

# Package ‘geonode4R’

September 1, 2025

**Type** Package

**Title** Interface to 'GeoNode' REST API

**Version** 0.1-2

**Date** 2025-09-01

**Maintainer** Emmanuel Blondel <emmanuel.blondel1@gmail.com>

**Description** Provides an interface to the 'GeoNode' API, allowing to upload and publish meta-data and data in 'GeoNode'.

For more information about the 'GeoNode' API, see <<https://geonode.org/>>.

**Depends** R (>= 3.1.0)

**Imports** R6, openssl, httr, keyring, readr

**Suggests** testthat, roxygen2, covr, shiny, knitr, markdown

**License** MIT + file LICENSE

**URL** <https://github.com/eblondel/geonode4R>,  
<https://eblondel.github.io/geonode4R/>, <https://geonode.org/>

**BugReports** <https://github.com/eblondel/geonode4R/issues>

**Encoding** UTF-8

**LazyLoad** yes

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Emmanuel Blondel [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-5870-5762>>)

**Repository** CRAN

**Date/Publication** 2025-09-01 10:40:02 UTC

## Contents

|                          |   |
|--------------------------|---|
| GeoNodeManager . . . . . | 2 |
| GeoNodeUtils . . . . .   | 6 |
| GeoNodeVersion . . . . . | 7 |

**Index****10**

GeoNodeManager

*GeoNode REST API Manager***Description**

GeoNode REST API Manager

GeoNode REST API Manager

**Format**[R6Class](#) object.**Value**Object of [R6Class](#) with methods for communication with the REST API of a GeoNode instance.**Public fields**

verbose.info if geonode4R logs have to be printed

verbose.debug if curl logs have to be printed

loggerType the type of logger

url the Base url of GeoNode

**Methods****Public methods:**

- [GeoNodeManager\\$logger\(\)](#)
- [GeoNodeManager\\$INFO\(\)](#)
- [GeoNodeManager\\$WARN\(\)](#)
- [GeoNodeManager\\$ERROR\(\)](#)
- [GeoNodeManager\\$new\(\)](#)
- [GeoNodeManager\\$getUrl\(\)](#)
- [GeoNodeManager\\$connect\(\)](#)
- [GeoNodeManager\\$getExecutionStatus\(\)](#)
- [GeoNodeManager\\$getCategories\(\)](#)
- [GeoNodeManager\\$getCategory\(\)](#)
- [GeoNodeManager\\$getResourceByUUID\(\)](#)
- [GeoNodeManager\\$getResourceByAlternate\(\)](#)
- [GeoNodeManager\\$getResource\(\)](#)
- [GeoNodeManager\\$deleteResource\(\)](#)
- [GeoNodeManager\\$upload\(\)](#)
- [GeoNodeManager\\$uploadMetadata\(\)](#)

- [GeoNodeManager\\$getDataset\(\)](#)
- [GeoNodeManager\\$clone\(\)](#)

**Method** `logger()`: Prints a log message

*Usage:*

`GeoNodeManager$logger(type, text)`

*Arguments:*

`type` type of log, "INFO", "WARN", "ERROR"

`text` text

**Method** `INFO()`: Prints an INFO log message

*Usage:*

`GeoNodeManager$INFO(text)`

*Arguments:*

`text` text

**Method** `WARN()`: Prints an WARN log message

*Usage:*

`GeoNodeManager$WARN(text)`

*Arguments:*

`text` text

**Method** `ERROR()`: Prints an ERROR log message

*Usage:*

`GeoNodeManager$ERROR(text)`

*Arguments:*

`text` text

**Method** `new()`: This method is used to instantiate a `GeoNodeManager` with the url of the GeoNode and credentials to authenticate (user/pwd).

By default, the `logger` argument will be set to `NULL` (no logger). This argument accepts two possible values: `INFO`: to print only geosapi logs, `DEBUG`: to print geosapi and CURL logs.

The `keyring_backend` can be set to use a different backend for storing the GeoNode user password with **keyring** (Default value is 'env').

*Usage:*

`GeoNodeManager$new(url, user, pwd, logger = NULL, keyring_backend = "env")`

*Arguments:*

`url` url

`user` user

`pwd` pwd

`logger` logger

`keyring_backend` keyring backend. Default is 'env'

**Method** getUrl(): Get URL

*Usage:*

GeoNodeManager\$getUrl()

*Returns:* the Geoserver URL

**Method** connect(): Connects to geoServer

*Usage:*

GeoNodeManager\$connect()

*Returns:* TRUE if connected, raises an error otherwise

**Method** getExecutionStatus(): Get execution status

*Usage:*

GeoNodeManager\$getExecutionStatus(execution\_id)

*Arguments:*

execution\_id the execution id

*Returns:* the status of execution

**Method** getCategories(): Get categories

*Usage:*

GeoNodeManager\$getCategories(raw = FALSE)

*Arguments:*

raw Controls the output. Default will return an object of class [data.frame](#).

*Returns:* an object of class [list](#)

**Method** getCategory(): Get category

*Usage:*

GeoNodeManager\$getCategory(id, raw = FALSE)

*Arguments:*

id category id

raw Controls the output. Default will return an object of class [data.frame](#).

*Returns:* an object of class [list](#)

**Method** getResourceByUUID(): Get resource by UUID

*Usage:*

GeoNodeManager\$getResourceByUUID(uuid)

*Arguments:*

uuid resource uuid (or semantic id if used in place of uuid)

*Returns:* an object of class [list](#)

**Method** getResourceByAlternate(): Get resource by Alternate

*Usage:*

GeoNodeManager\$getResourceByAlternate(alternate)

*Arguments:*

alternate resource alternate

*Returns:* an object of class [list](#)

**Method** getResource(): Get resource

*Usage:*

GeoNodeManager\$getResource(id)

*Arguments:*

id resource id

*Returns:* an object of class [list](#)

**Method** deleteResource(): Deletes a resource

*Usage:*

GeoNodeManager\$deleteResource(id)

*Arguments:*

id resource (either a dataset or document) id

*Returns:* TRUE if deleted, FALSE otherwise

**Method** upload(): Uploads resource files

*Usage:*

GeoNodeManager\$upload(files)

*Arguments:*

files files

*Returns:* an object of class [list](#) giving the upload status

**Method** uploadMetadata(): Uploads ISO 19115 dataset metadata

*Usage:*

GeoNodeManager\$uploadMetadata(id, file)

*Arguments:*

id dataset id

file a metadata XML file following ISO 19115 specification

*Returns:* an object

**Method** getDataset(): Get dataset standardized metadata

*Usage:*

GeoNodeManager\$getDataset(id)

*Arguments:*

id dataset id

*Returns:* an object of class [list](#)

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

GeoNodeManager\$clone(deep = FALSE)

*Arguments:*

deep Whether to make a deep clone.

**Author(s)**

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

**Examples**

```
## Not run:
  GeoNodeManager$new("http://localhost:8080", "user", "password")

## End(Not run)
```

---

 GeoNodeUtils

*GeoNode REST API Manager Utils*


---

**Description**

GeoNode REST API Manager Utils

GeoNode REST API Manager Utils

**Format**

[R6Class](#) object.

**Value**

Object of [R6Class](#) with static util methods for communication with the REST API of a GeoNode instance.

**Static methods**

`getUserAgent()` This method is used to get the user agent for performing GeoNode API requests. Here the user agent will be compound by geonode4R package name and version.

`getUserToken(user, pwd)` This method is used to get the user authentication token for performing GeoNode API requests. Token is given a Base64 encoded string.

`GET(url, user, pwd, path, verbose)` This method performs a GET request for a given path to GeoNode REST API

`PUT(url, user, pwd, path, filename, contentType, verbose)` This method performs a PUT request for a given path to GeoNode REST API, to upload a file of name filename with given contentType

`POST(url, user, pwd, path, content, contentType, verbose)` This method performs a POST request for a given path to GeoNode REST API, to post content of given contentType

`DELETE(url, user, pwd, path, verbose)` This method performs a DELETE request for a given GeoServer resource identified by a path in GeoNode REST API

## Methods

### Public methods:

- [GeoNodeUtils\\$clone\(\)](#)

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
GeoNodeUtils$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

---

GeoNodeVersion

*A GeoNode version*

---

## Description

This class allows to grab the GeoNode version.

## Format

[R6Class](#) object.

## Details

GeoNode REST API - GeoNode Version

## Value

Object of [R6Class](#) for modelling a GeoNode version

## Public fields

`version` version

`value` value

## Methods

### Public methods:

- [GeoNodeVersion\\$new\(\)](#)
- [GeoNodeVersion\\$lowerThan\(\)](#)
- [GeoNodeVersion\\$greaterThan\(\)](#)
- [GeoNodeVersion\\$equalTo\(\)](#)
- [GeoNodeVersion\\$clone\(\)](#)

**Method** [new\(\)](#): Initializes an object of class [GeoNodeVersion](#)

*Usage:*

```
GeoNodeVersion$new(url, user, pwd)
```

*Arguments:*

url url

user user

pwd pwd

**Method** [lowerThan\(\)](#): Compares to a version and returns TRUE if it is lower, FALSE otherwise

*Usage:*

```
GeoNodeVersion$lowerThan(version)
```

*Arguments:*

version version

*Returns:* TRUE if lower, FALSE otherwise

**Method** [greaterThan\(\)](#): Compares to a version and returns TRUE if it is greater, FALSE otherwise

*Usage:*

```
GeoNodeVersion$greaterThan(version)
```

*Arguments:*

version version

*Returns:* TRUE if greater, FALSE otherwise

**Method** [equalTo\(\)](#): Compares to a version and returns TRUE if it is equal, FALSE otherwise

*Usage:*

```
GeoNodeVersion$equalTo(version)
```

*Arguments:*

version version

*Returns:* TRUE if equal, FALSE otherwise

**Method** [clone\(\)](#): The objects of this class are cloneable with this method.

*Usage:*

```
GeoNodeVersion$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

**Author(s)**

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

**Examples**

```
## Not run:
version <- GeoNodeVersion$new(
  url = "http://localhost:8080/GeoNode",
  user = "admin", pwd = "GeoNode"
)

## End(Not run)
```

# Index

- \* **GeoNode**
  - GeoNodeVersion, [7](#)
- \* **api**
  - GeoNodeManager, [2](#)
  - GeoNodeUtils, [6](#)
  - GeoNodeVersion, [7](#)
- \* **geonode**
  - GeoNodeManager, [2](#)
  - GeoNodeUtils, [6](#)
- \* **rest**
  - GeoNodeManager, [2](#)
  - GeoNodeUtils, [6](#)
  - GeoNodeVersion, [7](#)
- \* **version**
  - GeoNodeVersion, [7](#)
- data.frame, [4](#)
- GeoNodeManager, [2](#)
- GeoNodeUtils, [6](#)
- GeoNodeVersion, [7](#), [8](#)
- list, [4](#), [5](#)
- R6Class, [2](#), [6](#), [7](#)