Package: krt (via r-universe)

November 18, 2026

Title Author, Validate, and Export Key Resources Tables

Version 0.1.0

Description A toolkit for creating, importing, validating, enriching, rendering, and depositing Key Resources Tables (KRTs). A KRT lists the resources used and generated in a study (antibodies, cell lines, organisms, chemicals, software, datasets, protocols, and more), each paired with a persistent identifier such as a Research Resource Identifier (RRID), a Digital Object Identifier (DOI), a repository accession, or a catalog number, so that resources are unambiguously identifiable and machine-actionable. The package models resources as typed, validated records around a neutral core schema and maps them to journal or funder output profiles, following the FAIR (Findable, Accessible, Interoperable, Reusable) principles of Wilkinson et al. (2016) <doi:10.1038/sdata.2016.18>. It normalizes and optionally resolves identifiers against public registries, extracts resources from manuscripts, and renders tables both in the STAR (Structured, Transparent, Accessible Reporting) Methods style used by Cell Press journals and in the style required by ASAP (Aligning Science Across Parkinson's), with an emphasis on transparency, reproducibility, and correct per-component licensing.

License GPL-3

URL https://github.com/choxos/krt, https://choxos.github.io/krt/

BugReports https://github.com/choxos/krt/issues

Depends R (>= 4.1)

Imports digest, httr2, jsonlite, methods, stats, tools, utils, xml2, yaml

Suggests bslib, covr, DT, knitr, officer, openxlsx, pdftools, rdflib, rmarkdown, S4Vectors, shiny, spelling, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Config/roxygen2/version 8.0.0

LazyData true

NeedsCompilation no

Author Ahmad Sofi-Mahmudi [aut, cre] (ORCID:
<https://orcid.org/0000-0001-6829-0823>), Aligning Science
Across Parkinson's [cph] (Copyright holder of the ASAP Key
Resources Table schema (CC BY 4.0; Zenodo
doi:10.5281/zenodo.17917979) from which the bundled ASAP
profile is derived. ASAP does not endorse this package.)

Maintainer Ahmad Sofi-Mahmudi <a.sofimahmudi@gmail.com>

Config/pak/sysreqs libxml2-dev libssl-dev

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

Date/Publication 2026-09-12 14:20:08 UTC

RemoteUrl https://github.com/cran/krt

RemoteRef HEAD

RemoteSha 2ddfe2f0ec0e97886860c667d85b2a1d865d5f3a

Contents

add_resource 4
append_provenance 5
apply_mapping 6
as.data.frame.krt_tbl 6
as.data.frame.krt_validation_report 7
as_krt 7
as_prov_json 8
as_rdf 9
as_rocrate 9
coerce_field 10
compose_identifier 11
detect_existing_krt 11
export_asap 12
export_citation 13
export_krt 14
export_tabular 15
extract_candidates 16
extract_krt 16
extract_llm 17
fetch_fulltext 18
field_registry 19
find_duplicates 19
get_profile 20

id_parse . . . 21
import_asap . . . 21
import_krt . . . 22
import_tabular . . . 23
is_pending_identifier . . . 24
KRT-class . . . 24
krt_as_dataframe . . . 25
krt_attribution . . . 25
krt_audit_licenses . . . 26
krt_cli . . . 26
krt_deposit_figshare . . . 27
krt_deposit_zenodo . . . 28
krt_diff . . . 29
krt_example . . . 29
krt_import_elabftw . . . 30
krt_import_protocolsio . . . 31
krt_llm . . . 31
krt_merge . . . 32
krt_meta . . . 33
krt_plugin_api . . . 34
krt_plugins . . . 34
krt_preflight . . . 35
krt_profile_info . . . 35
krt_profile_license . . . 36
krt_profiles . . . 37
krt_provenance . . . 37
krt_suggest . . . 38
krt_validation_report . . . 38
krt_vocab . . . 39
krt_vocabularies . . . 39
krt_write_attribution . . . 40
launch_krt . . . 40
list_llm_providers . . . 41
list_resolvers . . . 41
list_suggest_sources . . . 42
list_validators . . . 42
load_profile . . . 43
mapping_lossy_fields . . . 43
new_approval . . . 44
new_contributor . . . 45
new_krt . . . 45
new_resource . . . 47
normalize_ids . . . 48
parse_compound_identifier . . . 48
project_profile . . . 49
read_input_text . . . 49
read_krt_json . . . 50
read_krt_yaml . . . 51

redact_krt . . . 51
redaction_default . . . 52
redaction_policy . . . 52
register_llm_provider . . . 53
register_profile . . . 53
register_resolver . . . 54
register_suggest_source . . . 55
register_validator . . . 55
render_krt . . . 56
resolve_id . . . 57
resolvers . . . 58
resource_signature . . . 59
rrid_type . . . 59
scan_identifiers . . . 60
summary.krt_tbl . . . 60
summary.krt_validation_report . . . 61
validate_krt . . . 61
validate_plugin_contract . . . 62
vocab_match . . . 63
write_krt_json . . . 63
write_krt_yaml . . . 64

Index **65**

`add_resource` *Add, update, remove, or get a resource in a KRT*

Description

Add, update, remove, or get a resource in a KRT

Usage

```
add_resource(x, ...)

update_resource(x, resource_id, ...)

remove_resource(x, resource_id)

get_resource(x, resource_id)
```

Arguments

`x`	A krt_tbl.
`...`	For `add_resource()`, either a single `krt_resource` (from `new_resource()`) or the arguments of `new_resource()` (`resource_type`, `display_name`, and named fields).
`resource_id`	The id of the resource to update, remove, or get.

Value

`add_resource()`, `update_resource()`, and `remove_resource()` return the modified `krt_tbl`; `get_resource()` returns a `krt_resource` or `NULL`.

Examples

```
k <- new_krt("Demo")
k <- add_resource(k, "Software/code", "R", version = "4.4.0",
                  new_or_reuse = "reuse", rrid = "RRID:SCR_001905")
get_resource(k, k$resources[[1]]$resource_id)
```

---

append_provenance *Append a provenance entry to a KRT*

Description

Records that an `activity` (for example `"normalize_ids"`, `"import"`, `"validate"`, `"resolve"`, `"export"`) was applied to the table. Called automatically by the mutating functions; exported so custom pipelines can record their own steps.

Usage

```
append_provenance(x, activity, inputs = NULL, outputs = NULL, params = NULL)
```

Arguments

`x`	A krt_tbl.
`activity`	A short activity label.
`inputs, outputs`	Optional character vectors of input/output identifiers.
`params`	Optional named list of parameters for the activity.

Value

The `krt_tbl` with the entry appended.

Examples

```
k <- append_provenance(new_krt("Demo"), "import", params = list(format = "csv"))
length(krt_provenance(k))
```

apply_mapping *Map one resource to a profile's columns*

Description

Map one resource to a profile's columns

Usage

```
apply_mapping(resource, profile)
```

Arguments

resource	A krt_resource.
profile	A krt_profile or profile name.

Value

A named list of column values.

Examples

```
r <- new_resource("Antibody", "Anti-TH", vendor = "Millipore",
                  catalog_number = "AB152", rrid = "RRID:AB_390204",
                  new_or_reuse = "reuse")
apply_mapping(r, "asap")[["IDENTIFIER"]]
```

as.data.frame.krt_tbl *Coerce a KRT to a data frame (a rectangular view)*

Description

Produces a rectangular *view* of the table by filling the union of present fields with NA. Many-valued fields are collapsed with "; " in this view; the lossless representation is the JSON/YAML export.

Usage

```
## S3 method for class 'krt_tbl'
as.data.frame(x, row.names = NULL, optional = FALSE, ..., view = "wide")
```

Arguments

x	A krt_tbl.
row.names, optional	Present for consistency with the as.data.frame() generic; not used.
...	Ignored.
view	The view to produce: "wide" (the full union of core fields) or a profile name such as "asap" (a profile projection). Must be named.

Value

A data frame with one row per resource.

Examples

```
k <- add_resource(new_krt("Demo"), "Dataset", "RNA-seq",
                  new_or_reuse = "new", doi = "10.5281/zenodo.123")
as.data.frame(k)
as.data.frame(krt_example, view = "asap")
```

---

as.data.frame.krt_validation_report *Convert a validation report to a data frame*

Description

Convert a validation report to a data frame

Usage

```
## S3 method for class 'krt_validation_report'
as.data.frame(x, ...)
```

Arguments

x	A krt_validation_report.
...	Ignored.

Value

A data frame with one row per finding.

---

as_krt *Coerce to and from the S4* KRT *class*

Description

These are inverse coercions named after their target class. as_krt() returns the S3 krt_tbl (the package's primary object): pass it a krt_tbl and it is returned unchanged, or an S4 KRT and it is converted down. It is the helper to call when a function should accept either representation. as_KRT() is the opposite direction, a thin idempotent wrapper over methods::as(x, "KRT") for Bioconductor workflows.

Usage

```
as_krt(x)

as_KRT(x)
```

Arguments

x	A krt_tbl or KRT object.

Value

`as_krt()` returns a `krt_tbl`; `as_KRT()` returns an S4 `KRT`.

Examples

```
k4 <- as_KRT(krt_example)
identical(length(as_krt(k4)$resources), length(krt_example$resources))
```

---

as_prov_json *Export the provenance graph as PROV-JSON*

Description

Serializes the table's recorded activities as a W3C PROV-JSON document: the table is an entity, each recorded step is an activity associated with the `krt` software agent and generating the table.

Usage

```
as_prov_json(x, path = NULL, audience = c("author", "public"))
```

Arguments

x	A krt_tbl.
path	Output path, or `NULL` to return the JSON string.
audience	`"author"` (default) keeps provenance parameters; `"public"` redacts the table first, so recorded parameters that could echo sensitive input values are dropped.

Value

The JSON string, or the path (invisibly).

Examples

```
cat(substr(as_prov_json(normalize_ids(krt_example)), 1, 40))
```

as_rdf *Export a KRT as RDF (JSON-LD or Turtle)*

Description

Export a KRT as RDF (JSON-LD or Turtle)

Usage

```
as_rdf(
  x,
  format = c("jsonld", "turtle"),
  path = NULL,
  audience = c("author", "public")
)
```

Arguments

x	A krt_tbl.
format	"jsonld" (default) or "turtle" (requires the rdflib package).
path	Output path, or NULL to return the text.
audience	"author" (default) or "public" (redacts the table first).

Value

The serialized RDF text, or the path (invisibly).

Examples

```
invisible(as_rdf(krt_example, format = "jsonld"))
```

as_rocrate *Export a KRT as an RO-Crate 1.1 metadata document*

Description

Describes the table as an RO-Crate Dataset whose parts are its resources, each carrying its name, type, and identifiers.

Usage

```
as_rocrate(x, path = NULL, audience = c("author", "public"))
```

Arguments

x	A krt_tbl.
path	Output path, or NULL to return the JSON-LD string.
audience	"author" (default) or "public" (redacts the table first).

Value

The JSON-LD string, or the path (invisibly).

Examples

```
cat(substr(as_rocrate(krt_example), 1, 40))
```

---

coerce_field *Coerce a field value to its declared type.*

Description

Light, non-destructive coercion used when building resource records. Enum fields are lower/normalized where the vocabulary is lowercase; unknown enum values are passed through unchanged (validation reports on them separately).

Usage

```
coerce_field(name, value)
```

Arguments

name	Field name.
value	The value to coerce.

Value

The coerced value.

Examples

```
coerce_field("new_or_reuse", "NEW")
coerce_field("pmid", 12345)
```

compose_identifier *Compose a compound identifier string from a resource's typed fields*

Description

Compose a compound identifier string from a resource's typed fields

Usage

```
compose_identifier(resource, order = NULL)
```

Arguments

resource	A krt_resource (or a named list of fields).
order	Optional character vector giving the field order; defaults to a sensible canonical order.

Value

A single "; "-joined identifier string.

Examples

```
r <- new_resource("Antibody", "Anti-TH", catalog_number = "AB152",
                  rrid = "RRID:AB_390204", new_or_reuse = "reuse")
compose_identifier(r)
```

detect_existing_krt *Detect and parse a Key Resources Table embedded in a document*

Description

Detect and parse a Key Resources Table embedded in a document

Usage

```
detect_existing_krt(doc)
```

Arguments

doc	The result of read_input_text().

Value

A krt_tbl parsed from the embedded table, or NULL if none is found.

Examples

```
detect_existing_krt(read_input_text("No table here."))
```

`export_asap` *Export a KRT in the ASAP six-column format*

Description

Projects the table to the ASAP columns and writes it as CSV or into an xlsx template. If `template` is supplied (or the bundled ASAP template is used), the data is written into its `KRT` sheet, preserving its dropdowns and attribution worksheet.

Usage

```
export_asap(
  x,
  path = NULL,
  template = NULL,
  format = NULL,
  audience = c("author", "public"),
  redact = NULL,
  attribution = TRUE
)
```

Arguments

`x`	A krt_tbl.
`path`	Output path, or `NULL` to return CSV text.
`template`	Optional xlsx template path. When supplied, output is xlsx.
`format`	`"csv"` or `"xlsx"`; inferred from `path`/`template`.
`audience`	`"author"` (full) or `"public"` (redacted).
`redact`	Redaction strength for public output, or `FALSE` to disable.
`attribution`	If `TRUE` (default) and a `path` is given, write the ASAP CC BY 4.0 attribution block as a sidecar next to it.

Value

The path (invisibly) when written, or CSV text.

Examples

```
cat(substr(export_asap(krt_example), 1, 60))
```

`export_citation` *Export citable resources as RIS or BibTeX*

Description

Only resources that are citable scholarly objects (datasets, software, protocols, or anything with a DOI) are emitted. This format cannot encode the full biological or ethics semantics of a KRT; use it for reference managers, not as an archival copy.

Usage

```
export_citation(
  x,
  path = NULL,
  format = c("ris", "bibtex"),
  audience = c("author", "public"),
  redact = NULL
)
```

Arguments

`x`	A krt_tbl.
`path`	Output path, or `NULL` to return the text.
`format`	`"ris"` or `"bibtex"`.
`audience`	`"author"` (full) or `"public"` (redacted).
`redact`	Redaction strength (`"basic"`/`"strict"`) for public output, or `FALSE` to disable; `NULL` uses the profile default.

Value

The path (invisibly) when written, or the citation text.

Examples

```
cat(export_citation(krt_example, format = "bibtex"))
```

export_krt *Export a Key Resources Table*

Description

Writes a table in a chosen format. "json" and "yaml" are lossless; "csv", "tsv", "xlsx", "asap", "ris", and "bibtex" are lossy views and raise a lossy-export warning listing fields that are not preserved as columns. When audience = "public", sensitive ethics fields are redacted by default.

Usage

```
export_krt(
  x,
  path = NULL,
  format = c("json", "yaml", "csv", "tsv", "xlsx", "asap", "ris", "bibtex"),
  profile = NULL,
  audience = c("author", "public"),
  redact = NULL,
  attribution = TRUE,
  template = NULL,
  view = NULL
)

krt_write(
  x,
  path = NULL,
  format = c("json", "yaml", "csv", "tsv", "xlsx", "asap", "ris", "bibtex"),
  profile = NULL,
  audience = c("author", "public"),
  redact = NULL,
  attribution = TRUE,
  template = NULL,
  view = NULL
)
```

Arguments

x	A krt_tbl.
path	Output file path, or NULL to return the content as a string.
format	Output format; inferred from path when possible.
profile	Profile whose columns tabular exports use (default the table's profile).
audience	"author" (full) or "public" (redacted).
redact	Redaction strength ("basic"/"strict") for public exports, or FALSE to disable (with a warning).
attribution	If TRUE (default) and the profile carries an attribution requirement, write an attribution sidecar next to path.

template	Optional template path for ASAP/xlsx export.
view	Optional explicit view for tabular formats.

Value

The path (invisibly) when written, otherwise the content string.

Examples

```
export_krt(krt_example, format = "json") |> substr(1, 30)
suppressWarnings(export_krt(krt_example, format = "asap")) |> substr(1, 40)
```

---

export_tabular *Export a KRT as a delimited table or spreadsheet*

Description

Export a KRT as a delimited table or spreadsheet

Usage

```
export_tabular(
  x,
  path = NULL,
  format = c("csv", "tsv", "xlsx"),
  profile = NULL,
  view = NULL,
  audience = c("author", "public"),
  redact = NULL,
  attribution = TRUE
)
```

Arguments

x	A krt_tbl.
path	Output path, or NULL to return the content (csv/tsv only).
format	"csv", "tsv", or "xlsx".
profile	Profile whose columns to use (default the wide core view).
view	Optional explicit view name.
audience	"author" (full) or "public" (redacted).
redact	Redaction strength for public output, or FALSE to disable.
attribution	If TRUE (default), a path is given, and profile carries an attribution requirement, write it as a sidecar next to path.

Value

The path (invisibly) when written, or the delimited text.

Examples

```
cat(substr(export_tabular(krt_example, format = "csv"), 1, 60))
```

`extract_candidates` *Extract candidate resources from text (regex engine)*

Description

Extract candidate resources from text (regex engine)

Usage

```
extract_candidates(text)
```

Arguments

`text`	A character vector of manuscript text.

Value

A list of `krt_resource` candidates.

Examples

```
cand <- extract_candidates("Anti-TH (RRID:AB_390204); FIJI (RRID:SCR_002285)")
length(cand)
```

`extract_krt` *Extract a Key Resources Table from a manuscript*

Description

Reads a manuscript (PDF, JATS/NISO XML, DOCX, or text), extracts candidate resources with the deterministic regex engine (default) or an LLM, and returns them as a validated krt_tbl. Every result is provenance-stamped with the engine used.

Usage

```
extract_krt(
  input,
  engine = c("regex", "llm"),
  format = NULL,
  profile = "generic",
  llm = NULL,
  resolve = FALSE,
  existing = c("detect", "ignore"),
  title = NULL
)
```

Arguments

input	A file path or text string.
engine	"regex" (deterministic, offline) or "llm" (requires a configured provider and API key).
format	Optional explicit input format.
profile	Profile to assign to the extracted table.
llm	A krt_llm() configuration (for the LLM engine).
resolve	Whether to resolve identifiers during validation.
existing	"detect" to also parse an embedded KRT, or "ignore".
title	Optional title for the extracted table.

Value

A list with krt (the extracted krt_tbl), candidates (the raw resource candidates), existing_krt (a parsed embedded table or NULL), and report (a validation report).

Examples

```
res <- extract_krt("Anti-TH (RRID:AB_390204); FIJI (RRID:SCR_002285).")
nrow(as.data.frame(res$krt))
```

---

extract_llm *Extract resources from text with an LLM*

Description

Sends the text to the configured provider and parses the returned JSON into candidate resources. Requires an API key (from the environment); returns an empty list if none is configured or the call fails.

Usage

```
extract_llm(text, llm = krt_llm(), ...)
```

Arguments

text	Manuscript text.
llm	A krt_llm() configuration.
...	Reserved.

Value

A list of krt_resource candidates.

Examples

```
# Uses a mock provider so no network or key is needed:
register_llm_provider("mock",
  function(prompt, llm) '[{"resource_type":"Software/code","display_name":"R"}]')
extract_llm("text", krt_llm("openai"))  # empty without a key
extract_llm("text", structure(list(provider = "mock"), class = "krt_llm"))
```

---

fetch_fulltext *Fetch manuscript full text*

Description

Retrieves full text (JATS XML where available) from Europe PMC or bioRxiv, for feeding into extract_krt(). Returns NULL offline or when the text is not openly available.

Usage

```
fetch_fulltext(id, source = c("auto", "europepmc", "biorxiv"), timeout = 30)
```

Arguments

id	A PMCID, DOI, or bioRxiv DOI.
source	"auto", "europepmc", or "biorxiv".
timeout	Request timeout in seconds.

Value

A character string of the full text or abstract, or NULL.

Examples

```
# Contacts Europe PMC. Offline, or when the text is not openly available,
# this returns NULL rather than failing.
txt <- fetch_fulltext("PMC5334499")
is.null(txt) || nchar(txt) > 0
```

field_registry *Key Resources Table field registry*

Description

The field registry declares every field a resource record may carry, together with its group, type, cardinality, and the resource types it applies to.

Usage

```
all_fields()

field_spec(name)

fields_for_type(resource_type)

field_label(name, locale = NULL)
```

Arguments

name	A field name.
resource_type	A resource type (one of krt_resource_types()).
locale	Optional locale (currently advisory; labels fall back to a title-cased field name).

Value

all_fields() returns a named list of field specifications; field_spec() returns one specification (or NULL); fields_for_type() returns the names of fields that apply to a resource type; field_label() returns a human-readable label.

Examples

```
str(field_spec("rrid"))
fields_for_type("Antibody")
```

find_duplicates *Find duplicate resources in a KRT*

Description

Find duplicate resources in a KRT

Usage

```
find_duplicates(x, fuzzy = FALSE)
```

Arguments

x	A krt_tbl.
fuzzy	If TRUE, also report near-duplicates whose display name and vendor are close by edit distance.

Value

A list of duplicate groups; each element is a character vector of resource_ids that share a signature (or are near-duplicates).

Examples

```
k <- new_krt("Demo")
k <- add_resource(k, "Antibody", "Anti-TH", vendor = "Millipore",
                  catalog_number = "AB152", new_or_reuse = "reuse")
k <- add_resource(k, "Antibody", "Anti-TH (dup)", vendor = "Millipore",
                  catalog_number = "AB152", new_or_reuse = "reuse")
find_duplicates(k)
```

---

get_profile *Retrieve a registered profile*

Description

Retrieve a registered profile

Usage

```
get_profile(name)
```

Arguments

name	Profile name.

Value

A krt_profile object.

Examples

```
get_profile("asap")$columns
```

id_parse *Parse and classify an identifier string*

Description

Detects the scheme of a single identifier token (DOI, ORCID, ROR, RRID, PMID, PMCID, URL, or a database accession) using the bundled syntax patterns.

Usage

```
id_parse(idstring)
```

Arguments

idstring	A single identifier token.

Value

A list with `scheme` (character, or `NA` if unrecognized), `value` (the cleaned identifier), and `field` (the resource field the value belongs in, e.g. `"doi"`, `"rrid"`, `"accession"`).

Examples

```
id_parse("https://doi.org/10.5281/zenodo.123")
id_parse("RRID:AB_390204")
id_parse("GSE12345")
```

import_asap *Import an ASAP or Cell Press Key Resources Table*

Description

Import an ASAP or Cell Press Key Resources Table

Usage

```
import_asap(path, sheet = 1, title = NULL)
```

Arguments

path	A file path (csv/tsv/xlsx) or a data frame already read in.
sheet	Worksheet (for xlsx).
title	Optional title for the resulting table.

Value

A krt_tbl; profile "asap" for a six-column ASAP sheet, or "star-methods" for a Cell Press three-column table.

Examples

```
f <- tempfile(fileext = ".csv")
writeLines(export_asap(krt_example), f)
k <- import_asap(f)
length(k$resources)
```

---

import_krt *Import a Key Resources Table from a file or string*

Description

Detects the format (JSON, YAML, ASAP/Cell Press table, or generic tabular) and reads it into a krt_tbl.

Usage

```
import_krt(input, format = NULL, profile = NULL, mapping = NULL, sheet = 1)
```

Arguments

input	A file path, or a JSON/YAML string.
format	One of "json", "yaml", "asap", "tabular"; auto-detected when NULL.
profile	Optional profile to assign to the imported table.
mapping	Optional column-to-field mapping for tabular import.
sheet	Worksheet (for xlsx).

Value

A krt_tbl.

Examples

```
k <- import_krt(write_krt_json(krt_example))
length(k$resources)
```

`import_tabular` *Import a generic tabular Key Resources Table*

Description

Import a generic tabular Key Resources Table

Usage

```
import_tabular(
  path,
  mapping = NULL,
  sheet = 1,
  profile = "generic",
  title = NULL
)
```

Arguments

`path`	A file path (csv/tsv/xlsx) or a data frame.
`mapping`	Optional named list mapping column names to core field names; guessed from the headers when `NULL`.
`sheet`	Worksheet (for xlsx).
`profile`	Profile to assign to the imported table.
`title`	Optional title.

Value

A krt_tbl.

Examples

```
df <- data.frame(type = "Antibody", name = "Anti-TH", rrid = "RRID:AB_1",
                 "new/reuse" = "reuse", check.names = FALSE)
import_tabular(df)$resources[[1]]$resource_type
```

is_pending_identifier *Is a value the ASAP "identifier pending" placeholder?*

Description

Is a value the ASAP "identifier pending" placeholder?

Usage

```
is_pending_identifier(str)
```

Arguments

str	A candidate identifier string.

Value

TRUE if the string matches the "Identifier from ... pending" convention.

Examples

```
is_pending_identifier("Identifier from Cellosaurus pending")
```

KRT-class *The* KRT *S4 class*

Description

An S4 wrapper around a krt_tbl, for interoperability with Bioconductor. Convert with as(x, "KRT") and as(y, "krt_tbl"), or as_krt().

Slots

schema_version,profile,table_id,title Character metadata.

resources,approvals,contributors,provenance Record lists.

metadata A list of remaining metadata (study type, locale, timestamps).

Examples

```
k4 <- methods::as(krt_example, "KRT")
methods::is(k4, "KRT")
```

krt_as_dataframe *Coerce a KRT's resources to a DataFrame (Bioconductor)*

Description

Coerce a KRT's resources to a DataFrame (Bioconductor)

Usage

```
krt_as_dataframe(x, view = "wide")
```

Arguments

x	A krt_tbl.
view	The view to project (default "wide").

Value

An S4Vectors::DataFrame of the resources (requires the S4Vectors package).

Examples

```
if (requireNamespace("S4Vectors", quietly = TRUE)) krt_as_dataframe(krt_example)
```

krt_attribution *Attribution text for a table's profile*

Description

Returns the attribution block that should accompany outputs produced with the table's profile. For the ASAP profile (CC BY 4.0) this is the required attribution; for profiles with no redistribution obligation it is a short note.

Usage

```
krt_attribution(x)
```

Arguments

x	A krt_tbl or a profile name.

Value

A character string.

Examples

```
cat(krt_attribution(krt_example))
```

krt_audit_licenses *Audit the licenses of the package and its bundled assets*

Description

Audit the licenses of the package and its bundled assets

Usage

```
krt_audit_licenses()
```

Value

A data frame with one row per licensable component (the package code, each profile, the bundled reference data and license text), giving its source, license, DOI, whether it is redistributable, and notes.

Examples

```
krt_audit_licenses()
```

krt_cli *Run the krt command-line interface*

Description

Dispatches a subcommand. Used by the `krt` shell script (`system.file("scripts", "krt", package = "krt")`); can also be called directly with a character vector of arguments.

Usage

```
krt_cli(args = commandArgs(trailingOnly = TRUE))
```

Arguments

`args`	Command-line arguments (defaults to those passed to `Rscript`).

Value

Invisibly an integer status code (0 success, 1 validation failure or runtime error, 2 usage error such as an unknown command or a missing required `--out`).

Examples

```
krt_cli("audit-licenses")
f <- tempfile(fileext = ".json")
writeLines(write_krt_json(krt_example), f)
krt_cli(c("validate", f))
```

`krt_deposit_figshare` *Deposit a KRT to Figshare*

Description

Creates a Figshare article, reserves a DOI, and uploads the table as JSON. The DOI is recorded in the table's provenance. Deposits default to the redacted public audience.

Usage

```
krt_deposit_figshare(
  x,
  token = Sys.getenv("FIGSHARE_TOKEN"),
  sandbox = TRUE,
  metadata = NULL,
  audience = c("public", "author"),
  timeout = 60
)
```

Arguments

`x`	A krt_tbl.
`token`	Figshare token (defaults to `FIGSHARE_TOKEN`).
`sandbox`	Use the Figshare sandbox (default `TRUE`).
`metadata`	Optional article metadata.
`audience`	`"public"` (redacted, the default) or `"author"` (full).
`timeout`	Request timeout in seconds.

Value

A list with `article` (the API response), `doi`, `uploaded` (logical), and `x`.

Examples

```
## Not run:
# Not executable without a Figshare (sandbox) account and access token, so
# this example cannot be run during a check.
krt_deposit_figshare(krt_example)

## End(Not run)
```

`krt_deposit_zenodo` *Deposit a KRT to Zenodo*

Description

Creates a Zenodo deposition, uploads the table as JSON, sets metadata, and optionally publishes. The minted (pre-reserved) DOI is written back into the table's provenance and `table_id`. Deposits default to the redacted public audience, since a deposit is a shared artifact.

Usage

```
krt_deposit_zenodo(
  x,
  token = Sys.getenv("ZENODO_TOKEN"),
  sandbox = TRUE,
  metadata = NULL,
  publish = FALSE,
  audience = c("public", "author"),
  timeout = 60
)
```

Arguments

`x`	A krt_tbl.
`token`	Zenodo access token (defaults to `ZENODO_TOKEN`).
`sandbox`	Use the Zenodo sandbox (default `TRUE`).
`metadata`	Optional Zenodo metadata list (a sensible default otherwise).
`publish`	Whether to publish immediately (default `FALSE`).
`audience`	`"public"` (redacted, the default) or `"author"` (full).
`timeout`	Request timeout in seconds.

Value

A list with `deposit` (the API response), `doi`, `uploaded` and `published` (per-stage status), and `x` (the table, with the DOI recorded only when every stage succeeded). On failure, `deposit` is `NULL`.

Examples

```
## Not run:
# Not executable without a Zenodo (sandbox) account and access token, so
# this example cannot be run during a check.
res <- krt_deposit_zenodo(krt_example, sandbox = TRUE)
res$doi

## End(Not run)
```

krt_diff *Diff two Key Resources Tables*

Description

Compares resources by identity signature and reports which were added, removed, or changed (with field-level deltas).

Usage

```
krt_diff(x, y)
```

Arguments

x, y	Two krt_tbl objects (x is the baseline).

Value

A `krt_diff` object with `print()` and `as.data.frame()` methods.

Examples

```
a <- add_resource(new_krt("A"), "Dataset", "D", doi = "10.5281/zenodo.1",
                  new_or_reuse = "new")
b <- update_resource(a, a$resources[[1]]$resource_id, notes = "added")
as.data.frame(krt_diff(a, b))
```

krt_example *An example Key Resources Table*

Description

A small krt_tbl spanning several common resource types (antibody, cell line, software, dataset, organism/strain, and protocol). Used throughout the documentation and tests so that examples run offline.

Usage

```
krt_example
```

Format

An object of class `krt_tbl` with six resources.

Source

Constructed by `data-raw/06-build-example-krt.R`. Identifiers are real, publicly resolvable examples; the study is fictional.

Examples

```
krt_example
summary(krt_example)
```

`krt_import_elabftw` *Import resources from an eLabFTW experiment*

Description

Fetches an experiment and extracts candidate resources from its body text.

Usage

```
krt_import_elabftw(
  base_url,
  experiment_id,
  token = Sys.getenv("ELABFTW_TOKEN"),
  timeout = 30
)
```

Arguments

`base_url`	The eLabFTW instance base URL (e.g. `https://elab.example.org`).
`experiment_id`	The experiment id.
`token`	API token (defaults to `ELABFTW_TOKEN`).
`timeout`	Request timeout in seconds.

Value

A krt_tbl of extracted resources.

Examples

```
## Not run:
# Not executable without an eLabFTW instance and an API token, so this
# example cannot be run during a check.
krt_import_elabftw("https://elab.example.org", experiment_id = 42)

## End(Not run)
```

`krt_import_protocolsio` *Import a protocol from protocols.io*

Description

Fetches a protocol and records it as a Protocol resource.

Usage

```
krt_import_protocolsio(
  id,
  token = Sys.getenv("PROTOCOLSIO_TOKEN"),
  timeout = 30
)
```

Arguments

`id`	A protocols.io protocol id or DOI.
`token`	API token (defaults to `PROTOCOLSIO_TOKEN`).
`timeout`	Request timeout in seconds.

Value

A krt_tbl with a single Protocol resource.

Examples

```
# Contacts the public protocols.io API. Offline, this warns and returns an
# empty table rather than failing.
k <- suppressWarnings(krt_import_protocolsio("kxygx3w"))
length(k$resources)
```

`krt_llm` *Configure an LLM for extraction*

Description

Configure an LLM for extraction

Usage

```
krt_llm(
  provider = c("openai", "anthropic", "gemini", "openai_compat"),
  model = NULL,
  base_url = NULL,
  api_key = NULL,
  temperature = 0,
  max_tokens = 4096
)

## S3 method for class 'krt_llm'
format(x, ...)

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

Arguments

provider	One of the registered providers ("openai", "anthropic", "gemini", "openai_compat" for local/OpenAI-compatible servers).
model	Model id (a sensible default per provider when NULL).
base_url	Base URL for "openai_compat" (e.g. a local server).
api_key	API key; defaults to the provider's environment variable.
temperature, max_tokens	Generation parameters.
x	A krt_llm object.
...	Ignored.

Value

A krt_llm configuration object.

Examples

```
krt_llm("openai", model = "gpt-4o-mini")$provider
```

---

krt_merge *Merge Key Resources Tables*

Description

Combines resources from two or more tables, matching by normalized identity signature. Matching resources are merged field by field; the strategy resolves field-level conflicts. Conflicts are attached to the result as the "merge_conflicts" attribute.

Usage

```
krt_merge(
  x,
  y,
  ...,
  strategy = c("union", "prefer_x", "prefer_y", "manual"),
  by = resource_signature
)
```

Arguments

x, y, ...	Tables to merge (x supplies the result's metadata).
strategy	"union"/"prefer_x" (x wins conflicts), "prefer_y" (later table wins), or "manual" (x wins but every conflict is recorded).
by	A function computing a resource's match key (default resource_signature()).

Value

The merged krt_tbl.

Examples

```
a <- add_resource(new_krt("A"), "Antibody", "Anti-TH", vendor = "Millipore",
                  catalog_number = "AB152", new_or_reuse = "reuse")
b <- add_resource(new_krt("B"), "Dataset", "D", doi = "10.5281/zenodo.1",
                  new_or_reuse = "new")
krt_merge(a, b)
```

---

krt_meta *Get or set KRT metadata*

Description

Get or set KRT metadata

Usage

```
krt_meta(x)

krt_meta(x) <- value
```

Arguments

x	A krt_tbl.
value	A named list of metadata fields to set (any of title, profile, study_type, locale).

Value

`krt_meta()` returns a named list of the table's metadata.

Examples

```
k <- new_krt("Demo")
krt_meta(k)$title
krt_meta(k) <- list(title = "Renamed")
```

`krt_plugin_api` *The krt plugin API*

Description

The krt plugin API

Usage

```
krt_plugin_api()
```

Value

A data frame describing each plugin kind, its registration function, and its contract.

Examples

```
krt_plugin_api()
```

`krt_plugins` *Extend krt with plugins*

Description

krt is extensible through five registries. Register custom components with the corresponding function; `krt_plugin_api()` lists the entry points and their contracts, and `validate_plugin_contract()` checks an object before you register it.

See Also

`register_profile()`, `register_validator()`, `register_resolver()`, `register_llm_provider()`, `register_suggest_source()`.

krt_preflight *Pre-flight release readiness check for a Key Resources Table*

Description

Runs the checks that together decide whether a table is ready to share or deposit, and returns a single machine-readable verdict plus a human-readable checklist: profile validation (no errors), lossless JSON round-trip, a working public (redacted) export that leaks none of the policy's dropped fields, attribution availability, and whether the profile projection would drop fields.

Usage

```
krt_preflight(x, profile = NULL)
```

Arguments

x	A krt_tbl.
profile	Profile to check against (defaults to the table's profile).

Value

A `krt_preflight` object: a list with `profile`, `ok` (TRUE when no check fails), and `checks` (a data frame of `check`, `status`, `detail`), with `print()`, `format()`, and `as.data.frame()` methods.

Examples

```
pf <- krt_preflight(krt_example)
pf$ok
```

krt_profile_info *Describe a profile*

Description

Describe a profile

Usage

```
krt_profile_info(name)
```

Arguments

name	A profile name or a `krt_profile`.

Value

The `krt_profile` (invisibly); prints a human-readable summary including its license and redistribution status.

Examples

```
krt_profile_info("asap")
```

`krt_profile_license` *Licensing of a profile*

Description

Licensing of a profile

Usage

```
krt_profile_license(name)

krt_profile_sources(name)

krt_profile_attribution(name)
```

Arguments

`name`	A profile name or `krt_profile`.

Value

`krt_profile_license()` returns the SPDX license id; `krt_profile_sources()` returns the source metadata list; `krt_profile_attribution()` returns the attribution text (or `NULL`).

Examples

```
krt_profile_license("asap")
krt_profile_sources("asap")$doi
```

krt_profiles *List available profiles*

Description

List available profiles

Usage

```
krt_profiles()
```

Value

A data frame with one row per registered profile (name, title, license, whether it is officially endorsed).

Examples

```
krt_profiles()
```

krt_provenance *Provenance of a KRT*

Description

Returns the ordered provenance entries as a `krt_provenance` object (a list of `krt_prov_entry`, so `length()` gives the number of steps). It has `print()` and `as.data.frame()` methods; serialize the provenance graph with `as_prov_json()` and `as_rocrate()`.

Usage

```
krt_provenance(x)
```

Arguments

x	A krt_tbl.

Value

A `krt_provenance` object.

Examples

```
krt_provenance(normalize_ids(krt_example))
```

krt_suggest *Suggest canonical names and identifiers from public authorities*

Description

Queries ontology and registry search endpoints to autocomplete a resource's canonical name or identifier. Requires network access; returns an empty result offline.

Usage

```
krt_suggest(query, type = NULL, authority = "auto", n = 10, resolve = TRUE)
```

Arguments

query	The text to search for.
type	Optional resource type hint.
authority	Which authority to query: "auto", or the name of any registered source (built-ins: "taxonomy", "cellosaurus", "chebi", "ror"; plus any added with register_suggest_source())
n	Maximum number of suggestions.
resolve	Whether to contact the network (default TRUE).

Value

A data frame with columns label, id, authority, score, uri.

Examples

```
krt_suggest("dopamine", authority = "chebi", resolve = FALSE)
```

krt_validation_report *The validation report object*

Description

validate_krt() returns a krt_validation_report. It has print(), summary(), and as.data.frame() methods. valid is TRUE when there are no error findings.

See Also

validate_krt()

krt_vocab *Access a KRT controlled vocabulary*

Description

Access a KRT controlled vocabulary

Usage

```
krt_vocab(key)
```

Arguments

key	Vocabulary name (e.g. "resource_type", "new_or_reuse", "status").

Value

A character vector of allowed values.

Examples

```
krt_vocab("new_or_reuse")
```

krt_vocabularies *KRT controlled vocabularies*

Description

Convenience accessors for the controlled vocabularies used across the package.

Usage

```
krt_resource_types()

krt_new_or_reuse()

krt_statuses()

krt_approval_types()

krt_roles()

krt_redaction_levels()
```

Value

A character vector of allowed values.

Examples

```
krt_resource_types()
krt_new_or_reuse()
```

`krt_write_attribution` *Write attribution to a file*

Description

Write attribution to a file

Usage

```
krt_write_attribution(x, path)
```

Arguments

`x`	A krt_tbl or profile name.
`path`	Output file path.

Value

The path, invisibly.

Examples

```
f <- tempfile(fileext = ".md")
krt_write_attribution(krt_example, f)
```

`launch_krt` *Launch the interactive KRT editor*

Description

Starts a Shiny application for importing, editing (in the generic view), validating, normalizing identifiers, and exporting a Key Resources Table. Requires the `shiny`, `bslib`, and `DT` packages.

Usage

```
launch_krt(...)
```

Arguments

`...`	Passed to `shiny::runApp()`.

Value

Called for its side effect (runs the app).

Examples

```
if (interactive()) launch_krt()
```

---

list_llm_providers *List registered LLM providers*

Description

List registered LLM providers

Usage

```
list_llm_providers()
```

Value

A character vector of provider names.

Examples

```
list_llm_providers()
```

---

list_resolvers *List registered resolver schemes*

Description

List registered resolver schemes

Usage

```
list_resolvers()
```

Value

A character vector of scheme names.

Examples

```
list_resolvers()
```

list_suggest_sources *List autocomplete sources*

Description

List autocomplete sources

Usage

```
list_suggest_sources()
```

Value

A character vector of source names.

Examples

```
list_suggest_sources()
```

list_validators *List registered validation rules*

Description

List registered validation rules

Usage

```
list_validators()
```

Value

A data frame of registered rules (id, layer, default severity, standard).

Examples

```
head(list_validators())
```

load_profile *Load a profile from a directory*

Description

Reads `schema.yml`, `mappings.yml`, and (optionally) `validation.yml`, `provenance.json`, and `ATTRIBUTION.md` from a profile directory.

Usage

```
load_profile(path)

is_profile(x)
```

Arguments

`path`	Path to the profile directory.
`x`	An object to test.

Value

A `krt_profile` object.

Examples

```
p <- load_profile(system.file("extdata", "profiles", "asap", package = "krt"))
p$columns
```

mapping_lossy_fields *Fields lost or folded when projecting to a profile*

Description

Returns the core fields that are present in the table but are not preserved as their own column by the profile (they are either folded into a free-text catch-all column or dropped). These drive the lossy-export warning.

Usage

```
mapping_lossy_fields(x, profile)
```

Arguments

`x`	A krt_tbl.
`profile`	A `krt_profile` or profile name.

Value

A character vector of field names (empty for a lossless profile).

Examples

```
mapping_lossy_fields(krt_example, "asap")
```

`new_approval` *Create and add an ethics or governance approval*

Description

Create and add an ethics or governance approval

Usage

```
new_approval(approval_type, ..., .id = NULL)

add_approval(x, ...)
```

Arguments

`approval_type`	One of `krt_approval_types()` (e.g. `"IACUC"`, `"IRB"`, `"REB"`, `"ethics"`).
`...`	Additional named fields: `board_name`, `protocol_number`, `institution_name`, `institution_ror`, `jurisdiction`, `approved_on`, `consent_obtained`, `consent_scope`, `data_use_restrictions`, `redaction_level`.
`.id`	Optional explicit approval id.
`x`	A krt_tbl.

Value

`new_approval()` returns a `krt_approval`; `add_approval()` returns the updated krt_tbl.

Examples

```
a <- new_approval("IACUC", protocol_number = "2026-017",
                  board_name = "Example IACUC")
k <- add_approval(new_krt("Demo"), a)
```

new_contributor *Create and add a contributor*

Description

Create and add a contributor

Usage

```
new_contributor(name, ..., .id = NULL)

add_contributor(x, ...)
```

Arguments

name	Contributor name.
...	Additional named fields: orcid, role (see krt_roles()), affiliation, affiliation_ror.
.id	Optional explicit contributor id.
x	A krt_tbl.

Value

new_contributor() returns a krt_contributor; add_contributor() returns the updated krt_tbl.

Examples

```
k <- add_contributor(new_krt("Demo"), "Ada Researcher",
                     orcid = "0000-0002-1825-0097", role = "author")
```

new_krt *Create a Key Resources Table*

Description

Constructs an empty (or pre-populated) krt_tbl: the neutral, typed core object of the package. Resources, approvals, and contributors are stored as present-only records; the rectangular table an author sees is a *view* produced on demand by as.data.frame().

Usage

```
new_krt(
  title = NULL,
  profile = "generic",
  study_type = NULL,
  locale = NULL,
  resources = list(),
  approvals = list(),
  contributors = list()
)

krt_new(
  title = NULL,
  profile = "generic",
  study_type = NULL,
  locale = NULL,
  resources = list(),
  approvals = list(),
  contributors = list()
)

is_krt(x)
```

Arguments

`title`	A short table or study title.
`profile`	Output profile name (default `"generic"`); see `krt_profiles()`.
`study_type`	Optional character vector describing the study (e.g. `c("wet-lab", "computational")`).
`locale`	Optional locale string (e.g. `"en-US"`).
`resources`, `approvals`, `contributors`	Optional lists of records to seed the table with.
`x`	An object to test.

Value

An object of class `krt_tbl`.

Examples

```
k <- new_krt("Example study", study_type = "wet-lab")
k <- add_resource(k, "Antibody", "Rabbit Anti-TH", vendor = "Millipore",
                  catalog_number = "AB152", rrid = "RRID:AB_390204",
                  new_or_reuse = "reuse")
k
```

new_resource *Create a resource record*

Description

Builds a single krt_resource: a flat, present-only record describing one research resource. Identifier fields are stored separately by type (catalog_number, rrid, doi, ...); they are only combined into a compound identifier string at export time.

Usage

```
new_resource(
  resource_type,
  display_name = NULL,
  ...,
  .id = NULL,
  .validate = TRUE
)

is_resource(x)
```

Arguments

resource_type	One of krt_resource_types().
display_name	The resource name as it appears in the manuscript.
...	Additional named fields (see all_fields() for the vocabulary), for example vendor, catalog_number, rrid, doi, new_or_reuse, notes.
.id	Optional explicit resource id; generated from content if omitted.
.validate	If TRUE (default), check resource_type against the controlled vocabulary and warn about unknown fields.
x	An object to test.

Value

An object of class krt_resource.

Examples

```
new_resource("Antibody", "Rabbit Anti-TH", vendor = "Millipore",
             catalog_number = "AB152", rrid = "RRID:AB_390204",
             new_or_reuse = "reuse")
```

normalize_ids *Normalize identifiers to canonical forms*

Description

Canonicalizes the identifier fields of a table, a resource, or a bare character vector: strips resolver prefixes from DOIs, hyphenates ORCIDs, ensures the `RRID:` prefix, and so on. Applied consistently, this prevents the same identifier from appearing in several syntactic forms.

Usage

```
normalize_ids(x, ...)
```

Arguments

`x`	A krt_tbl, a `krt_resource`, or a character vector of identifiers.
`...`	Ignored.

Value

An object of the same type as `x`, with identifiers normalized.

Examples

```
normalize_ids("https://doi.org/10.1038/SDATA.2016.18")
r <- new_resource("Software/code", "Fiji", rrid = "SCR_002285",
                  new_or_reuse = "reuse")
normalize_ids(r)$rrid
```

parse_compound_identifier *Parse a compound identifier string into typed fields*

Description

Splits an ASAP-style `IDENTIFIER` value (parts joined by `;` or newlines) and classifies each part into a resource field.

Usage

```
parse_compound_identifier(str)
```

Arguments

`str`	A compound identifier string.

Value

A named list of fields (e.g. `catalog_number`, `rrid`, `doi`, `accession`, `url`), plus `other` for unclassified parts.

Examples

```
parse_compound_identifier("Cat# AB152; RRID:AB_390204")
```

`project_profile` *Project a KRT onto a profile as a data frame*

Description

Project a KRT onto a profile as a data frame

Usage

```
project_profile(x, profile)
```

Arguments

`x`	A krt_tbl.
`profile`	A `krt_profile` or profile name.

Value

A data frame with the profile's columns, one row per resource.

Examples

```
project_profile(krt_example, "asap")[, c("RESOURCE TYPE", "IDENTIFIER")]
```

`read_input_text` *Read manuscript text and tables from an input*

Description

Read manuscript text and tables from an input

Usage

```
read_input_text(input, format = NULL)
```

Arguments

input	A file path (pdf/xml/jats/docx/txt) or a plain-text string.
format	Optional explicit format; auto-detected when NULL.

Value

A list with text (character), tables (list of data frames), and format.

Examples

```
read_input_text("We used FIJI (RRID:SCR_002285).")$text
```

---

read_krt_json *Read a KRT from canonical JSON*

Description

Read a KRT from canonical JSON

Usage

```
read_krt_json(input)
```

Arguments

input	A file path or a JSON string.

Value

A krt_tbl.

Examples

```
k <- read_krt_json(write_krt_json(krt_example))
identical(length(k$resources), length(krt_example$resources))
```

read_krt_yaml *Read a KRT from canonical YAML*

Description

Read a KRT from canonical YAML

Usage

```
read_krt_yaml(input)
```

Arguments

input	A file path or a YAML string.

Value

A krt_tbl.

Examples

```
k <- read_krt_yaml(write_krt_yaml(krt_example))
identical(length(k$resources), length(krt_example$resources))
```

redact_krt *Redact sensitive fields for public sharing*

Description

Removes or generalizes fields flagged by the redaction policy, so a table can be shared publicly without exposing internal ethics or consent details.

Usage

```
redact_krt(x, level = c("basic", "strict"), policy = NULL)
```

Arguments

x	A krt_tbl.
level	"basic" (default) removes basic-tagged fields; "strict" removes basic- and strict-tagged fields.
policy	An optional policy data frame overriding redaction_policy().

Value

The redacted krt_tbl.

Examples

```
k <- add_approval(new_krt("Demo"), "IRB", protocol_number = "IRB-1",
                  consent_scope = "study-specific")
redact_krt(k)$approvals[[1]]$protocol_number
```

`redaction_default` *Default redaction strength for a profile's public exports*

Description

Default redaction strength for a profile's public exports

Usage

```
redaction_default(profile = NULL)
```

Arguments

`profile`	A profile name or `krt_profile`. A profile may declare a `redaction_default` in its `schema.yml`; otherwise the strength is `"basic"`.

Value

`"basic"` or `"strict"`.

Examples

```
redaction_default("asap")
```

`redaction_policy` *The default redaction policy*

Description

The default redaction policy

Usage

```
redaction_policy()
```

Value

A data frame with columns `scope` (`"approval"` or `"resource"`), `field`, `level` (the strip strength at which the field is removed: `"basic"` fields are removed at both `"basic"` and `"strict"`; `"strict"` fields only at `"strict"`), and `action` (`"drop"` or `"generalize"`).

Examples

```
redaction_policy()
```

register_llm_provider *Register an LLM provider*

Description

Register an LLM provider

Usage

```
register_llm_provider(name, request_fn, parse_fn = NULL, replace = FALSE)
```

Arguments

name	Provider name (e.g. "openai", "anthropic", a local endpoint).
request_fn	A function function(prompt, llm) returning the model's text output, or NULL on failure.
parse_fn	Optional custom parser function(text); defaults to JSON array extraction.
replace	Overwrite an existing provider named name? Defaults to FALSE so a plugin cannot silently replace a built-in provider.

Value

Invisibly NULL.

Examples

```
register_llm_provider("echo", function(prompt, llm) "[]", replace = TRUE)
"echo" %in% list_llm_providers()
```

register_profile *Register an output profile*

Description

Register an output profile

Usage

```
register_profile(name = NULL, path = NULL, profile = NULL, replace = FALSE)
```

Arguments

name	Profile name. If omitted, taken from the profile's schema.yml or the supplied object.
path	Path to a profile directory containing schema.yml and mappings.yml (loaded lazily), or NULL.
profile	A pre-built krt_profile object, or NULL.
replace	Overwrite a profile already registered under name? Defaults to FALSE, so a plugin cannot silently replace a built-in profile (such as asap); pass TRUE to deliberately override one.

Value

Invisibly the profile name.

Examples

```
krt_profiles()
```

---

register_resolver *Register an identifier resolver*

Description

Register an identifier resolver

Usage

```
register_resolver(scheme, fn, replace = FALSE)
```

Arguments

scheme	The identifier scheme (e.g. "rrid", "doi", "orcid").
fn	A function function(id, resolve = TRUE, ...) returning a normalized result list with at least input, normalized, and resolved.
replace	Overwrite an existing resolver for scheme? Defaults to FALSE so a plugin cannot silently replace a built-in resolver.

Value

Invisibly NULL.

Examples

```
"rrid" %in% list_resolvers()
```

register_suggest_source *Register an autocomplete source*

Description

Register an autocomplete source

Usage

```
register_suggest_source(name, fn, replace = FALSE)
```

Arguments

name	Source name (e.g. "taxonomy", "ror").
fn	A function function(query, n) returning a data frame with columns label, id, authority, score, uri.
replace	Overwrite an existing source named name? Defaults to FALSE so a plugin cannot silently replace a built-in source.

Value

Invisibly NULL.

Examples

```
"ror" %in% list_suggest_sources()
```

register_validator *Register a validation rule*

Description

Adds a rule to the validation engine. A rule is a function fn(x, ctx) that inspects a krt_tbl and returns a list of issues (each created with the internal issue helper); the engine attaches the rule id, layer, standard, and resolved severity.

Usage

```
register_validator(
  rule_id,
  fn,
  layer = c("structural", "semantic"),
  severity = c("error", "warning", "note", "info"),
  applies = function(x) TRUE,
  standard = NA_character_,
  replace = FALSE
)
```

Arguments

rule_id	A unique rule identifier, e.g. "struct-missing-name".
fn	The rule function function(x, ctx) returning a list of issues.
layer	Either "structural" or "semantic".
severity	Default severity: one of "error", "warning", "note", "info".
applies	A predicate function(x); the rule runs only when it returns TRUE (used by conditional rule packs).
standard	Optional reporting standard the rule enforces (e.g. "cell-line-auth-minimum").
replace	Overwrite an existing rule with the same rule_id? Defaults to FALSE, so a plugin cannot silently replace a built-in rule; pass TRUE to deliberately override one.

Value

Invisibly NULL; called for its side effect.

Examples

```
# A rule inspects the table and returns a list of issues. This demonstration
# rule reports nothing, so registering it leaves validation results unchanged.
demo_rule <- function(x, ctx) list()
register_validator("demo-no-op", demo_rule, layer = "semantic",
                   severity = "note", replace = TRUE)
"demo-no-op" %in% list_validators()$rule_id
validate_krt(krt_example)$valid
```

---

render_krt *Render a KRT as a formatted table*

Description

Produces a human-readable Key Resources Table. The "star-methods" profile projects the table to the three Cell Press columns (REAGENT or RESOURCE, SOURCE, IDENTIFIER) and groups resources under the twelve standard STAR Methods category headers, in the order the template uses. The "generic" profile (the default) renders the ASAP six-column layout; any other named profile renders through its own declared columns.

Usage

```
render_krt(
  x,
  path = NULL,
  format = c("md", "html", "docx"),
  profile = NULL,
  template = NULL,
  audience = c("author", "public"),
  redact = NULL
)
```

Arguments

x	A krt_tbl.
path	Output path, or NULL to return the text (md/html).
format	"md", "html", or "docx".
profile	Profile controlling the layout (default the table's profile).
template	Unused placeholder for a future Word template.
audience	"author" (full) or "public" (redacted) for shared tables.
redact	Redaction strength for public output, or FALSE to disable.

Value

The rendered text (md/html), or the path (invisibly) for docx.

Examples

```
cat(substr(render_krt(krt_example, profile = "star-methods"), 1, 80))
```

---

resolve_id *Resolve any identifier by detecting its scheme*

Description

Resolve any identifier by detecting its scheme

Usage

```
resolve_id(id, resolve = TRUE, ...)
```

Arguments

id	An identifier string.
resolve	Whether to contact the registry.
...	Passed to the scheme-specific resolver.

Value

A resolver result list, or NULL if the scheme is unsupported.

Examples

```
resolve_id("RRID:AB_390204", resolve = FALSE)$normalized
```

resolvers *Resolve identifiers against public registries*

Description

Each resolver normalizes an identifier and, when `resolve = TRUE` and the registry is reachable, retrieves a display name and type. All calls degrade gracefully: offline or on error, `resolved` is `FALSE`.

Usage

```
resolve_rrid(rrid, resolve = TRUE, timeout = 15)

resolve_doi(doi, resolve = TRUE, timeout = 15)

resolve_orcid(orcid, resolve = TRUE, timeout = 15)

resolve_pubmed(pmid, resolve = TRUE, timeout = 15)

resolve_ror(ror, resolve = TRUE, timeout = 15)

resolve_cellosaurus(cvcl, resolve = TRUE, timeout = 15)
```

Arguments

`rrid`, `doi`, `orcid`, `pmid`, `ror`, `cvcl`	The identifier to resolve.
`resolve`	Whether to contact the registry (default `TRUE`). Set `FALSE` for a purely offline, normalize-only result.
`timeout`	Request timeout in seconds.

Value

A list with `input`, `normalized`, `resolved`, `source`, `name`, `type`, and `url`.

Examples

```
# Offline: normalize only (no network).
resolve_rrid("RRID:AB_390204", resolve = FALSE)
resolve_doi("10.1038/sdata.2016.18", resolve = FALSE)$normalized

# Live lookup. This contacts Crossref; offline it simply returns the
# normalized identifier with `resolved = FALSE` instead of failing.
resolve_doi("10.1038/sdata.2016.18", resolve = TRUE)$resolved
```

resource_signature *Normalized signature of a resource*

Description

Builds a lowercased, trimmed tuple from the identifying fields (resource_type, vendor, catalog_number, lot_number, Two resources with the same signature are considered duplicates.

Usage

```
resource_signature(resource)
```

Arguments

resource	A krt_resource or named list.

Value

A single signature string.

Examples

```
r <- new_resource("Antibody", "Anti-TH", vendor = "Millipore",
                  catalog_number = "AB152", new_or_reuse = "reuse")
resource_signature(r)
```

rrid_type *Resource type implied by an RRID*

Description

Resource type implied by an RRID

Usage

```
rrid_type(rrid)
```

Arguments

rrid	An RRID string (with or without the RRID: prefix).

Value

The implied resource type (character) or NA if the authority is unknown.

Examples

```
rrid_type("RRID:AB_390204")
rrid_type("CVCL_0063")
```

scan_identifiers *Scan free text for research identifiers*

Description

Finds RRIDs, DOIs, catalog numbers, database accessions, and PMIDs in a block of text and classifies each. Used by the regex extraction engine and available on its own.

Usage

```
scan_identifiers(text)
```

Arguments

text	A character vector of text.

Value

A data frame with columns `value`, `field`, `type` (an inferred resource type, or `NA`), and `confidence` (`"high"` for precisely anchored schemes such as RRID/DOI/accession/PMID, `"medium"` for looser catalog-number matches) so the results can be triaged before acceptance.

Examples

```
scan_identifiers("We used anti-TH (RRID:AB_390204) and FIJI (RRID:SCR_002285).")
```

summary.krt_tbl *Summarize a KRT*

Description

Summarize a KRT

Usage

```
## S3 method for class 'krt_tbl'
summary(object, ...)
```

Arguments

object	A krt_tbl.
...	Ignored.

Value

A data frame with resource counts per resource type, including the number of newly generated versus reused resources.

Examples

```
summary(krt_example)
```

`summary.krt_validation_report` *Summarize a validation report*

Description

Summarize a validation report

Usage

```
## S3 method for class 'krt_validation_report'
summary(object, ...)
```

Arguments

`object`	A `krt_validation_report`.
`...`	Ignored.

Value

A data frame of finding counts by severity, layer, and standard.

`validate_krt` *Validate a Key Resources Table*

Description

Runs the registered validation rules over a table and returns a krt_validation_report. Structural rules check schema conformance offline; semantic rules check cross-field consistency and, when `resolve = TRUE`, identifier existence. Conditional packs (cell-line authentication, organism metadata, ethics/consent) apply only when the relevant resource types are present; they are minimal checks, not full ICLAC or ARRIVE assessments.

Usage

```
validate_krt(
  x,
  profile = NULL,
  layers = c("structural", "semantic"),
  resolve = FALSE,
  severity = NULL,
  attach = FALSE
)
```

Arguments

x	A krt_tbl.
profile	Profile whose severity overrides apply (default the table's profile).
layers	Which layers to run: "structural", "semantic", or both.
resolve	If TRUE, semantic rules may perform online existence checks (off by default; never on CRAN).
severity	Optional named list mapping rule_id to a severity (or "off"), overriding rule and profile defaults.
attach	If TRUE, return the table with the findings stored in its validation slot instead of returning the report.

Details

Each finding's severity is resolved from the rule default, then any profile override, then any per-rule value in severity. A severity of "off" disables the rule.

Value

A krt_validation_report, or the krt_tbl when attach = TRUE.

Examples

```
validate_krt(krt_example)
validate_krt(krt_example, layers = "structural")
```

---

validate_plugin_contract

Check that an object satisfies a plugin contract

Description

Check that an object satisfies a plugin contract

Usage

```
validate_plugin_contract(kind, obj)
```

Arguments

kind	One of "profile", "validator", "resolver", "llm_provider", "suggest_source".
obj	The plugin object or function to check.

Value

Invisibly TRUE; errors early if the contract is not met.

Examples

```
validate_plugin_contract("validator", function(x, ctx) list())
validate_plugin_contract("suggest_source", function(query, n) NULL)
```

---

vocab_match *Match a value against a controlled vocabulary*

Description

Match a value against a controlled vocabulary

Usage

```
vocab_match(value, vocab, fuzzy = FALSE)
```

Arguments

value	A character value to check.
vocab	A character vector of allowed values, or the name of a vocabulary (resolved with krt_vocab()).
fuzzy	If TRUE, when there is no exact match return the nearest vocabulary term (by edit distance) as a suggestion instead of NA.

Value

A list with ok (logical), value (the matched canonical term or NA), and suggestion (nearest term when fuzzy and not matched).

Examples

```
vocab_match("Antibody", "resource_type")
vocab_match("antibodies", "resource_type", fuzzy = TRUE)
```

---

write_krt_json *Write a KRT to canonical JSON*

Description

Write a KRT to canonical JSON

Usage

```
write_krt_json(x, path = NULL, pretty = TRUE)
```

Arguments

x	A krt_tbl.
path	Output file path, or NULL to return the JSON as a string.
pretty	Whether to pretty-print (default TRUE).

Value

The JSON string (invisibly, the path when written to a file).

Examples

```
json <- write_krt_json(krt_example)
substr(json, 1, 40)
```

---

write_krt_yaml *Write a KRT to canonical YAML*

Description

Write a KRT to canonical YAML

Usage

```
write_krt_yaml(x, path = NULL)
```

Arguments

x	A krt_tbl.
path	Output file path, or NULL to return the YAML as a string.

Value

The YAML string (invisibly, the path when written to a file).

Examples

```
cat(substr(write_krt_yaml(krt_example), 1, 40))
```

Index

∗ **datasets**
krt_example, 29

add_approval *(*new_approval*)*, 44
add_contributor *(*new_contributor*)*, 45
add_resource, 4
all_fields *(*field_registry*)*, 19
all_fields(), *47*
append_provenance, 5
apply_mapping, 6
as.data.frame(), *6*, *38*, *45*
as.data.frame.krt_tbl, 6
as.data.frame.krt_validation_report, 7
as_KRT *(*as_krt*)*, 7
as_krt, 7
as_krt(), *24*
as_prov_json, 8
as_prov_json(), *37*
as_rdf, 9
as_rocrate, 9
as_rocrate(), *37*

coerce_field, 10
compose_identifier, 11

detect_existing_krt, 11

export_asap, 12
export_citation, 13
export_krt, 14
export_tabular, 15
extract_candidates, 16
extract_krt, 16
extract_krt(), *18*
extract_llm, 17

fetch_fulltext, 18
field_label *(*field_registry*)*, 19
field_registry, 19
field_spec *(*field_registry*)*, 19
fields_for_type *(*field_registry*)*, 19
find_duplicates, 19
format.krt_llm *(*krt_llm*)*, 31

get_profile, 20
get_resource *(*add_resource*)*, 4

id_parse, 21
import_asap, 21
import_krt, 22
import_tabular, 23
is_krt *(*new_krt*)*, 45
is_pending_identifier, 24
is_profile *(*load_profile*)*, 43
is_resource *(*new_resource*)*, 47

KRT-class, 24
krt_approval_types *(*krt_vocabularies*)*, 39
krt_approval_types(), *44*
krt_as_dataframe, 25
krt_attribution, 25
krt_audit_licenses, 26
krt_cli, 26
krt_deposit_figshare, 27
krt_deposit_zenodo, 28
krt_diff, 29
krt_example, 29
krt_import_elabftw, 30
krt_import_protocolsio, 31
krt_llm, 31
krt_llm(), *17*, *18*
krt_merge, 32
krt_meta, 33
krt_meta<- *(*krt_meta*)*, 33
krt_new *(*new_krt*)*, 45
krt_new_or_reuse *(*krt_vocabularies*)*, 39
krt_plugin_api, 34
krt_plugin_api(), *34*
krt_plugins, 34
krt_preflight, 35

krt_profile_attribution (krt_profile_license), 36
krt_profile_info, 35
krt_profile_license, 36
krt_profile_sources (krt_profile_license), 36
krt_profiles, 37
krt_provenance, 37
krt_redaction_levels (krt_vocabularies), 39
krt_resource_types (krt_vocabularies), 39
krt_resource_types(), *19*, *47*
krt_roles (krt_vocabularies), 39
krt_roles(), *45*
krt_statuses (krt_vocabularies), 39
krt_suggest, 38
krt_tbl, *4–17*, *20*, *22–25*, *27–31*, *33*, *35*, *37*, *40*, *43–45*, *48–51*, *55*, *57*, *60*, *62*, *64*
krt_tbl (new_krt), 45
krt_validation_report, 38, *61*
krt_vocab, 39
krt_vocab(), *63*
krt_vocabularies, 39
krt_write (export_krt), 14
krt_write_attribution, 40

launch_krt, 40
list_llm_providers, 41
list_resolvers, 41
list_suggest_sources, 42
list_validators, 42
load_profile, 43

mapping_lossy_fields, 43

new_approval, 44
new_contributor, 45
new_krt, 45
new_resource, 47
new_resource(), *4*
normalize_ids, 48

parse_compound_identifier, 48
print.krt_llm (krt_llm), 31
project_profile, 49

read_input_text, 49
read_input_text(), *11*
read_krt_json, 50
read_krt_yaml, 51
redact_krt, 51
redaction_default, 52
redaction_policy, 52
redaction_policy(), *51*
register_llm_provider, 53
register_llm_provider(), *34*
register_profile, 53
register_profile(), *34*
register_resolver, 54
register_resolver(), *34*
register_suggest_source, 55
register_suggest_source(), *34*, *38*
register_validator, 55
register_validator(), *34*
remove_resource (add_resource), 4
render_krt, 56
resolve_cellosaurus (resolvers), 58
resolve_doi (resolvers), 58
resolve_id, 57
resolve_orcid (resolvers), 58
resolve_pubmed (resolvers), 58
resolve_ror (resolvers), 58
resolve_rrid (resolvers), 58
resolvers, 58
resource_signature, 59
resource_signature(), *33*
rrid_type, 59

scan_identifiers, 60
shiny::runApp(), *40*
summary(), *38*
summary.krt_tbl, 60
summary.krt_validation_report, 61

update_resource (add_resource), 4

validate_krt, 61
validate_krt(), *38*
validate_plugin_contract, 62
validate_plugin_contract(), *34*
vocab_match, 63

write_krt_json, 63
write_krt_yaml, 64