

# Package ‘biodbHmdb’

September 8, 2024

**Title** biodbHmdb, a library for connecting to the HMDB Database

**Version** 1.10.1

**Description** The biodbHmdb library is an extension of the biodb framework package that provides access to the HMDB Metabolites database. It allows to download the whole HMDB Metabolites database locally, access entries and search for entries by name or description. A future version of this package will also include a search by mass and mass spectra annotation.

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

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

**biocViews** Software, Infrastructure, DataImport

**Depends** R (>= 4.1)

**License** AGPL-3

**Encoding** UTF-8

**VignetteBuilder** knitr

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

**Imports** R6, biodb (>= 1.3.2), Rcpp, zip

**LinkingTo** Rcpp, testthat

**NeedsCompilation** yes

**RoxygenNote** 7.3.2

**Collate** 'HmdbMetabolitesConn.R' 'HmdbMetabolitesEntry.R' fcts.R  
RcppExports.R 'package.R' 'catch-routine-registration.R'

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

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** 49eb501

**git\_last\_commit\_date** 2024-08-05

**Repository** Bioconductor 3.19

**Date/Publication** 2024-09-08

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

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

## Contents

HmdbMetabolitesConn . . . . .	2
HmdbMetabolitesEntry . . . . .	3
hmdbToSqlite . . . . .	4
<b>Index</b>	<b>5</b>

---

HmdbMetabolitesConn    *The connector class for the HMDB Metabolites database.*

---

### Description

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](#) -> [HmdbMetabolitesConn](#)

### Methods

#### Public methods:

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

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

*Usage:*

```
HmdbMetabolitesConn$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('hmdb.metabolites')

# Get an entry
# Getting one entry requires the download of the whole database.
e <- conn$getEntry('HMDB0000001')

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

---

HmdbMetabolitesEntry *HMDB Metabolites entry class.*

---

### Description

HMDB Metabolites entry class.

HMDB Metabolites entry class.

### Details

This is the entry class for the HMDB Metabolites database.

### Super classes

`biodb::BiodbEntry` -> `biodb::BiodbXmlEntry` -> `HmdbMetabolitesEntry`

### Methods

#### Public methods:

- `HmdbMetabolitesEntry$new()`
- `HmdbMetabolitesEntry$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:*

```
HmdbMetabolitesEntry$new(...)
```

*Arguments:*

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

*Returns:* Nothing.

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

*Usage:*

```
HmdbMetabolitesEntry$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('hmdb.metabolites')

# Get an entry
```

```
# Getting one entry requires the download of the whole database.
e <- conn$getEntry('HMDB0000001')

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

---

hmdbToSqlite

*Convert HMDB Metabolites to a SQLite database.*

---

### **Description**

Create an SQLite compound database

### **Usage**

```
hmdbToSqlite(sqlite.file = "hmdb.sqlite")
```

### **Arguments**

sqlite.file      Path to the SQLite file to create.

### **Value**

Nothing.

### **Examples**

```
# Copy all HMDB entries into a new SQLite database:
biodbHmdb::hmdbToSqlite('myfile.sqlite')
```

# Index

[biodb::BiodbConn](#), [2](#)  
[biodb::BiodbConnBase](#), [2](#)  
[biodb::BiodbEntry](#), [3](#)  
[biodb::BiodbXmlEntry](#), [3](#)  
[BiodbFactory](#), [2](#)

[HmdbMetabolitesConn](#), [2](#)  
[HmdbMetabolitesEntry](#), [3](#)  
[hmdbToSqlite](#), [4](#)