Package: multiverse.internals (via r-universe)

March 9, 2025

Title Internal Infrastructure for R-multiverse

Description R-multiverse requires this internal infrastructure package to automate contribution reviews and populate universes.

Version 0.4.2

License MIT + file LICENSE

URL https://r-multiverse.org/multiverse.internals/, https://github.com/r-multiverse/multiverse.internals

BugReports https://github.com/r-multiverse/multiverse.internals/issues

Depends R (>= 4.4.0)

Imports base64enc, desc, gh, igraph, jsonlite, nanonext, pkgsearch, rversions, stats, utils, vctrs, yaml

Suggests gert, testthat (>= 3.0.0)

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/testthat/edition 3

Config/pak/sysreqs git libxml2-dev libssl-dev

Repository https://r-multiverse-staging.r-universe.dev

RemoteUrl https://github.com/r-multiverse/multiverse.internals

RemoteRef ae919e610a6dc01bd85cee85166b2637a0569bd9

RemoteSha ae919e610a6dc01bd85cee85166b2637a0569bd9

Contents

`date_snapshot` *Snapshot date*

Description

Get the target snapshot date of the current or most recent Staging period.

Usage

```
date_snapshot(today = Sys.Date())
```

Arguments

`today`	A `"Date"` object with today's date, or an object that `as.Date()` can convert to a `"Date"` object.

Value

Character string in `yyyy-mm-dd` format.

See Also

Other snapshot metadata: `date_staging()`, `r_version_staging()`

Examples

```
date_snapshot()
```

date_staging *Staging date*

Description

Get the start date of the current or most recent Staging period.

Usage

```
date_staging(today = Sys.Date())
```

Arguments

today	A "Date" object with today's date, or an object that as.Date() can convert to a "Date" object.

Value

Character string in yyyy-mm-dd format.

See Also

Other snapshot metadata: date_snapshot(), r_version_staging()

Examples

```
date_staging()
```

interpret_status *Interpret the status of a package*

Description

Summarize the issues of a package in human-readable text.

Usage

```
interpret_status(package, issues)
```

Arguments

package	Character string, name of the package.
issues	A list with one issue per package. Obtained by reading the results of record_issues().

Value

A character string summarizing the issues of a package in prose.

See Also

Other status: update_status()

---

issues_checks *Report issues from R-universe package check results.*

Description

Check R-universe package check results.

Usage

```
issues_checks(meta = meta_checks())
```

Arguments

meta	A data frame with R-universe package check results returned by meta_checks().

Details

issues_checks() reads output from the R-universe check API to scan all R-multiverse packages for issues that may have happened during building and testing.

Value

A named list of information about packages which do not comply with DESCRPTION checks. Each name is a package name, and each element contains specific information about non-compliance.

Package issues

Functions like issues_versions() and issues_descriptions() perform health checks for all packages in R-multiverse. For a complete list of checks, see the issues_*() functions listed at https://r-multiverse.org/multiverse.internals/reference/index.html. record_versions() updates the version number history of releases in R-multiverse, and record_issues() gathers together all the issues about R-multiverse packages.

See Also

Other issues: issues_dependencies(), issues_descriptions(), issues_versions()

Examples

```
meta <- meta_checks(repo = "https://wlandau.r-universe.dev")
issues <- issues_checks(meta = meta)
str(issues)
```

issues_dependencies *Report package dependency issues*

Description

Flag packages which have issues in their strong dependencies (`Imports:`, `Depends:`, and `LinkingTo:` in the `DESCRIPTION`.) These include indirect/upstream dependencies, as well, not just the explicit mentions in the `DESCRIPTION` file.

Usage

```
issues_dependencies(packages, meta = meta_packages(), verbose = FALSE)
```

Arguments

`packages`	Character vector of names of packages with other issues.
`meta`	A data frame with R-universe package check results returned by `meta_checks()`.
`verbose`	`TRUE` to print progress while checking issues with dependencies, `FALSE` otherwise.

Value

A nested list of problems triggered by dependencies. The names of top-level elements are packages affected downstream. The value of each top-level element is a list whose names are Each element of this inner list is a character vector of relevant dependencies of the downstream package.

For example, consider a linear dependency graph where `crew.cluster` depends on `crew`, `crew` depends on `mirai`, and `mirai` depends on `nanonext`. We represent the graph like this: `nanonext -> mirai -> crew -> crew.cluster`. If `nanonext` has an issue, then `issues_dependencies()` returns `list(crew.cluster = list(nanonext = "crew"), ...)`, where `...` stands for additional named list entries. From this list, we deduce that `nanonext` is causing an issue affecting `crew.cluster` through the direct dependency on `crew`.

The choice in output format from `issues_dependencies()` allows package maintainers to more easily figure out which direct dependencies are contributing issues and drop those direct dependencies if necessary.

Package issues

Functions like `issues_versions()` and `issues_descriptions()` perform health checks for all packages in R-multiverse. For a complete list of checks, see the `issues_*()` functions listed at https://r-multiverse.org/multiverse.internals/reference/index.html. `record_versions()` updates the version number history of releases in R-multiverse, and `record_issues()` gathers together all the issues about R-multiverse packages.

See Also

Other issues: `issues_checks()`, `issues_descriptions()`, `issues_versions()`

Examples

```
meta <- meta_packages(repo = "https://wlandau.r-universe.dev")
issues_dependencies(packages = character(0L), meta = meta)
issues_dependencies(packages = "crew.aws.batch", meta = meta)
issues_dependencies(packages = "nanonext", meta = meta)
issues_dependencies(packages = "crew", meta = meta)
issues_dependencies(packages = c("crew", "mirai"), meta = meta)
```

---

issues_descriptions *Report* DESCRIPTION*-level issues.*

Description

Report issues with DESCRIPTION-level metadata of packages.

Usage

```
issues_descriptions(meta = meta_packages())
```

Arguments

meta	A data frame with R-universe package check results returned by meta_checks().

Details

issues_descriptions() reports specific issues in the DESCRIPTION-level metadata of packages:

1. Security advisories at https://github.com/RConsortium/r-advisory-database.
2. Licenses that cannot be verified as free and open-source.
3. The presence of a "Remotes" field.
4. A lower version number on R-multiverse than on CRAN, if the package is also published there.

Value

A named list of information about packages which do not comply with DESCRPTION checks. Each name is a package name, and each element contains specific information about non-compliance.

Package issues

Functions like issues_versions() and issues_descriptions() perform health checks for all packages in R-multiverse. For a complete list of checks, see the issues_*() functions listed at https://r-multiverse.org/multiverse.internals/reference/index.html. record_versions() updates the version number history of releases in R-multiverse, and record_issues() gathers together all the issues about R-multiverse packages.

See Also

Other issues: issues_checks(), issues_dependencies(), issues_versions()

Examples

```
meta <- meta_packages(repo = "https://wlandau.r-universe.dev")
issues <- issues_descriptions(meta = meta)
str(issues)
```

`issues_versions` *Check package versions.*

Description

Check package version number history for compliance.

Usage

```
issues_versions(versions)
```

Arguments

`versions`	Character of length 1, file path to a JSON manifest tracking the history of released versions of packages.

Details

This function checks the version number history of packages in R-multiverse and reports any packages with issues. The current released version of a given package must be unique, and it must be greater than all the versions of all the previous package releases.

Value

A named list of information about packages which do not comply with version number history checks. Each name is a package name, and each element contains specific information about version non-compliance: the current version number, the current version hash, and the analogous versions and hashes of the highest-versioned release recorded.

Package issues

Functions like issues_versions() and issues_descriptions() perform health checks for all packages in R-multiverse. For a complete list of checks, see the issues_*() functions listed at https://r-multiverse.org/multiverse.internals/reference/index.html. record_versions() updates the version number history of releases in R-multiverse, and record_issues() gathers together all the issues about R-multiverse packages.

See Also

Other issues: issues_checks(), issues_dependencies(), issues_descriptions()

Examples

```
lines <- c(
  "[",
  " {",
  " \"package\": \"package_unmodified\",",
  " \"version_current\": \"1.0.0\",",
  " \"hash_current\": \"hash_1.0.0\",",
  " \"version_highest\": \"1.0.0\",",
  " \"hash_highest\": \"hash_1.0.0\"",
  " },",
  " {",
  " \"package\": \"version_decremented\",",
  " \"version_current\": \"0.0.1\",",
  " \"hash_current\": \"hash_0.0.1\",",
  " \"version_highest\": \"1.0.0\",",
  " \"hash_highest\": \"hash_1.0.0\"",
  " },",
  " {",
  " \"package\": \"version_incremented\",",
  " \"version_current\": \"2.0.0\",",
  " \"hash_current\": \"hash_2.0.0\",",
  " \"version_highest\": \"2.0.0\",",
  " \"hash_highest\": \"hash_2.0.0\"",
  " },",
  " {",
  " \"package\": \"version_unmodified\",",
  " \"version_current\": \"1.0.0\",",
  " \"hash_current\": \"hash_1.0.0-modified\",",
  " \"version_highest\": \"1.0.0\",",
  " \"hash_highest\": \"hash_1.0.0\"",
  " }",
  "]"
)
versions <- tempfile()
writeLines(lines, versions)
out <- issues_versions(versions)
str(out)
```

`meta_checks` *List metadata about R-universe package checks*

Description

List package checks results reported by the R-universe package API.

Usage

```
meta_checks(repo = "https://community.r-multiverse.org")
```

Arguments

repo	Character of length 1, URL of the package repository. R-multiverse uses "https://community.r-multiv

Value

A data frame with one row per package and columns with package check details.

See Also

Other meta: meta_packages()

Examples

```
meta_checks(repo = "https://wlandau.r-universe.dev")
```

---

meta_packages *List package metadata*

Description

List package metadata in an R universe.

Usage

```
meta_packages(
  repo = "https://community.r-multiverse.org",
  fields = c("Version", "License", "Remotes", "RemoteSha")
)
```

Arguments

repo	Character of length 1, URL of the package repository. R-multiverse uses "https://community.r-multiv
fields	Character string of fields to query.

Value

A data frame with one row per package and columns with package metadata. The most important columns are:

- package: character vector of package names.
- version: character vector of package versions in the repo.
- license: character vector of license names.
- remotesha: character vector of GitHub/GitLab commit hashes.
- remotes: list of character vectors with dependencies in the Remotes: field of the DESCRIPTION files.
- foss: TRUE if the package has a valid free open-source license, FALSE otherwise.
- cran: character vector of versions. Each version is the version of the package that was on CRAN during the first day of the most recent R-multiverse staging period.

See Also

Other meta: meta_checks()

Examples

```
meta_packages(repo = "https://wlandau.r-universe.dev")
```

---

propose_snapshot *Propose snapshot*

Description

Propose a Production snapshot of Staging.

Usage

```
propose_snapshot(path_staging, types = c("src", "win", "mac"))
```

Arguments

path_staging	Character string, directory path to the source files of the staging universe.
types	Character vector, what to pass to the types field in the snapshot API URL. Controls the types of binaries and documentation included in the snapshot.

Details

propose_snapshot() proposes a snapshot of Staging to migrate to Production. The recommended snapshot is the list of packages for which (1) the build and check results of the current release are in Staging, and (2) there are no issues. Writes a snapshot.url file containing an R-universe snapshot API URL to download those packages. This files is written to the directory given by the path_staging argument.

Value

NULL (invisibly). Called for its side effects. propose_snapshot() writes a snapshot.url file containing an R-universe snapshot API URL to download packages ready for Production. This file is written to the directory given by the path_staging argument.

See Also

Other staging: update_staging()

Examples

```
## Not run:
url_staging = "https://github.com/r-multiverse/staging"
path_staging <- tempfile()
gert::git_clone(url = url_staging, path = path_staging)
propose_snapshot(path_staging = path_staging)

## End(Not run)
```

`record_issues` *Record package issues.*

Description

Record R-multiverse package issues in package-specific JSON files.

Usage

```
record_issues(
  repo = "https://community.r-multiverse.org",
  versions = "versions.json",
  output = "issues.json",
  mock = NULL,
  verbose = FALSE
)
```

Arguments

`repo`	Character of length 1, URL of the package repository. R-multiverse uses "https://community.r-multiv
`versions`	Character of length 1, file path to a JSON manifest tracking the history of released versions of packages.
`output`	Character of length 1, file path to the JSON file to record new package issues. Each call to `record_issues()` overwrites the contents of the file.
`mock`	For testing purposes only, a named list of data frames for inputs to various intermediate functions.
`verbose`	`TRUE` to print progress while checking issues with dependencies, `FALSE` otherwise.

Value

`NULL` (invisibly).

Package issues

Functions like `issues_versions()` and `issues_descriptions()` perform health checks for all packages in R-multiverse. For a complete list of checks, see the `issues_*()` functions listed at https://r-multiverse.org/multiverse.internals/reference/index.html. `record_versions()` updates the version number history of releases in R-multiverse, and `record_issues()` gathers together all the issues about R-multiverse packages.

Issue data

For each package with observed problems, `record_issues()` writes a JSON list entry in the output JSON file with one element for each type of failing check. Each check-specific element has an informative name (for example, `checks`, `descriptions`, or `versions`) and a list of diagnostic information. In addition, there is a `date` field to indicate when an issue was first detected. The `date` automatically resets the next time all the issues in the package are resolved.

Examples

```
repo <- "https://wlandau.r-universe.dev"
output <- tempfile()
versions <- tempfile()
record_versions(
  versions = versions,
  repo = repo
)
record_issues(
  repo = repo,
  versions = versions,
  output = output
)
writeLines(readLines(output))
```

---

record_nonstandard_licenses

Record nonstandard licenses

Description

R-multiverse packages must have valid free and open-source (FOSS) licenses to protect the intellectual property rights of the package owners (c.f. https://en.wikipedia.org/wiki/Free_and_open-source_software). `record_nonstandard_licenses()` records packages with nonstandard licenses.

Usage

```
record_nonstandard_licenses(
  repo = "https://community.r-multiverse.org",
  path = "nonstandard_licenses.json"
)
```

Arguments

repo	Character string, URL of the repository.
path	Character string, output path to write JSON data with the names and licenses of packages with non-standard licenses.

Value

NULL (invisibly). Called for its side effects.

---

record_versions *Record the manifest of package versions.*

Description

Record the manifest of versions of packages and their hashes.

Usage

```
record_versions(
  versions = "versions.json",
  repo = "https://community.r-multiverse.org",
  current = multiverse.internals::get_current_versions(repo = repo)
)
```

Arguments

versions	Character of length 1, file path to a JSON manifest tracking the history of released versions of packages.
repo	Character of length 1, URL of the package repository. R-multiverse uses "https://community.r-multiv
current	A data frame of current versions and hashes of packages in repo. This argument is exposed for testing only.

Details

This function tracks a manifest containing the current version, the current hash, the highest version ever released, and the hash of the highest version ever released. issues_versions() uses this information to determine whether the package complies with best practices for version numbers.

Value

NULL (invisibly). Writes a package version manifest and a manifest of version issues as JSON files.

Package issues

Functions like issues_versions() and issues_descriptions() perform health checks for all packages in R-multiverse. For a complete list of checks, see the issues_*() functions listed at https://r-multiverse.org/multiverse.internals/reference/index.html. record_versions() updates the version number history of releases in R-multiverse, and record_issues() gathers together all the issues about R-multiverse packages.

Examples

```
# R-multiverse uses https://community.r-multiverse.org as the repo.
repo <- "https://wlandau.r-universe.dev" # just for testing and examples
output <- tempfile()
versions <- tempfile()
# First snapshot:
record_versions(
  versions = versions,
  repo = repo
)
readLines(versions)
# In subsequent snapshots, we have historical information about versions.
record_versions(
  versions = versions,
  repo = repo
)
readLines(versions)
```

---

review_pull_request *Review an R-multiverse contribution*

Description

Review a pull request to add packages to R-multiverse.

Usage

```
review_pull_request(
  owner = "r-multiverse",
  repo = "contributions",
  number,
  advisories = character(0L),
  organizations = character(0L)
)
```

Arguments

owner	Character of length 1, name of the package repository owner.
repo	Character of length 1, URL of the package repository. R-multiverse uses "https://community.r-multiv
number	Positive integer of length 1, index of the pull request in the repo.

advisories	Character vector of names of packages with advisories in the R Consortium Advisory Database.
organizations	Character vector of names of GitHub organizations. Pull requests from authors who are not members of at least one of these organizations will be flagged for manual review.

Value

NULL (invisibly).

See Also

Other pull request reviews: review_pull_requests()

---

review_pull_requests *Review pull requests.*

Description

Review pull requests which add packages to packages.json.

Usage

```
review_pull_requests(owner = "r-multiverse", repo = "contributions")
```

Arguments

owner	Character of length 1, name of the package repository owner.
repo	Character of length 1, URL of the package repository. R-multiverse uses "https://community.r-multiv

Value

NULL (invisibly).

See Also

Other pull request reviews: review_pull_request()

r_version_staging *Staging R version*

Description

Get the version of R from the date of the start of the most recent Staging period.

Get the version of R from the date of the start of the most recent Staging period.

Usage

```
r_version_staging()
```


```
r_version_staging()
```


Value

A data frame with one row and columns for the `POSIXct` date, patch version, and minor version of the targeted version of R.

A data frame with one row and columns for the `POSIXct` date, patch version, and minor version of the targeted version of R.

See Also

Other snapshot metadata: `date_snapshot()`, `date_staging()`

Examples

```
r_version_staging()
r_version_staging()
```

update_staging *Update staging*

Description

Update the staging universe.

Usage

```
update_staging(
  path_staging,
  path_community,
  repo_community = "https://community.r-multiverse.org",
  mock = NULL
)
```

Arguments

path_staging	Character string, directory path to the source files of the staging universe.
path_community	Character string, directory path to the source files of the community universe.
repo_community	Character string, URL of the community universe.
mock	For testing purposes only, a named list of data frames for inputs to various intermediate functions.

Details

update_staging() controls how packages enter and leave the staging universe. It updates the staging packages.json manifest depending on the contents of the community universe and issues with package checks.

Value

NULL (invisibly)

See Also

Other staging: propose_snapshot()

Examples

```
## Not run:
url_staging = "https://github.com/r-multiverse/staging"
url_community = "https://github.com/r-multiverse/community"
path_staging <- tempfile()
path_community <- tempfile()
gert::git_clone(url = url_staging, path = path_staging)
gert::git_clone(url = url_community, path = path_community)
update_staging(
  path_staging = path_staging,
  path_community = path_community,
  repo_community = "https://community.r-multiverse.org"
)

## End(Not run)
```

---

update_status *Update the package status repository*

Description

Update the repository which reports the status on individual packages.

Usage

```
update_status(
  path_status,
  path_staging,
  path_community,
  repo_staging = "https://staging.r-multiverse.org",
  repo_community = "https://community.r-multiverse.org",
  mock = NULL
)
```

Arguments

path_status	Character string, directory path to the source files of the package status repository.
path_staging	Character string, directory path to the source files of the staging universe.
path_community	Character string, directory path to the source files of the community universe.
repo_staging	Character string, URL of the staging universe.
repo_community	Character string, URL of the community universe.
mock	For testing purposes only, a named list of data frames for inputs to various intermediate functions.

See Also

Other status: interpret_status()

Examples

```
## Not run:
url_staging = "https://github.com/r-multiverse/staging"
url_community = "https://github.com/r-multiverse/community"
path_status <- tempfile()
path_staging <- tempfile()
path_community <- tempfile()
gert::git_clone(url = url_staging, path = path_staging)
gert::git_clone(url = url_community, path = path_community)
update_status(
  path_status = path_status,
  path_staging = path_staging,
  path_community = path_community,
  repo_staging = "https://staging.r-multiverse.org",
  repo_community = "https://community.r-multiverse.org"
)
writeLines(
  readLines(
    file.path(path_status, "community", "multiverse.internals.html")
  )
)
writeLines(
  readLines(
```

```
    file.path(path_status, "community", "multiverse.internals.xml")
  )
)

## End(Not run)
```

`update_topics` *Update topics*

Description

Update the list of packages for each R-multiverse topic.

Usage

```
update_topics(path, repo = "https://community.r-multiverse.org", mock = NULL)
```

Arguments

`path`	Character string, local file path to the topics repository source code.
`repo`	Character string, URL of the Community universe.
`mock`	For testing purposes only, a named list of data frames for inputs to various intermediate functions.

Value

`NULL` (invisibly). Called for its side effects.

Index