

# Package ‘biodbNcbi’

December 18, 2024

**Title** biodbNcbi, a library for connecting to NCBI Databases.

**Version** 1.11.0

**Description** The biodbNcbi library provides access to the NCBI databases CCDS, Gene, Pubchem Comp and Pubchem Subst, using biodb package framework. It allows to retrieve entries by their accession number. Web services can be accessed for searching the database by name or mass.

**License** AGPL-3

**URL** <https://github.com/pkrog/biodbNcbi>

**BugReports** <https://github.com/pkrog/biodbNcbi/issues>

**biocViews** Software, Infrastructure, DataImport

**VignetteBuilder** knitr

**Encoding** UTF-8

**Depends** R (>= 4.1)

**Imports** biodb (>= 1.3.2), R6, XML, chk

**Suggests** roxygen2, BiocStyle, testthat (>= 2.0.0), devtools, knitr, rmarkdown, covr, lgr

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.1.2

**Collate** 'NcbiCcdsConn.R' 'NcbiCcdsEntry.R' 'NcbiEntrezConn.R' 'NcbiGeneConn.R' 'NcbiGeneEntry.R' 'NcbiPubchemConn.R' 'NcbiPubchemCompConn.R' 'NcbiPubchemEntry.R' 'NcbiPubchemCompEntry.R' 'NcbiPubchemSubstConn.R' 'NcbiPubchemSubstEntry.R' 'package.R'

**git\_url** <https://git.bioconductor.org/packages/biodbNcbi>

**git\_branch** devel

**git\_last\_commit** 4a3f8f2

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2024-12-18

**Author** Pierrick Roger [aut, cre] (ORCID:  
<https://orcid.org/0000-0001-8177-4873>)

**Maintainer** Pierrick Roger <pierrick.roger@cea.fr>

## Contents

biodbNcbi-package . . . . .	2
NcbiCcdsConn . . . . .	3
NcbiCcdsEntry . . . . .	4
NcbiEntrezConn . . . . .	5
NcbiGeneConn . . . . .	7
NcbiGeneEntry . . . . .	8
NcbiPubchemCompConn . . . . .	9
NcbiPubchemCompEntry . . . . .	10
NcbiPubchemConn . . . . .	11
NcbiPubchemEntry . . . . .	13
NcbiPubchemSubstConn . . . . .	14
NcbiPubchemSubstEntry . . . . .	15
<b>Index</b>	<b>16</b>

---

biodbNcbi-package	<i>biodbNcbi: biodbNcbi, a library for connecting to NCBI Databases.</i>
-------------------	--

---

## Description

The biodbNcbi library provides access to the NCBI databases CCDS, Gene, Pubchem Comp and Pubchem Subst, using biodb package framework. It allows to retrieve entries by their accession number. Web services can be accessed for searching the database by name or mass.

## Details

See vignette biodbNcbi:

```
vignette('biodbNcbi', package='biodbNcbi')
```

## Author(s)

**Maintainer:** Pierrick Roger <pierrick.roger@cea.fr> (ORCID)

## See Also

[NcbiGeneConn](#), [NcbiCcdsConn](#), [NcbiPubchemCompConn](#) and [NcbiPubchemSubstConn](#).

---

NcbiCcdsConn	<i>NCBI CCDS connector class.</i>
--------------	-----------------------------------

---

## Description

Connector class for NCBI CCDS database.

## Details

This is a concrete connector class. It must never be instantiated directly, but instead be instantiated through the factory [BiodbFactory](#). Only specific methods are described here. See super classes for the description of inherited methods.

## Super classes

[biodb::BiodbConnBase](#) -> [biodb::BiodbConn](#) -> NcbiCcdsConn

## Methods

### Public methods:

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

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

## See Also

[BiodbConn](#).

## Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Get a connector:
conn <- mybiodb$getFactory()$createConn('ncbi.ccds')

# Get the first entry
e <- conn$getEntry('CCDS12227.1')

# Terminate instance.
mybiodb$terminate()
```

---

NcbiCcdsEntry            *NCBI CCDS entry class.*

---

## Description

Entry class for NCBI CCDS database.

## Super classes

[biodb::BiodbEntry](#) -> [biodb::BiodbXmlEntry](#) -> [biodb::BiodbHtmlEntry](#) -> NcbiCcdsEntry

## Methods

### Public methods:

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

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

## See Also

[BiodbHtmlEntry](#).

## Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Get a connector for CCDS database:
conn <- mybiodb$getFactory()$createConn('ncbi.ccds')

# Get the first entry
e <- conn$getEntry('CCDS12227.1')

# Terminate instance.
mybiodb$terminate()
```

---

NcbiEntrezConn      *NCBI Entrez connector abstract class.*

---

## Description

NCBI Entrez connector abstract class.

NCBI Entrez connector abstract class.

## Details

This is an abstract class, mother class of all NCBI Entrez connector classes.

Note: the implementation of the `getEntryIds()` method uses a last resort solution. It returns only a small subset of Ncbi entries.

## Super classes

`biodb::BiodbConnBase` -> `biodb::BiodbConn` -> `NcbiEntrezConn`

## Methods

### Public methods:

- `NcbiEntrezConn$new()`
- `NcbiEntrezConn$wsEfetch()`
- `NcbiEntrezConn$wsEsearch()`
- `NcbiEntrezConn$wsEinfo()`
- `NcbiEntrezConn$clone()`

**Method** `new()`: New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the `createConn()` method of the factory class.

#### Usage:

```
NcbiEntrezConn$new(entrez.name, entrez.tag = NULL, entrez.id.tag = NULL, ...)
```

#### Arguments:

`entrez.name` Entrez database name (ex: "gene").

`entrez.tag` Entrez database tag (ex: "Entrezgene").

`entrez.id.tag` Entrez database ID tag (ex: "Gene-track\_geneid").

... All other parameters are passed to the super class initializer.

**Returns:** Nothing.

**Method** `wsEfetch()`: Calls Entrez efetch web service. See <https://www.ncbi.nlm.nih.gov/books/NBK25499/#chapter4.EF>

#### Usage:

```
NcbiEntrezConn$wsEfetch(
  id,
  rettype = NULL,
  retmode = NULL,
  retfmt = c("plain", "parsed", "request")
)
```

*Arguments:*

*id* A character vector of entry IDs.

*rettype* The retrieval type. See NCBI documentation.

*retmode* The retrieval mode. See NCBI documentation.

*retfmt* Use to set the format of the returned value. 'plain' will return the raw results from the server, as a character value. 'parsed' will return the parsed results, as an XML object. 'request' will return a BiodbRequest object representing the request as it would have been sent.

*Returns:* Depending on *retfmt* parameter.

**Method** `wsEsearch()`: Calls Entrez esearch web service. See <https://www.ncbi.nlm.nih.gov/books/NBK25499/#chapter4>.

*Usage:*

```
NcbiEntrezConn$wsEsearch(
  term,
  field = NULL,
  retmax = NULL,
  retfmt = c("plain", "parsed", "request", "ids")
)
```

*Arguments:*

*term* Text query. See NCBI documentation.

*field* Entrez field to which to limit the search. See NCBI documentation.

*retmax* Maximum number of entry IDs to return.

*retfmt* Use to set the format of the returned value. 'plain' will return the raw results from the server, as a character value. 'parsed' will return the parsed results, as an XML object. 'request' will return a BiodbRequest object representing the request as it would have been sent. 'ids' will return a character vector containing the IDs of the matching entries.

*Returns:* Depending on *retfmt* parameter.

**Method** `wsEinfo()`: Calls Entrez einfo web service, returning information about this database. See <https://www.ncbi.nlm.nih.gov/books/NBK25499/#chapter4.EInfo>.

*Usage:*

```
NcbiEntrezConn$wsEinfo(retfmt = c("plain", "request", "parsed"))
```

*Arguments:*

*retfmt* Use to set the format of the returned value. 'plain' will return the raw results from the server, as a character value. 'parsed' will return the parsed results, as an XML object. 'request' will return a BiodbRequest object representing the request as it would have been sent.

*Returns:* Depending on *retfmt* parameter.

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

### See Also

[BiodbConn](#).

### Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('ncbi.gene')

# Get an entry
e <- conn$getEntry('2833')

# Terminate instance.
mybiodb$terminate()
```

---

NcbiGeneConn

*NCBI Gene connector class.*

---

### Description

NCBI Gene connector class.

NCBI Gene connector class.

### Details

This is the connector class for a NCBI Gene database.

### Super classes

[biodb::BiodbConnBase](#) -> [biodb::BiodbConn](#) -> [biodbNcbi::NcbiEntrezConn](#) -> NcbiGeneConn

### Methods

#### Public methods:

- [NcbiGeneConn\\$new\(\)](#)
- [NcbiGeneConn\\$clone\(\)](#)

**Method** `new()`: New instance initializer.

*Usage:*

```
NcbiGeneConn$new(...)
```

*Arguments:*

... All other parameters are passed to the super class initializer.

*Returns:* Nothing.

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

**See Also**

[NcbiEntrezConn](#).

**Examples**

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('ncbi.gene')

# Get an entry
e <- conn$getEntry('2833')

# Terminate instance.
mybiodb$terminate()
```

---

NcbiGeneEntry

*NCBI Gene entry class.*

---

**Description**

This is the entry class for a NCBI Gene database.

**Super classes**

```
biodb::BiodbEntry -> biodb::BiodbXmlEntry -> NcbiGeneEntry
```



## Methods

### Public methods:

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

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('ncbi.gene')

# Get an entry
e <- conn$getEntry('2833')

# Terminate instance.
mybiodb$terminate()
```

---

NcbiPubchemCompConn    *NCBI PubChem Compound connector class.*

---

## Description

NCBI PubChem Compound connector class.

NCBI PubChem Compound connector class.

## Details

This is the connector class for a NCBI PubChen Compound database.

## Super classes

```
biodb::BiodbConnBase -> biodb::BiodbConn -> biodbNcbi::NcbiEntrezConn -> biodbNcbi::NcbiPubchemConn
-> NcbiPubchemCompConn
```

## Methods

### Public methods:

- [NcbiPubchemCompConn\\$new\(\)](#)
- [NcbiPubchemCompConn\\$clone\(\)](#)

**Method** `new()`: New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the `createConn()` method of the factory class.

*Usage:*

```
NcbiPubchemCompConn$new(...)
```

*Arguments:*

... All other parameters are passed to the super class initializer.

*Returns:* Nothing.

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('ncbi.pubchem.comp')

# Get an entry
e <- conn$getEntry('2')

# Terminate instance.
mybiodb$terminate()
```

---

NcbiPubchemCompEntry *NCBI PubChem Compound entry class.*

---

## Description

This is the entry class for a NCBI PubChen Compound database.

## Super classes

```
biodb::BiodbEntry -> biodb::BiodbXmlEntry -> biodbNcbi::NcbiPubchemEntry -> NcbiPubchemCompEntry
```

## Methods

### Public methods:

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

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('ncbi.pubchem.comp')

# Get an entry
e <- conn$getEntry('2')

# Terminate instance.
mybiodb$terminate()
```

---

NcbiPubchemConn

*NCBI PubChem connector abstractclass.*

---

## Description

NCBI PubChem connector abstractclass.

NCBI PubChem connector abstractclass.

## Details

This is an abstract class, mother class of all NCBI PubChem connector classes.

## Super classes

[biodb::BiodbConnBase](#) -> [biodb::BiodbConn](#) -> [biodbNcbi::NcbiEntrezConn](#) -> NcbiPubchemConn

## Methods

### Public methods:

- [NcbiPubchemConn\\$new\(\)](#)
- [NcbiPubchemConn\\$clone\(\)](#)

**Method new():** New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the createConn() method of the factory class.

*Usage:*

```
NcbiPubchemConn$new(db.name, id.xmltag, entry.xmltag, id.urlfield, ...)
```

*Arguments:*

db.name PubChem database name.

id.xmltag XML tag for ID.

entry.xmltag XML tag for entry.

id.urlfield Database ID to use when building URL.

... All other parameters are passed to the super class initializer.

*Returns:* Nothing.

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

*Usage:*

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

*Arguments:*

deep Whether to make a deep clone.

## See Also

[NcbiEntrezConn](#).

## Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('ncbi.pubchem.comp')

# Get an entry
e <- conn$getEntry('2')

# Terminate instance.
mybiodb$terminate()
```

---

NcbiPubchemEntry      *NCBI PubChem entry class.*

---

### Description

NCBI PubChem entry class.

NCBI PubChem entry class.

### Details

This the abstract entry class for all NCBI PubChem entry classes.

### Super classes

`biodb::BiodbEntry` -> `biodb::BiodbXmlEntry` -> `NcbiPubchemEntry`

### Methods

#### Public methods:

- `NcbiPubchemEntry$new()`
- `NcbiPubchemEntry$clone()`

**Method** `new()`: New instance initializer.

*Usage:*

```
NcbiPubchemEntry$new(...)
```

*Arguments:*

... All other parameters are passed to the super class initializer.

*Returns:* Nothing.

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

*Usage:*

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

*Arguments:*

deep Whether to make a deep clone.

### Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('ncbi.pubchem.comp')

# Get an entry
e <- conn$getEntry('2')
```

```
# Terminate instance.  
mybiodb$terminate()
```

---

NcbiPubchemSubstConn *NCBI PubChem Substance connector class.*

---

## Description

NCBI PubChem Substance connector class.

NCBI PubChem Substance connector class.

## Details

This is the connector class for a NCBI PubChen Substance database.

## Super classes

```
biodb::BiodbConnBase -> biodb::BiodbConn -> biodbNcbi::NcbiEntrezConn -> biodbNcbi::NcbiPubchemConn  
-> NcbiPubchemSubstConn
```

## Methods

### Public methods:

- [NcbiPubchemSubstConn\\$new\(\)](#)
- [NcbiPubchemSubstConn\\$clone\(\)](#)

**Method** `new()`: New instance initializer. Connector classes must not be instantiated directly. Instead, you must use the `createConn()` method of the factory class.

*Usage:*

```
NcbiPubchemSubstConn$new(...)
```

*Arguments:*

... All other parameters are passed to the super class initializer.

*Returns:* Nothing.

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('ncbi.pubchem.subst')

# Get an entry
e <- conn$getEntry('2')

# Terminate instance.
mybiodb$terminate()
```

---

NcbiPubchemSubstEntry *NCBI PubChem Substance entry class.*

---

## Description

This is the entry class for a NCBI PubChen Substance database.

## Super classes

```
biodb::BiodbEntry -> biodb::BiodbXmlEntry -> biodbNcbi::NcbiPubchemEntry -> NcbiPubchemSubstEntry
```

## Methods

### Public methods:

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

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
# Create an instance with default settings:
mybiodb <- biodb::newInst()

# Create a connector
conn <- mybiodb$getFactory()$createConn('ncbi.pubchem.subst')

# Get an entry
e <- conn$getEntry('2')

# Terminate instance.
mybiodb$terminate()
```

# Index

`biodb::BiodbConn`, [3](#), [5](#), [7](#), [9](#), [11](#), [14](#)  
`biodb::BiodbConnBase`, [3](#), [5](#), [7](#), [9](#), [11](#), [14](#)  
`biodb::BiodbEntry`, [4](#), [8](#), [10](#), [13](#), [15](#)  
`biodb::BiodbHtmlEntry`, [4](#)  
`biodb::BiodbXmlEntry`, [4](#), [8](#), [10](#), [13](#), [15](#)  
`BiodbConn`, [3](#), [7](#)  
`BiodbFactory`, [3](#)  
`BiodbHtmlEntry`, [4](#)  
`biodbNcbi` (`biodbNcbi`-package), [2](#)  
`biodbNcbi`-package, [2](#)  
`biodbNcbi::NcbiEntrezConn`, [7](#), [9](#), [11](#), [14](#)  
`biodbNcbi::NcbiPubchemConn`, [9](#), [14](#)  
`biodbNcbi::NcbiPubchemEntry`, [10](#), [15](#)

`NcbiCcdsConn`, [2](#), [3](#)  
`NcbiCcdsEntry`, [4](#)  
`NcbiEntrezConn`, [5](#), [8](#), [12](#)  
`NcbiGeneConn`, [2](#), [7](#)  
`NcbiGeneEntry`, [8](#)  
`NcbiPubchemCompConn`, [2](#), [9](#)  
`NcbiPubchemCompEntry`, [10](#)  
`NcbiPubchemConn`, [11](#)  
`NcbiPubchemEntry`, [13](#)  
`NcbiPubchemSubstConn`, [2](#), [14](#)  
`NcbiPubchemSubstEntry`, [15](#)