Package: multiverse.internals (via r-universe)

October 2, 2024

Title Internal Infrastructure for R-multiverse

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

Version 0.2.13

License MIT + file LICENSE

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

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

Depends R (>= 3.6)

Imports gh, igraph, jsonlite, nanonext, pkgsearch, 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

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

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

RemoteRef 0.2.13

RemoteSha 63e2b5db98e12e9c4ee1db22011d365122f22610

Contents

`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 *file issues.*

Description

Report issues with the DESCRIPTION files 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()` scans downloaded metadata from the `PACKAGES.json` file of an R universe and scans for specific issues in a package's description file:

1. The presence of a `"Remotes"` field.
2. There is a security advisory at https://github.com/RConsortium/r-advisory-database for the given package version.

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 results.

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")
```

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 metadata.

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,
  repo_staging = "https://staging.r-multiverse.org",
  types = c("src", "win", "mac"),
  r_versions = NULL,
  mock = NULL
)
```

Arguments

path_staging	Character string, directory path to the source files of the staging universe.
repo_staging	Character string, URL 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.
r_versions	Character vector of major.minor versions of R to download binaries. For example, r_versions = c("4.4", "4.3"). Set to NULL to let R-universe choose default versions.
mock	For testing purposes only, a named list of data frames for inputs to various intermediate functions.

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 snapshot.json with an R-universe-like manifest of the packages recommended for the snapshot, and a snapshot.url file containing an R-universe snapshot API URL to download those packages. Both these files are written to the directory given by the path_staging argument.

Value

NULL (invisibly). Called for its side effects. propose_snapshot() writes snapshot.json with an R-universe-like manifest of the packages recommended for the snapshot, and a snapshot.url file containing an R-universe snapshot API URL to download those packages. Both these files are written to the directory given by the path_staging argument.

See Also

Other staging: staging_is_active(), 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,
  repo_staging = "https://staging.r-multiverse.org"
)

## 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",
  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 folder to record new package issues. Each call to record_issues() overwrites the contents of the repo.
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 files

For each package with observed problems, record_issues() writes an issue file. This issue file is a JSON list with one element per type of failing check. Each element has an informative name (for example, checks, descriptions, or versions) and a list of diagnostic information.

Each issue file also has a date field. This date the day that an issue was first noticed. It automatically resets the next time all 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
)
files <- list.files(output)
print(files)
package <- head(files, n = 1)
if (length(package)) {
  print(package)
}
if (length(package)) {
  print(readLines(file.path(output, package)))
}
```

---

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 a pull request.*

Description

Review a pull request to add packages to `packages.json`.

Usage

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

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.

Value

`NULL` (invisibly).

Testing

Testing of this function unfortunately needs to be manual. Test cases:

1. Add a package correctly (automatically merge).
2. Add a bad URL (manual review).
3. Change a URL (manual review).
4. Add a file in a forbidden place (close).
5. Add a custom JSON file which can be parsed (manual review).

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).

Testing

Testing of this function unfortunately needs to be manual. Test cases:

1. Add a package correctly (automatically merge).
2. Add a bad URL (manual review).
3. Change a URL (manual review).
4. Add a file in a forbidden place (close).
5. Add a custom JSON file which can be parsed (manual review).

See Also

Other pull request reviews: `review_pull_request()`

staging_is_active *Check if the stating universe is active.*

Description

Check if the stating universe is active.

Usage

```
staging_is_active(
  start = c("01-15", "04-15", "07-15", "10-15"),
  today = Sys.Date()
)
```

Arguments

start	Character vector of "%m-%d" dates that the staging universe becomes active. Staging will then last for a full calendar month. For example, if you supply a start date of "01-15", then the staging period will include all days from "01-15" through "02-14" and not include "02-15".
today	Character string with today's date in "%Y-%m-%d" format or an object convertible to POSIXlt format.

Value

TRUE if the staging universe is active, FALSE otherwise.

See Also

Other staging: propose_snapshot(), update_staging()

Examples

```
staging_is_active()
```

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(), staging_is_active()

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)
```

Index