis = NULL, client = vmx_client())
```

Arguments

dv	A data-version id or vmx_data_version.
time_basis	Optional time basis; see vmx_data_version_table().
client	A vmx_client.

Value

A tibble.

vmx_dosing_input *Create a dosing input for a fit*

Description

POST /model-fits/{mf_id}/simulation-dosing-inputs.

Usage

```
vmx_dosing_input(
  fit,
  dosing_text,
  scenario_name,
  client = vmx_client(),
  wait = FALSE,
  ...
)
```

Arguments

fit	A fit id (mf_...) or vmx_model_fit.
dosing_text	The dosing regimen text.
scenario_name	One or more scenario names.
client	A vmx_client.
wait	If TRUE, block until parsing succeeds or fails.
...	Polling controls forwarded to vmx_wait().

Value

A vmx_dosing_input (carries dosing_input_id).

---

vmx_dosing_input_status

Dosing-input status

Description

Dosing-input status

Usage

```
vmx_dosing_input_status(dosing_input, client = vmx_client())
```

Arguments

dosing_input	A dosing-input id or vmx_dosing_input.
client	A vmx_client.

Value

A vmx_dosing_input.

---

vmx_fit_global_estimates

Global (population) parameter estimates (tidy)

Description

One row per estimate, preserving the server-selected point statistic, interval kind, interval level, and tagged estimate metadata.

Usage

```
vmx_fit_global_estimates(fit, client = vmx_client())
```

Arguments

fit	A fit id or vmx_model_fit.
client	A vmx_client.

Value

A tibble.

vmx_fit_obs_vs_pred *Observed-vs-predicted diagnostic (tidy)*

Description

Reshapes the PK block into one row per observation. The returned prediction columns retain the server-selected point statistic and interval semantics. A named list of equivalently reshaped PD-marker tibbles is attached as the `"pd_markers"` attribute; units and marker references remain attributes on their respective tibbles.

Usage

```
vmx_fit_obs_vs_pred(fit, client = vmx_client())
```

Arguments

`fit`	A fit id or `vmx_model_fit`.
`client`	A `vmx_client`.

Value

A tibble (one row per PK observation).

vmx_fit_subject_estimates *Subject-level parameter estimates (tidy, long)*

Description

One row per subject x estimate. `value_statistic`, `interval_kind`, and `interval_level` preserve the server-selected estimate semantics; `interval_lower` and `interval_upper` are the corresponding bounds. Tagged estimate metadata is retained as columns rather than interpreted by vmxr.

Usage

```
vmx_fit_subject_estimates(fit, client = vmx_client())
```

Arguments

`fit`	A fit id or `vmx_model_fit`.
`client`	A `vmx_client`.

Value

A tibble.

vmx_fit_vpc *Visual predictive check artifact*

Description

Returns the parsed Visual Predictive Check artifact. Its subject and dose-group channels contain model-implied response trajectories with the server-provided point statistic and interval. The nested wire shape is retained verbatim.

Usage

```
vmx_fit_vpc(fit, client = vmx_client())
```

Arguments

fit	A fit id or vmx_model_fit.
client	A vmx_client.

Value

A list (the parsed artifact).

vmx_health *Connectivity probe*

Description

Calls GET /health.

Usage

```
vmx_health(client = vmx_client())
```

Arguments

client	A vmx_client.

Value

Invisibly, a vmx_health object (status, version, api_contract_version, engine, engine_version); raises a classed error if the API is unreachable or unhealthy.

vmx_login *Authenticate to VeloMetrix via OIDC device-code*

Description

Runs the OIDC device-code flow (RFC 8628, via `httr2::oauth_flow_device()`) against the workspace's Authentik provider and caches the resulting token so later `vmx_client()` calls authenticate automatically. Endpoints are resolved from the issuer's `.well-known/openid-configuration` discovery document; the flow uses a **public client** (no secret, no PKCE) and requests `offline_access` so Authentik issues a **refresh token**.

Usage

```
vmx_login(issuer = NULL, client_id = NULL, scopes = NULL, cache_path = NULL)
```

Arguments

`issuer`	OIDC issuer base URL. Defaults to `VMX_OIDC_ISSUER`.
`client_id`	Public OIDC client id. Defaults to `VMX_OIDC_CLIENT_ID`.
`scopes`	Space-separated scopes. Defaults to `VMX_OIDC_SCOPES`, then to `"openid profile email offline_access goauthentik.io/api"`.
`cache_path`	Where to cache the token. Defaults to the CLI-shared `~/.config/vmx/oidc-token.json`. The `VMX_OIDC_TOKEN_CACHE` env var can override this, but it is a **testing-only** override (the `vmx` CLI has no such variable): pointing it elsewhere makes vmxr read/write a cache the CLI never sees, breaking the "one login serves both" invariant. Leave it unset in normal use.

Details

The token is written as plain JSON to `~/.config/vmx/oidc-token.json` (the same path and shape the `vmx` CLI uses, with `0600` permissions), so one `vmx_login()` serves both R and the terminal CLI and the session survives a fresh R process or workspace pod restart. Because the refresh token is persisted on the home PVC, you log in once per provider-configured refresh-token lifetime.

Configuration is read from environment variables (matching the CLI): `VMX_OIDC_ISSUER`, `VMX_OIDC_CLIENT_ID`, and optionally `VMX_OIDC_SCOPES`. Workspace deployments provision the issuer and client id; vmxr does not assume values from a different workspace or environment.

Value

Invisibly, the cached token (a list; the access/refresh tokens are secret and never printed).

vmx_marker_selection *Accepted specimen selections of an NCA, model-build run or model fit*

Description

Reads the selections the API accepted for a resource (`GET .../marker-selection`). These are what the analysis actually used and are never recomputed from the data version's current metadata.

Usage

```
vmx_nca_marker_selection(nca, client = vmx_client())

vmx_model_build_marker_selection(run, client = vmx_client())

vmx_model_fit_marker_selection(fit, client = vmx_client())
```

Arguments

nca	An NCA id (`nca_...`) or `vmx_nca_analysis`.
client	A `vmx_client`.
run	A build-run id (`run_...`) or `vmx_model_build_run`.
fit	A fit id (`mf_...`) or `vmx_model_fit`.

Details

- `vmx_nca_marker_selection()` returns `list(pk_specimen)`.
- `vmx_model_build_marker_selection()` returns `list(pk_specimen, pd_markers)`, where each PD entry carries `gen_uuid`, either `specimen_type` or `all_specimens = TRUE`, and `direction`. A run accepted before PD specimens were selectable may report only `gen_uuid` and `direction`.
- `vmx_model_fit_marker_selection()` returns `list(pk_specimen, pd_marker)`; `pd_marker` is `NULL` for a PK fit, and for a fit saved before specimen selection it may carry only `gen_uuid`.

`pk_specimen` is `list(specimen_type = "...")` or `list(all_specimens = TRUE)`. An analysis accepted before PK specimens were selectable reports all specimens.

Value

A named list, as described above.

`vmx_model_build` *Start a model build run (optionally wait)*

Description

`pk_specimen` is required: one PK specimen type (for example `"plasma"`) or `vmx_all_specimens()`. vmxr never fills it in, not even from the data version's recommendation; a call without it fails and lists the specimens available. Each PD marker is a `vmx_pd_marker()` naming one specimen or all specimens, with its `direction`. Two specimens of one marker may be selected as separate entries. Read the accepted selections back with `vmx_model_build_marker_selection()`.

Usage

```
vmx_model_build(
  data_version,
  time_basis,
  pk_specimen,
  pd_marker = NULL,
  covariate = NULL,
  idempotency_key = NULL,
  retried_from = NULL,
  wait = FALSE,
  ...,
  client = vmx_client()
)
```

Arguments

`data_version`	A data-version id or `vmx_data_version`.
`time_basis`	Time basis.
`pk_specimen`	One PK specimen type string, or `vmx_all_specimens()`. Required.
`pd_marker`	`NULL` (the default) for a PK-only build, or one `vmx_pd_marker()` or a list of them, each with a `direction`.
`covariate`	Optional covariate name(s).
`idempotency_key`	Optional idempotency key.
`retried_from`	Optional prior run to retry from.
`wait`	If `TRUE`, block until the run settles.
`...`	Polling controls forwarded to `vmx_wait()` when `wait = TRUE` (e.g. `timeout`, `interval`, `progress`).
`client`	A `vmx_client`.

Value

A `vmx_model_build_run`.

vmx_model_build_artifacts *Download build-run artifacts*

Description

Download build-run artifacts

Usage

```
vmx_model_build_artifacts(run, dest = ".", client = vmx_client())
```

Arguments

run	A build-run id or object.
dest	Destination directory.
client	A vmx_client.

Value

Not yet implemented.

vmx_model_build_cancel *Cancel a build run*

Description

Cancel a build run

Usage

```
vmx_model_build_cancel(run, client = vmx_client())
```

Arguments

run	A build-run id or object.
client	A vmx_client.

Value

A vmx_model_build_run.

vmx_model_build_events

Build-run events (SSE stream)

Description

Build-run events (SSE stream)

Usage

```
vmx_model_build_events(run, client = vmx_client())
```

Arguments

run	A build-run id or object.
client	A vmx_client.

Value

Not yet implemented.

vmx_model_build_export

Markdown export of a build run

Description

Markdown export of a build run

Usage

```
vmx_model_build_export(run, client = vmx_client())
```

Arguments

run	A build-run id or object.
client	A vmx_client.

Value

The export markdown as a length-1 character vector.

vmx_model_build_logs *Build-run logs*

Description

Build-run logs

Usage

```
vmx_model_build_logs(run, client = vmx_client(), order = c("desc", "asc"))
```

Arguments

run	A build-run id or object.
client	A vmx_client.
order	Newest-first ("desc") or oldest-first ("asc").

Value

A tibble containing all log lines in the requested order.

vmx_model_build_report *Build-run report status (signed HTML report URL when ready)*

Description

Build-run report status (signed HTML report URL when ready)

Usage

```
vmx_model_build_report(run, client = vmx_client())
```

Arguments

run	A build-run id or object.
client	A vmx_client.

Value

A list with status and, when ready, url.

vmx_model_build_report_create

Request build-run report generation

Description

`POST /model-build-runs/{run_id}/report` queues HTML report generation.

Usage

```
vmx_model_build_report_create(
  run,
  subject_plot_mode = c("all", "none"),
  client = vmx_client()
)
```

Arguments

`run`	A build-run id or object.
`subject_plot_mode`	One of `"all"` or `"none"`.
`client`	A `vmx_client`.

Value

A list with report status.

vmx_model_build_results

Build-run results

Description

Build-run results

Usage

```
vmx_model_build_results(run, client = vmx_client())
```

Arguments

`run`	A build-run id or object.
`client`	A `vmx_client`.

Value

A list (fits summary, modeling population, PK structure selection).

vmx_model_build_runs *List model build runs*

Description

Run and fit reads request the API's marker-scopes representation, so runs that select a PD marker across all specimens are listed and readable, and PD selections report `specimen_type` or `all_specimens`.

Usage

```
vmx_model_build_runs(
  data_version = NULL,
  study = NULL,
  treatment = NULL,
  status = NULL,
  client = vmx_client()
)
```

Arguments

`data_version`, `study`, `treatment`, `status`	Optional filters.
`client`	A `vmx_client`.

Value

A tibble containing all matching model-build runs.

vmx_model_build_status *Build-run status*

Description

Build-run status

Usage

```
vmx_model_build_status(run, client = vmx_client())
```

Arguments

`run`	A build-run id (`run_...`) or object.
`client`	A `vmx_client`.

Value

A `vmx_model_build_run`.

vmx_model_catalog *Model catalog*

Description

GET /model-catalog returns categories of models; this flattens them into one tibble with a category column.

Usage

```
vmx_model_catalog(data_version = NULL, client = vmx_client())
```

Arguments

data_version	Optional data-version to tailor the catalog to.
client	A vmx_client.

Value

A tibble.

vmx_model_data *Fetch model-ready tidy tables for a data version*

Description

Returns a vmx_model_data bundle with $subjects, $pk, $dosing, $pd, $labs and $covariates (each a tibble, or NULL when the DataVersion has no such prepared table), and $meta (units, time bases, the selected time_basis, PD-marker manifest, subject count) read from the DataVersion. Only domains flagged in the DV's table_availability are fetched, so absent optional tables don't 404. Every table is projected on the same time basis.

Usage

```
vmx_model_data(dv, time_basis = NULL, client = vmx_client())
```

Arguments

dv	A data-version id or vmx_data_version.
time_basis	Optional time basis; defaults to the DataVersion's recommended basis. See vmx_data_version_table().
client	A vmx_client.

Value

A vmx_model_data object.

vmx_model_describe *Describe a model*

Description

Describe a model

Usage

```
vmx_model_describe(model_name, client = vmx_client())
```

Arguments

model_name	Catalog model name.
client	A vmx_client.

Value

A named list.

vmx_model_fit *Fetch one model fit's details*

Description

Fetch one model fit's details

Usage

```
vmx_model_fit(id, client = vmx_client())
```

Arguments

id	A fit id (mf_...) or vmx_model_fit.
client	A vmx_client.

Value

A vmx_model_fit (metadata / model / inference).

vmx_model_fit_postprocessor_status *Model-fit postprocessor status*

Description

Model-fit postprocessor status

Usage

```
vmx_model_fit_postprocessor_status(fit, client = vmx_client())
```

Arguments

fit	A fit id or vmx_model_fit.
client	A vmx_client.

Value

A list with postprocessor status.

vmx_model_fits *List model fits*

Description

Requests the marker-scopes representation, so fits of a PD marker selected across all specimens are listed and each PD marker reports its specimen scope.

Usage

```
vmx_model_fits(
  run = NULL,
  data_version = NULL,
  model_type = NULL,
  marker_name = NULL,
  status = NULL,
  client = vmx_client(),
  treatment = NULL,
  study = NULL,
  source_pk_model_fit = NULL
)
```

Arguments

run	Optional model-build run filter.
data_version	Optional data-version filter.
model_type	Optional model-type filter.
marker_name	Optional marker-name filter.
status	Optional model-fit status filter.
client	A vmx_client.
treatment	Optional treatment filter.
study	Optional study filter.
source_pk_model_fit	Optional source PK model-fit filter.

Value

A tibble containing all matching model fits.

---

vmx_modeling_options *Preview modeling options for a data version*

Description

POST /modeling-options previews the modeling population and available covariates for one PK specimen scope, a set of PD marker selections and a time basis. Nothing is persisted.

Usage

```
vmx_modeling_options(
  data_version,
  time_basis,
  pk_specimen,
  pd_marker = NULL,
  covariate = NULL,
  client = vmx_client()
)
```

Arguments

data_version	A data-version id or vmx_data_version.
time_basis	Time basis.
pk_specimen	One PK specimen type string, or vmx_all_specimens(). Required.
pd_marker	NULL (the default) to preview a PK-only build, or one vmx_pd_marker() or a list of them.
covariate	Optional covariate name(s).
client	A vmx_client.

Details

pk_specimen is required: one PK specimen type (for example "plasma") or vmx_all_specimens(). vmxr never fills it in, not even from the data version's recommendation; a call without it fails and lists the specimens available. Each PD marker is a vmx_pd_marker() naming one specimen or all specimens; any direction on it is ignored here, so the same selections can be passed on to vmx_model_build().

Value

A list (the selection preview).

---

vmx_nca *Run an NCA analysis*

Description

Creates the analysis (POST /nca-analyses) and, by default, blocks until it settles. time_basis is one of "observed", "nominal", "nominal_from_observed_dose", or "observed_from_predose_pk" (validated server-side against the DataVersion's available bases).

Usage

```
vmx_nca(
  data_version,
  time_basis,
  pk_specimen,
  idempotency_key = NULL,
  retried_from = NULL,
  wait = TRUE,
  ...,
  client = vmx_client()
)
```

Arguments

data_version	A data-version id (dv_...) or vmx_data_version.
time_basis	The time basis to compute on.
pk_specimen	One PK specimen type string, or vmx_all_specimens(). Required.
idempotency_key, retried_from	Optional create fields.
wait	If TRUE (default), block until the analysis is terminal.
...	Polling controls forwarded to vmx_wait() when wait = TRUE (e.g. timeout, interval, progress).
client	A vmx_client.

Details

pk_specimen is required: name one PK specimen (for example "plasma") or pass vmx_all_specimens() to pool every specimen. vmxr never chooses one for you, not even the data version's recommended specimen; a call without it fails and lists the specimens the data version offers. A named specimen needs the data version's per-specimen marker analysis. Read the accepted selection back with vmx_nca_marker_selection().

Value

A vmx_nca_analysis.

---

vmx_nca_analyses *List NCA analyses*

Description

List NCA analyses

Usage

```
vmx_nca_analyses(
  data_version = NULL,
  study = NULL,
  treatment = NULL,
  status = NULL,
  time_basis = NULL,
  client = vmx_client()
)
```

Arguments

data_version	Optional data-version (dv_...) filter.
study	Optional study (std_...) filter.
treatment	Optional treatment (tmt_...) filter.
status	Optional status filter (queued/running/completed/ degraded/failed).
time_basis	Optional time-basis filter.
client	A vmx_client.

Value

A tibble containing all matching analyses.

vmx_nca_get *Fetch one NCA analysis*

Description

Fetch one NCA analysis

Usage

```
vmx_nca_get(id, client = vmx_client())
```

Arguments

id	An NCA id (nca_...) or vmx_nca_analysis.
client	A vmx_client.

Value

A vmx_nca_analysis.

vmx_nca_result *NCA result table (PK parameters, per subject and result interval)*

Description

Reads the NCA result and returns a tidy tibble of point estimates, handling both server shapes transparently:

Usage

```
vmx_nca_result(nca, client = vmx_client())
```

Arguments

nca	An NCA id or vmx_nca_analysis.
client	A vmx_client.

Details

- **API 0.3** returns a cursor-paged collection whose `items[]` are `NcaResultItems` — one per result interval of interest. Every page is followed to the end via `next_cursor` and the items are assembled without truncation. Each row is one subject within one interval, and the per-interval dimension is carried in the `item_index`, `label`, `interval_start_hours`, and `interval_end_hours` columns (the two `*_hours` columns are `NA` when the server did not resolve a time interval for that subject).
- **API 0.2.2** returns a single flat object with top-level per-subject parallel arrays and one `point_estimates` map. Each row is one subject (there is no per-interval dimension, so the interval columns are absent).

In both shapes the `point_estimates` map (metric -> per-subject values, parallel to `gen_subject_uuid`) becomes one column per PK quantity, and `not_estimable_reasons` — the per-subject map of metric -> reason strings — is surfaced as the `not_estimable_reasons` list-column, so a null point estimate carries its reason. Quantity metadata (display names, units, explanations) is attached as the `"quantities"` attribute and the compact unit lookup as `"units"`; for 0.3 these are the de-duplicated union across items.

Value

A tibble.

`vmx_nlmixr_data` *NONMEM-layout event dataset for nlmixr2 / rxode2*

Description

Assembles one event table from a DataVersion's prepared `pk`, `dosing`, `subjects`, `covariates` and (optionally) `pd` tables, in the layout nlmixr2 and rxode2 consume directly: one row per dose event and per observation, ordered by subject and time.

Usage

```
vmx_nlmixr_data(
  dv,
  analyte = NULL,
  time_basis = NULL,
  eligibility = c("after_qc", "before_qc", "all"),
  pd_markers = NULL,
  cmt = NULL,
  replicate = FALSE,
  client = vmx_client(),
  pk_specimen = NULL
)
```

Arguments

dv	A data-version id (`dv_...`) or `vmx_data_version`.
analyte	PK analyte to assemble, matched against `biomarker_name` or `biomarker_code`. Required when the PK table carries more than one analyte. Specimens of one analyte are chosen with `pk_specimen`.
time_basis	Time basis to project on (e.g. `"observed"`). Defaults to the DataVersion's recommended basis; a basis the DataVersion does not list as available is refused, and a DataVersion with no recommended basis needs an explicit one.
eligibility	Which modeling-eligibility flag admits a row: `"after_qc"` (default, the contract's admission filter), `"before_qc"`, or `"all"` (no filter). The latter two are for review only.
pd_markers	PD markers to include as additional observation endpoints: `NULL`/`FALSE` (default, PK only); `TRUE` (every marker-and-specimen pair the DataVersion marks eligible on the selected basis, each its own endpoint; specimens are never pooled); a character vector of marker names, each recorded in a single specimen; or a named list mapping each marker name to one or more specimen names (one endpoint per specimen) or to `vmx_all_specimens()` (one pooled endpoint), e.g. `list(effect = c("plasma", "serum"), resp = vmx_all_specimens())`; or `vmx_pd_marker()` selections (one, or an unnamed list), as the modeling requests take them, whose `direction` is ignored here. An explicitly requested scope that is ineligible is a usage error.
cmt	Named integer vector mapping dosing routes and `"observation"` to compartments; see `vmx_nlmixr_default_cmt()`. PD markers get compartments after the highest mapped value, in the order they are included.
replicate	If `TRUE`, add a `REPLICATE` column: the 1-based rank of each observation within its subject, endpoint and time (`(ID, DVID, TIME)`), ordered by `gen_measurement_uuid` then source-row order; `NA` on dose rows and `1` for an observation with no same-time partner. Off by default. The index is derived in R, not read from the DataVersion.
client	A `vmx_client`.
pk_specimen	PK specimen to assemble: one recorded specimen name (e.g. `"plasma"`) or `vmx_all_specimens()` to pool every specimen. Required when the PK data records more than one specimen; with one or none it may be omitted.

Details

Rows are ordered by subject and time; ties on `TIME` are ordered dose before observation, which is how rxode2 and nlmixr2 resolve them, so a pre-dose sample that shares its time with an IV bolus into the observation compartment must carry its own earlier time in the source data.

Observations that share a subject, endpoint and time are always emitted in a deterministic order — by the measurement uuid (`gen_measurement_uuid`, the row identity on the pk/pd tables) and then source-row order — whether or not the replicate index is requested. With `replicate = TRUE` the layout also carries a `REPLICATE` column: the 1-based rank of each observation within its `(ID, DVID, TIME)` group in that order, `NA` on dose rows and `1` for an observation with no same-time partner. The index is derived here in R and is deliberately not part of the DataVersion. rxode2

and nlmixr2 treat any extra column as a covariate, and REPLICATE is NA on dose rows, so drop it (dplyr::select(-REPLICATE)) before passing the table to a fit unless the model uses it explicitly.

Columns: ID (dense integer per subject with admitted rows), TIME (hours on the selected time basis), DV, AMT (mg), EVID (0 observation / 1 dose), MDV, CMT, RATE (mg/h, > 0 for infusions), II, ADDL, SS (always 0 — the DataVersion already expands doses per administration), CENS / LIMIT (censoring: CENS = 1 with DV = lloq and LIMIT = 0 for BLQ rows, CENS = -1 with DV = uloq and LIMIT = Inf for ALQ rows), DVID (only when PD markers are included: 1 = PK, 2.. = one per PD endpoint, and dose rows carry 1), subject_id, specimen_type (only when the DataVersion records specimens: each observation's own recorded specimen, also inside a pooled endpoint, and NA on dose rows), and one column per subject-level covariate.

A marker measured in more than one specimen (plasma and serum, say) is one marker with one input per specimen. The specimen is always the caller's choice: with more than one PK specimen, pk_specimen is required, and a PD marker with more than one specimen needs its specimens named in pd_markers. The DataVersion's recommended PK specimen is never used as an implicit choice. Each selection is admitted by the DataVersion's recorded eligibility verdict for that scope on the selected basis: the specimen's own verdict (by marker and specimen), or the All-specimens verdict when pooled. A scope verdict that is not on the DataVersion itself is read from its marker-analysis metadata. Pooling specimens that do not share one explicitly known unit is an error that names the units found. A DataVersion that predates specimens (no specimen_type column) is assembled exactly as before.

Only rows the DataVersion marks as eligible for modeling are kept; the after-QC flag is the contract's mandatory admission filter, and the result is analysis-ready only under that default. before_qc and all are review modes: they warn, mark the result analysis_ready = FALSE, and drop (with a count) any retained-ineligible row that cannot be encoded — a null time, a dose without canonical mass, a route outside the compartment map, an infusion without a canonical rate. Every dropped-row count is on the "vmx" attribute so nothing disappears silently.

Value

A tibble in NONMEM layout. The "vmx" attribute is a list with data_version_id, time_basis, units, analyte, pk_specimen (the accepted PK scope, list(specimen_type =) or list(all_specimens = TRUE), or NULL when no specimen is recorded), eligibility, analysis_ready, cmt, endpoints (a tibble of dvid, name, cmt, unit, has_repeated_times — whether the endpoint carries any observation that shares its time — and, when the DataVersion records specimens, specimen_type (NA for a pooled endpoint) and all_specimens), dropped (named integer counts), and n_subjects. With replicate = TRUE the layout gains a REPLICATE column (see the argument).

See Also

vmx_model_data() for the underlying tidy tables.

---

vmx_nlmixr_default_cmt *Default route-to-compartment mapping for* vmx_nlmixr_data()

Description

Extravascular routes (`po`, `sc`, `im`) dose into compartment 1 (depot); intravenous routes (`iv-bolus`, `iv-infusion`) dose into compartment 2 (central); PK observations are read from compartment 2. Override by passing a named integer vector with the same names to `cmt`.

Usage

```
vmx_nlmixr_default_cmt()
```

`vmx_pd` *PD observations table*

Description

PD observations table

Usage

```
vmx_pd(dv, time_basis = NULL, client = vmx_client())
```

Arguments

`dv`	A data-version id or `vmx_data_version`.
`time_basis`	Optional time basis; see `vmx_data_version_table()`.
`client`	A `vmx_client`.

Value

A tibble.

`vmx_pd_marker` *Select a PD marker in one specimen or in all specimens*

Description

Builds one PD marker selection for `vmx_modeling_options()` and `vmx_model_build()`. Each selection names the marker's `gen_uuid` and either one specimen (for example `"serum"`) or `vmx_all_specimens()`. Two specimens of the same marker may be selected as two separate entries; selecting one marker both in a named specimen and in all specimens is an error.

Usage

```
vmx_pd_marker(gen_uuid, specimen, direction = NULL)
```

Arguments

gen_uuid	The PD marker's gen_uuid, as listed in the data version's pd_markers.
specimen	One specimen type string (a value of the marker's specimen_type, which may be "unspecified"), or vmx_all_specimens(). Required.
direction	The run-scoped response direction, "increasing" or "decreasing". Required by vmx_model_build(); vmx_modeling_options() ignores it, so the same selections can be previewed and then built.

Value

A vmx_pd_marker object.

Examples

```
vmx_pd_marker("8ffb3792-8bc9-4d38-b123-fc3e2170f6ee", "serum", "decreasing")
vmx_pd_marker(
  "8ffb3792-8bc9-4d38-b123-fc3e2170f6ee", vmx_all_specimens(), "decreasing"
)
```

---

vmx_pk *PK observations table*

Description

PK observations table

Usage

```
vmx_pk(dv, time_basis = NULL, client = vmx_client())
```

Arguments

dv	A data-version id or vmx_data_version.
time_basis	Optional time basis; see vmx_data_version_table().
client	A vmx_client.

Value

A tibble.

vmx_prep_answer *Answer prep questions and resume formatting*

Description

Answer prep questions and resume formatting

Usage

```
vmx_prep_answer(
  dataset,
  answers,
  client = vmx_client(),
  idempotency_key = NULL
)
```

Arguments

dataset	A dataset id or vmx_dataset.
answers	A named list mapping each prompt field to its answer value.
client	A vmx_client.
idempotency_key	Optional idempotency key for safely repeating the submission.

vmx_prep_questions *Questions raised by prep (when awaiting input)*

Description

Questions raised by prep (when awaiting input)

Usage

```
vmx_prep_questions(dataset, client = vmx_client())
```

Arguments

dataset	A dataset id or vmx_dataset.
client	A vmx_client.

Value

A tibble of pending questions. Variable-shape values such as options, default, and data_preview are retained as list-columns.

vmx_prep_status *Prep status for a dataset*

Description

Calls `GET /datasets/{ds_id}/prep-status`.

Usage

```
vmx_prep_status(dataset, client = vmx_client())
```

Arguments

dataset	A dataset id (`ds_...`) or `vmx_dataset`.
client	A `vmx_client`.

Value

A `vmx_prep_status` (`status`, and `data_version_id` once settled).

vmx_sim_cancel *Cancel a simulation job*

Description

Cancel a simulation job

Usage

```
vmx_sim_cancel(job, client = vmx_client())
```

Arguments

job	A job id or `vmx_simulation_job`.
client	A `vmx_client`.

Value

A `vmx_simulation_job`.

vmx_sim_existing_subject *Simulate existing (observed) subjects*

Description

POST /model-fits/{mf_id}/existing-subject-simulation-jobs.

Usage

```
vmx_sim_existing_subject(
  fit,
  dosing_input,
  subjects,
  idempotency_key = NULL,
  retried_from = NULL,
  min_timepoints = NULL,
  wait = FALSE,
  ...,
  client = vmx_client()
)
```

Arguments

fit	A fit id or vmx_model_fit.
dosing_input	A dosing-input id or vmx_dosing_input.
subjects	Subjects to simulate: a data.frame/tibble with gen_subject_uuid + subject_name columns, or a list of such records.
idempotency_key, retried_from	Optional create fields.
min_timepoints	Optional minimum number of simulated timepoints.
wait	If TRUE, block until the job settles.
...	Polling controls forwarded to vmx_wait().
client	A vmx_client.

Value

A vmx_simulation_job.

vmx_sim_existing_subject_from_text
Simulate existing subjects from dosing text

Description

POST /model-fits/{mf_id}/existing-subject-simulation-jobs/from-text.

Usage

```
vmx_sim_existing_subject_from_text(
  fit,
  dosing_text,
  subjects,
  idempotency_key = NULL,
  retried_from = NULL,
  min_timepoints = NULL,
  wait = FALSE,
  ...,
  client = vmx_client()
)
```

Arguments

fit	A fit id or vmx_model_fit.
dosing_text	The dosing regimen text.
subjects	Subjects to simulate: a data.frame/tibble with gen_subject_uuid + subject_name columns, or a list of such records.
idempotency_key, retried_from	Optional create fields.
min_timepoints	Optional minimum number of simulated timepoints.
wait	If TRUE, block until the job settles.
...	Polling controls forwarded to vmx_wait().
client	A vmx_client.

Value

A vmx_simulation_job.

`vmx_sim_hypothetical_subject` *Simulate hypothetical subjects*

Description

`POST /model-fits/{mf_id}/hypothetical-subject-simulation-jobs`.

Usage

```
vmx_sim_hypothetical_subject(
  fit,
  dosing_input,
  subjects,
  idempotency_key = NULL,
  retried_from = NULL,
  min_timepoints = NULL,
  wait = FALSE,
  ...,
  client = vmx_client()
)
```

Arguments

`fit`	A fit id or `vmx_model_fit`.
`dosing_input`	A dosing-input id or `vmx_dosing_input`.
`subjects`	A data.frame/tibble with a `subject_name` column plus one column per covariate, or a list of `{subject_name, covariates}` records.
`idempotency_key, retried_from`	Optional create fields.
`min_timepoints`	Optional minimum number of simulated timepoints.
`wait`	If `TRUE`, block until the job settles.
`...`	Polling controls forwarded to `vmx_wait()`.
`client`	A `vmx_client`.

Value

A `vmx_simulation_job`.

vmx_sim_hypothetical_subject_from_text
Simulate hypothetical subjects from dosing text

Description

POST /model-fits/{mf_id}/hypothetical-subject-simulation-jobs/from-text.

Usage

```
vmx_sim_hypothetical_subject_from_text(
  fit,
  dosing_text,
  subjects,
  idempotency_key = NULL,
  retried_from = NULL,
  min_timepoints = NULL,
  wait = FALSE,
  ...,
  client = vmx_client()
)
```

Arguments

fit	A fit id or vmx_model_fit.
dosing_text	The dosing regimen text.
subjects	A data.frame/tibble with a subject_name column plus one column per covariate, or a list of {subject_name, covariates} records.
idempotency_key, retried_from	Optional create fields.
min_timepoints	Optional minimum number of simulated timepoints.
wait	If TRUE, block until the job settles.
...	Polling controls forwarded to vmx_wait().
client	A vmx_client.

Value

A vmx_simulation_job.

vmx_sim_jobs *List simulation jobs for a model fit*

Description

List simulation jobs for a model fit

Usage

```
vmx_sim_jobs(fit, client = vmx_client())
```

Arguments

fit	A fit id or vmx_model_fit.
client	A vmx_client.

Value

A tibble containing all simulation jobs for the fit.

vmx_sim_population *Simulate a population scenario*

Description

POST /model-fits/{mf_id}/population-simulation-jobs.

Usage

```
vmx_sim_population(
  fit,
  dosing_input,
  scenario_name,
  idempotency_key = NULL,
  retried_from = NULL,
  min_timepoints = NULL,
  wait = FALSE,
  ...,
  client = vmx_client()
)
```

Arguments

fit	A fit id or vmx_model_fit.
dosing_input	A dosing-input id or vmx_dosing_input.
scenario_name	The population scenario name.
idempotency_key, retried_from	Optional create fields.
min_timepoints	Optional minimum number of simulated timepoints.
wait	If TRUE, block until the job settles.
...	Polling controls forwarded to vmx_wait().
client	A vmx_client.

Value

A vmx_simulation_job.

---

vmx_sim_population_from_text

Simulate a population scenario from dosing text

Description

POST /model-fits/{mf_id}/population-simulation-jobs/from-text.

Usage

```
vmx_sim_population_from_text(
  fit,
  dosing_text,
  scenario_name,
  idempotency_key = NULL,
  retried_from = NULL,
  min_timepoints = NULL,
  wait = FALSE,
  ...,
  client = vmx_client()
)
```

Arguments

fit	A fit id or vmx_model_fit.
dosing_text	The dosing regimen text.
scenario_name	The population scenario name.
idempotency_key, retried_from	Optional create fields.

min_timepoints	Optional minimum number of simulated timepoints.
wait	If TRUE, block until the job settles.
...	Polling controls forwarded to vmx_wait().
client	A vmx_client.

Value

A vmx_simulation_job.

---

vmx_sim_result *Simulation result*

Description

GET /simulation-jobs/{id}/result. Returns the parsed result payload containing model-implied response trajectories with the server-provided point statistic and interval. The nested wire shape is retained verbatim.

Usage

```
vmx_sim_result(job, grouping_variable = NULL, client = vmx_client())
```

Arguments

job	A job id or vmx_simulation_job.
grouping_variable	Optional server-side grouping.
client	A vmx_client.

Value

A list (the parsed result).

---

vmx_sim_status *Simulation job status*

Description

Simulation job status

Usage

```
vmx_sim_status(job, client = vmx_client())
```

Arguments

job	A job id (simjob_...) or vmx_simulation_job.
client	A vmx_client.

Value

A vmx_simulation_job.

---

vmx_studies *List studies for a treatment*

Description

List studies for a treatment

Usage

```
vmx_studies(
  treatment = NULL,
  status = NULL,
  client = vmx_client(),
  created_since = NULL
)
```

Arguments

treatment	A treatment id (tmt_...) or vmx_treatment; NULL lists across all treatments.
status	Optional status filter.
client	A vmx_client.
created_since	Optional lower creation-time bound: a POSIXct/Date or ISO-8601 string.

Value

A tibble containing all matching studies.

vmx_study *Fetch one study*

Description

Fetch one study

Usage

```
vmx_study(id, client = vmx_client())
```

Arguments

id	A study id (std_...) or vmx_study object.
client	A vmx_client.

Value

A vmx_study.

vmx_study_create *Create a study*

Description

Create a study

Usage

```
vmx_study_create(
  treatment,
  name,
  study_type = "clinical",
  phase = NULL,
  ...,
  client = vmx_client()
)
```

Arguments

treatment	A treatment id or vmx_treatment.
name	Study name.
study_type	Study type; defaults to "clinical".
phase	Optional clinical phase.
...	Additional fields (description, route_of_administration, pd_markers).
client	A vmx_client.

Value

A vmx_study.

---

vmx_study_update *Update a study*

Description

Only the fields you pass are changed (the server applies exclude_unset).

Usage

```
vmx_study_update(id, ..., client = vmx_client())
```

Arguments

id	A study id or vmx_study.
...	Fields to update.
client	A vmx_client.

Value

A vmx_study.

---

vmx_subjects *Subjects table (one row per subject)*

Description

Subjects table (one row per subject)

Usage

```
vmx_subjects(dv, time_basis = NULL, client = vmx_client())
```

Arguments

dv	A data-version id or vmx_data_version.
time_basis	Optional time basis; see vmx_data_version_table().
client	A vmx_client.

Value

A tibble.

vmx_torsten_data *Ragged-array data list for Stan / Torsten*

Description

Not yet implemented. The per-subject `start[i]`/`end[i]` index ranges and `iObs` observation index must be derived and verified against real data before shipping (this is the error-prone derivation the design flags); see the package NEWS. When implemented it should reuse the event assembly in `vmx_nlmixr_data()` (eligibility admission, basis selection, PK and PD specimen selection, dose and censoring encoding) and only add the ragged indexing on top.

Usage

```
vmx_torsten_data(dv, analyte = NULL, client = vmx_client())
```

Arguments

`dv`	A data-version id or `vmx_data_version`.
`analyte`	Analyte to assemble.
`client`	A `vmx_client`.

Value

Not yet implemented.

vmx_treatment *Fetch one treatment*

Description

Fetch one treatment

Usage

```
vmx_treatment(id, client = vmx_client())
```

Arguments

`id`	A treatment id (`tmt_...`) or `vmx_treatment` object.
`client`	A `vmx_client`.

Value

A `vmx_treatment`.

vmx_treatment_create *Create a treatment*

Description

Create a treatment

Usage

```
vmx_treatment_create(
  name,
  indication = NULL,
  description = NULL,
  client = vmx_client()
)
```

Arguments

name	Treatment name.
indication	Optional indication.
description	Optional free-text description.
client	A vmx_client.

Value

A vmx_treatment.

vmx_treatment_update *Update a treatment*

Description

Only the fields you pass are changed (the server applies exclude_unset).

Usage

```
vmx_treatment_update(id, ..., client = vmx_client())
```

Arguments

id	A treatment id or vmx_treatment.
...	Fields to update (name, indication, description, status).
client	A vmx_client.

Value

A vmx_treatment.

vmx_treatments *List treatments*

Description

List treatments

Usage

```
vmx_treatments(status = NULL, client = vmx_client())
```

Arguments

status	Optional status filter.
client	A vmx_client.

Value

A tibble containing all matching treatments.

vmx_upload *Upload files to a study*

Description

Streamed multipart upload. With wait = TRUE, blocks through the prep pipeline (see vmx_wait()).

Usage

```
vmx_upload(
  study,
  files,
  mode = c("initial", "incremental", "replacement"),
  treatment = NULL,
  config = NULL,
  wait = FALSE,
  client = vmx_client(),
  ...
)
```

Arguments

study	A study id or vmx_study.
files	Character vector of local file paths.
mode	One of "initial" (auto-formats a default DataVersion), "incremental", or "replacement".
treatment	Optional treatment; inferred from study when possible.
config	Optional gecodata v2 project.yaml path (warm-start).
wait	If TRUE, block until prep settles.
client	A vmx_client.
...	Polling controls forwarded to vmx_wait() when wait = TRUE.

Value

A vmx_dataset (status "uploaded").

---

vmx_upload_ignore *Ignore an upload within a dataset*

Description

POST /datasets/{ds_id}/ignore-upload. Soft-ignores one delivery.

Usage

```
vmx_upload_ignore(dataset, upload, client = vmx_client())
```

Arguments

dataset	A dataset id or vmx_dataset.
upload	The upload id (upl_...) to ignore.
client	A vmx_client.

Value

The updated vmx_prep_status.

vmx_upload_unignore *Reverse a prior upload ignore*

Description

POST /datasets/{ds_id}/unignore-upload.

Usage

```
vmx_upload_unignore(dataset, upload, client = vmx_client())
```

Arguments

dataset	A dataset id or vmx_dataset.
upload	The upload id (upl_...) to ignore.
client	A vmx_client.

Value

The updated vmx_prep_status.

vmx_wait *Block until an async handle reaches a terminal state*

Description

An S3 generic dispatching on the handle type. Each method has a sensible default terminal state. Terminal-but-unsuccessful states raise a classed error so scripts fail loudly rather than hang.

Usage

```
vmx_wait(
  x,
  until = NULL,
  timeout = 900,
  interval = 5,
  progress = interactive(),
  client = vmx_client(),
  ...
)
```

Arguments

x	A pollable handle: a dataset / prep-status or an NCA analysis (more types as the API surface lands: model-build-run, simulation-job).
until	Target terminal state(s); a sensible default per type when NULL. An explicitly requested failure state is returned; any other terminal failure still raises immediately.
timeout	Timeout in seconds. Resource methods use long-running defaults: 70 minutes for prep, NCA, and dosing input; 130 minutes for simulation; and 24 hours 10 minutes for model builds. The NCA/modeling worker defaults include a short persistence cushion beyond their execution ceilings; prep uses a client-side wait policy because its worker has no equivalent hard wall-clock ceiling.
interval	Positive poll interval in seconds (exponential backoff up to 30s).
progress	Show a progress message each poll; defaults to interactive().
client	A vmx_client.
...	Passed to methods.

Value

The updated object, or a vmx_timeout_error.

---

vmx_whoami *Confirm the configured PAT and report identity*

Description

Calls GET /me.

Usage

```
vmx_whoami(client = vmx_client())
```

Arguments

client	A vmx_client.

Value

A vmx_me object: user_id, email, name, workspace_id, roles, and counts.

Index

httr2::oauth_flow_device(), *18*

interactive(), *55*

print(), *5*

vmx_all_specimens, 3
vmx_all_specimens(), *20*, *29–31*, *34*, *36, 37*
vmx_analysis_log, 4
vmx_client, 5
vmx_client(), *18*
vmx_data_version, 6
vmx_data_version_archive, 7
vmx_data_version_create, 7
vmx_data_version_export, 8
vmx_data_version_export(), *12*
vmx_data_version_table, 8
vmx_data_version_table(), *14*, *26*, *36, 37*, *49*
vmx_data_version_unarchive, 9
vmx_data_versions, 10
vmx_dataset, 11
vmx_dataset_cancel, 11
vmx_dataset_download, 12
vmx_dataset_files, 12
vmx_dataset_tags, 13
vmx_datasets, 13
vmx_dosing, 14
vmx_dosing_input, 14
vmx_dosing_input_status, 15
vmx_fit_global_estimates, 15
vmx_fit_obs_vs_pred, 16
vmx_fit_subject_estimates, 16
vmx_fit_vpc, 17
vmx_health, 17
vmx_login, 18
vmx_login(), *5, 6*
vmx_marker_selection, 19
vmx_model_build, 20
vmx_model_build(), *3*, *30*, *36, 37*
vmx_model_build_artifacts, 21
vmx_model_build_cancel, 21
vmx_model_build_events, 22
vmx_model_build_export, 22
vmx_model_build_logs, 23
vmx_model_build_marker_selection *(*vmx_marker_selection*)*, 19
vmx_model_build_marker_selection(), *20*
vmx_model_build_report, 23
vmx_model_build_report_create, 24
vmx_model_build_results, 24
vmx_model_build_runs, 25
vmx_model_build_status, 25
vmx_model_catalog, 26
vmx_model_data, 26
vmx_model_data(), *35*
vmx_model_describe, 27
vmx_model_fit, 27
vmx_model_fit_marker_selection *(*vmx_marker_selection*)*, 19
vmx_model_fit_postprocessor_status, 28
vmx_model_fits, 28
vmx_modeling_options, 29
vmx_modeling_options(), *3*, *36, 37*
vmx_nca, 30
vmx_nca(), *3*
vmx_nca_analyses, 31
vmx_nca_get, 32
vmx_nca_marker_selection *(*vmx_marker_selection*)*, 19
vmx_nca_marker_selection(), *31*
vmx_nca_result, 32
vmx_nlmixr_data, 33
vmx_nlmixr_data(), *35*, *50*
vmx_nlmixr_default_cmt, 35
vmx_nlmixr_default_cmt(), *34*
vmx_pd, 36
vmx_pd_marker, 36
vmx_pd_marker(), *3, 4*, *20*, *29, 30*, *34*

vmx_pk, 37
vmx_prep_answer, 38
vmx_prep_questions, 38
vmx_prep_status, 39
vmx_sim_cancel, 39
vmx_sim_existing_subject, 40
vmx_sim_existing_subject_from_text, 41
vmx_sim_hypothetical_subject, 42
vmx_sim_hypothetical_subject_from_text, 43
vmx_sim_jobs, 44
vmx_sim_population, 44
vmx_sim_population_from_text, 45
vmx_sim_result, 46
vmx_sim_status, 46
vmx_studies, 47
vmx_study, 48
vmx_study_create, 48
vmx_study_update, 49
vmx_subjects, 49
vmx_torsten_data, 50
vmx_treatment, 50
vmx_treatment_create, 51
vmx_treatment_update, 51
vmx_treatments, 52
vmx_upload, 52
vmx_upload_ignore, 53
vmx_upload_unignore, 54
vmx_wait, 54
vmx_wait(), *7*, *14*, *20*, *30*, *40–43*, *45*, *46*, *52*, *53*
vmx_whoami, 55