

# Package ‘biodbLipidmaps’

October 15, 2023

**Title** biodbLipidmaps, a library for connecting to the Lipidmaps Structure database

**Version** 1.6.0

**Description** The biodbLipidmaps library provides access to the Lipidmaps Structure Database, using biodb package framework. It allows to retrieve entries by their accession number, and run web the services lmsdSearch and lmsdRecord.

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

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

**biocViews** Software, Infrastructure, DataImport

**License** AGPL-3

**Encoding** UTF-8

**VignetteBuilder** knitr

**Depends** R (>= 4.1)

**Imports** biodb (>= 1.3.2), lifecycle, R6

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

**RoxygenNote** 7.2.2

**Collate** 'LipidmapsStructureConn.R' 'LipidmapsStructureEntry.R' 'package.R'

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

**git\_branch** RELEASE\_3\_17

**git\_last\_commit** 3555696

**git\_last\_commit\_date** 2023-04-25

**Date/Publication** 2023-10-15

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

**Maintainer** Pierrick Roger <[pierrick.roger@cea.fr](mailto:pierrick.roger@cea.fr)>

**R topics documented:**

biodbLipidmaps-package . . . . .	2
LipidmapsStructureConn . . . . .	2
LipidmapsStructureEntry . . . . .	5

<b>Index</b>	<b>7</b>
--------------	----------

---

biodbLipidmaps-package

*biodbLipidmaps: biodbLipidmaps, a library for connecting to the Lipidmaps Structure database*

---

**Description**

The biodbLipidmaps library provides access to the Lipidmaps Structure Database, using biodb package framework. It allows to retrieve entries by their accession number, and run web the services lmsdSearch and lmsdRecord.

**Details**

See vignette intro: ““ vignette('intro', package='biodbLipidmaps') ““

**Author(s)**

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

**See Also**

[LipidmapsStructureConn](#).

---

LipidmapsStructureConn

*Lipidmaps Structure connector class.*

---

**Description**

Lipidmaps Structure connector class.

Lipidmaps Structure connector class.

**Details**

Connector class for Lipidmaps Structure.

**Super classes**

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

## Methods

### Public methods:

- [LipidmapsStructureConn\\$wsLmsdSearch\(\)](#)
- [LipidmapsStructureConn\\$wsLmsd\(\)](#)
- [LipidmapsStructureConn\\$wsLmsdRecord\(\)](#)
- [LipidmapsStructureConn\\$clone\(\)](#)

**Method** `wsLmsdSearch()`: Calls LMSDSearch web service. See <https://www.lipidmaps.org/data/structure/programmatically> for details.

#### Usage:

```
LipidmapsStructureConn$wsLmsdSearch(
  mode = NULL,
  output.mode = NULL,
  output.type = NULL,
  output.delimiter = NULL,
  output.quote = NULL,
  output.column.header = NULL,
  lmid = NULL,
  name = NULL,
  formula = NULL,
  search.type = NULL,
  smiles.string = NULL,
  exact.mass = NULL,
  exact.mass.offset = NULL,
  core.class = NULL,
  main.class = NULL,
  sub.class = NULL,
  retfmt = c("plain", "request", "parsed", "ids")
)
```

#### Arguments:

`mode` The search mode: 'ProcessStrSearch', 'ProcessTextSearch' or 'ProcessTextOntology-Search'. Compulsory.

`output.mode` If set to 'File', will output a in format 'output.type', otherwise will output HTML.

`output.type` The output format: 'TSV', 'CSV' or 'SDF'.

`output.delimiter` The delimiter for TSV or CSV formats: 'Tab', 'Comma', 'Semicolon'.

`output.quote` If quotes are to be used: 'Yes' or 'No'.

`output.column.header` If header must be output: 'Yes' or 'No'.

`lmid` a Lipidmaps ID.

`name` The name to search for.

`formula` The chemical formula to search for.

`search.type` The search type: 'SubStructure' or 'ExactMatch'.

`smiles.string` A SMILES to search for.

`exact.mass` The mass to search for.

`exact.mass.offset` The tolerance on the mass search.

`core.class` An integer number from 1 to 8.  
`main.class` An integer number. See Lipidmaps documentation.  
`sub.class` An integer number. See Lipidmaps documentation.  
`retfmt` Use to set the format of the returned value. 'plain' will return the raw results from the server, as a character value. 'request' will return the request that would have been sent, as a `BiodbRequest` object. 'parsed' will return data frame. 'ids' will return a character vector containing the IDs of the matching entries.

*Returns:* Depending on 'retfmt'.

**Method** `wsLmsd()`: Calls LMSD web service for downloading one entry.

*Usage:*

```
LipidmapsStructureConn$wsLmsd(
  lmid,
  format = c("tsv", "csv"),
  retfmt = c("plain", "request", "parsed")
)
```

*Arguments:*

`lmid` The accession number of the entry to retrieve.  
`format` The output format (either 'tsv' or 'csv').  
`retfmt` Use to set the format of the returned value. 'plain' will return the raw results from the server, as a character value. 'request' will return the request that would have been sent, as a `BiodbRequest` object. 'parsed' will return data frame.

*Returns:* Depending on 'retfmt'.

**Method** `wsLmsdRecord()`: Calls LMSDRecord web service. See <http://www.lipidmaps.org/data/structure/programmatica>

*Usage:*

```
LipidmapsStructureConn$wsLmsdRecord(
  lmid,
  mode = NULL,
  output.type = NULL,
  output.delimiter = NULL,
  output.quote = NULL,
  output.column.header = NULL,
  retfmt = c("plain", "request", "parsed")
)
```

*Arguments:*

`lmid` A character vector containing the IDs of the wanted entries.  
`mode` If set to 'File', will output a in format 'output.type', otherwise will output HTML.  
`output.type` The output format: 'TSV', 'CSV' or 'SDF'.  
`output.delimiter` The delimiter for TSV or CSV formats: 'Tab', 'Comma', 'Semicolon'.  
`output.quote` If quotes are to be used: 'Yes' or 'No'.  
`output.column.header` If header must be output: 'Yes' or 'No'.  
`retfmt` Use to set the format of the returned value. 'plain' will return the raw results from the server, as a character value. 'request' will return the request that would have been sent, as a `BiodbRequest` object. 'parsed' will return data frame.

*Returns:* Depending on 'retfmt'.

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

*Usage:*

```
LipidmapsStructureConn$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('lipidmaps.structure')

# Get an entry
e <- conn$getEntry('LMFA00000001')

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

---

LipidmapsStructureEntry

*Lipidmaps Structure entry class.*

---

### Description

Lipidmaps Structure entry class.

Lipidmaps Structure entry class.

### Details

Entry class for Lipidmaps Structure.

### Super classes

[biodb::BiodbEntry](#) -> [biodb::BiodbCsvEntry](#) -> LipidmapsStructureEntry

### Methods

#### Public methods:

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

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

*Usage:*

```
LipidmapsStructureEntry$new(...)
```

*Arguments:*

... All parameters are passed to super class' initializer.

*Returns:* Nothing.

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

*Usage:*

```
LipidmapsStructureEntry$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('lipidmaps.structure')

# Get an entry
e <- conn$getEntry('LMFA00000001')

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

# Index

`biodb::BiodbConn`, [2](#)  
`biodb::BiodbConnBase`, [2](#)  
`biodb::BiodbCsvEntry`, [5](#)  
`biodb::BiodbEntry`, [5](#)  
`biodbLipidmaps`  
    (`biodbLipidmaps-package`), [2](#)  
`biodbLipidmaps-package`, [2](#)  
  
`LipidmapsStructureConn`, [2](#), [2](#)  
`LipidmapsStructureEntry`, [5](#)