Package 'BED'

May 22, 2024

Type Package

Title Biological Entity Dictionary (BED)

Version 1.5.2

Description An interface for the 'Neo4j' database providing mapping between different identifiers of biological entities. This Biological Entity Dictionary (BED) has been developed to address three main challenges. The first one is related to the completeness of identifier mappings. Indeed, direct mapping information provided by the different systems are not always complete and can be enriched by mappings provided by other resources. More interestingly, direct mappings not identified by any of these resources can be indirectly inferred by using mappings to a third reference. For example, many human Ensembl gene ID are not directly mapped to any Entrez gene ID but such mappings can be inferred using respective mappings to HGNC ID. The second challenge is related to the mapping of deprecated identifiers. Indeed, entity identifiers can change from one resource release to another. The identifier history is provided by some resources, such as Ensembl or the NCBI, but it is generally not used by mapping tools. The third challenge is related to the automation of the mapping process according to the relationships between the biological entities of interest. Indeed, mapping between gene and protein ID scopes should not be done the same way than between two scopes regarding gene ID. Also, converting identifiers from different organisms should be possible using gene orthologs information. The method has been published by Godard and van Eyll (2018) <doi:10.12688/f1000research.13925.3>.

URL `https://patzaw.github.io/BED/`, `https://github.com/patzaw/BED`

BugReports `https://github.com/patzaw/BED/issues`

License GPL-3

Depends R (>= 3.6), neo2R (>= 2.4.1), visNetwork

Imports dplyr, readr, stringr, utils, shiny (>= 0.13), DT, miniUI (>= 0.1.1), rstudioapi (>= 0.5)

Suggests knitr, rmarkdown, biomaRt, GEOquery, base64enc, htmltools, webshot2, RCurl

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.3.1

NeedsCompilation no

Author Patrice Godard [aut, cre, cph] (<https://orcid.org/0000-0001-6257-9730>)

Maintainer Patrice Godard <`patrice.godard@gmail.com`>

Repository CRAN

Date/Publication 2024-05-22 10:10:02 UTC

R topics documented:

`BED` *Biological Entity Dictionary (BED)*

Description

An interface for the neo4j database providing mapping between different identifiers of biological entities. This Biological Entity Dictionary (BED) has been developed to address three main challenges. The first one is related to the completeness of identifier mappings. Indeed, direct mapping information provided by the different systems are not always complete and can be enriched by mappings provided by other resources. More interestingly, direct mappings not identified by any of these resources can be indirectly inferred by using mappings to a third reference. For example, many human Ensembl gene ID are not directly mapped to any Entrez gene ID but such mappings can be inferred using respective mappings to HGNC ID. The second challenge is related to the mapping of deprecated identifiers. Indeed, entity identifiers can change from one resource release to another. The identifier history is provided by some resources, such as Ensembl or the NCBI, but it is generally not used by mapping tools. The third challenge is related to the automation of the mapping process according to the relationships between the biological entities of interest. Indeed, mapping between gene and protein ID scopes should not be done the same way than between two scopes regarding gene ID. Also, converting identifiers from different organisms should be possible using gene orthologs information.

- Vignette
- Available database instance: `https://github.com/patzaw/BED#bed-database-instance-available-as-a-docke`
- Building a database instance: `https://github.com/patzaw/BED#build-a-bed-database-instance`
- Repository: `https://github.com/patzaw/BED`
- Bug reports: `https://github.com/patzaw/BED/issues`

Author(s)

Patrice Godard

`bedCall` *Call a function on the BED graph*

Description

Call a function on the BED graph

Usage

```
bedCall(f, ..., bedCheck = FALSE)
```

Arguments

`f`	the function to call
`...`	params for f
`bedCheck`	check if a connection to BED exists (default: FALSE).

Value

The output of the called function.

See Also

checkBedConn

Examples

```
## Not run:
result <- bedCall(
   cypher,
   query=prepCql(
      'MATCH (n:BEID)',
      'WHERE n.value IN $values',
      'RETURN n.value AS value, n.labels, n.database'
   ),
   parameters=list(values=c("10", "100"))
)

## End(Not run)
```

`bedImport` *Feeding BED: Imports a data.frame in the BED graph database*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
bedImport(cql, toImport, periodicCommit = 10000, ...)
```

Arguments

`cql`	the CQL query to be applied on each row of toImport
`toImport`	the data.frame to be imported as "row". Use "row.FIELD" in the cql query to refer to one FIELD of the toImport data.frame
`periodicCommit`	use periodic commit when loading the data (default: 1000).
`...`	additional parameters for bedCall

Value

the results of the query

See Also

bedCall, neo2R::import_from_df

`BEIDList` *Create a BEIDList*

Description

Create a BEIDList

Usage

```
BEIDList(l, metadata, scope)
```

Arguments

`l`	a named list of BEID vectors
`metadata`	a data.frame with rownames or a column "**.lname**" all in names of l. If missing, the metadata is constructed with **.lname** being the names of l.
`scope`	a list with 3 character vectors of length one named "be", "source" and "organism". If missing, it is guessed from l.

Value

A BEIDList object which is a list of BEID vectors with 2 additional attributes:

- **metadata**: a data.frame with metadata about list elements. The "**.lname**" column correspond to the names of the BEIDList.
- **scope**: the BEID scope ("be", "source" and "organism")

Examples

```
## Not run:
bel <- BEIDList(
   l=list(
      kinases=c("117283", "3706", "3707", "51447", "80271", "9807"),
      phosphatases=c(
         "130367", "249", "283871", "493911", "57026", "5723", "81537"
      )
   ),
   scope=list(be="Gene", source="EntrezGene", organism="Homo sapiens")
)
scope(bel)
metadata(bel)
metadata(bel) <- dplyr::mutate(
   metadata(bel),
   "description"=c("A few kinases", "A few phosphatases")
)
metadata(bel)

## End(Not run)
```

`BEIDs` *Get the BEIDs from an object*

Description

Get the BEIDs from an object

Usage

```
BEIDs(x, ...)
```

Arguments

`x`	an object representing a collection of BEID (e.g. BEIDList)
`...`	method specific parameters

Value

A tibble with at least 4 columns:

- value
- be
- source
- organism
- ...

`beidsServer` *Shiny module for searching BEIDs*

Description

Shiny module for searching BEIDs

Usage

```
beidsServer(
  id,
  toGene = TRUE,
  multiple = FALSE,
  beOfInt = NULL,
  selectBe = TRUE,
  orgOfInt = NULL,
  selectOrg = TRUE,
  tableHeight = 150
)

beidsUI(id)
```

Arguments

`id`	an identifier for the module instance
`toGene`	focus on gene entities (default=TRUE): matches from other BE are converted to genes.
`multiple`	allow multiple selections (default=FALSE)
`beOfInt`	if toGene==FALSE, BE to consider (default=NULL ==> all)
`selectBe`	if toGene==FALSE, display an interface for selecting BE
`orgOfInt`	organism to consider (default=NULL ==> all)
`selectOrg`	display an interface for selecting organisms
`tableHeight`	height of the result table (default: 150)

Value

A reactive data.frame with the following columns:

- **beid**: the BE identifier
- **preferred**: preferred identifier for the same BE in the same scope
- **be**: the type of biological entity
- **source**: the source of the identifier
- **organism**: the BE organism
- **entity**: internal identifier of the BE
- **match**: the matching character string

Functions

- `beidsUI()`:

Examples

```
## Not run:
library(shiny)
library(BED)
library(DT)

ui <- fluidPage(
   beidsUI("be"),
   fluidRow(
      column(
         12,
         tags$br(),
         h3("Selected gene entities"),
         DTOutput("result")
      )
   )
)

server <- function(input, output){
   found <- beidsServer("be", toGene=TRUE, multiple=TRUE, tableHeight=250)
   output$result <- renderDT({
      req(found())
      toRet <- found()
      datatable(toRet, rownames=FALSE)
   })
}

shinyApp(ui = ui, server = server)

## End(Not run)
```

`beIDsToAllScopes` *Find all BEID and ProbeID corresponding to a BE*

Description

Find all BEID and ProbeID corresponding to a BE

Usage

```
beIDsToAllScopes(
  beids,
  be,
  source,
  organism,
  entities = NULL,
  canonical_symbols = TRUE
)
```

Arguments

`beids`	a character vector of gene identifiers
`be`	one BE. **Guessed if not provided**
`source`	the source of gene identifiers. **Guessed if not provided**
`organism`	the gene organism. **Guessed if not provided**
`entities`	a numeric vector of gene entity. If NULL (default), beids, source and organism arguments are used to identify BEs. Be carefull when using entities as these identifiers are not stable.
`canonical_symbols`	return only canonical symbols (default: TRUE).

Value

A data.frame with the following fields:

- **value**: the identifier
- **be**: the type of BE
- **source**: the source of the identifier
- **organism**: the BE organism
- **symbol**: canonical symbol of the identifier
- **BE_entity**: the BE entity input
- **BEID** (optional): the BE ID input
- **BE_source** (optional): the BE source input

`cacheBedCall` *Cached neo4j call*

Description

This function calls neo4j DB the first time a query is sent and puts the result in the cache SQLite database. The next time the same query is called, it loads the results directly from cache SQLite database.

Usage

```
cacheBedCall(..., tn, recache = FALSE)
```

Arguments

`...`	params for bedCall
`tn`	the name of the cached table
`recache`	boolean indicating if the CQL query should be run even if the table is already in cache

Details

Use only with "row" result returned by DB request.

Internal use.

Value

The results of the bedCall.

See Also

cacheBedResult, bedCall

`cacheBedResult` *Put a BED query result in cache*

Description

Internal use

Usage

```
cacheBedResult(value, name)
```

Arguments

`value`	the result to cache
`name`	the name of the query

See Also

cacheBedCall, loadBedResult

`checkBedCache` *Check BED cache*

Description

This function checks information recorded into BED cache and resets it if not relevant.

Usage

```
checkBedCache(newCon = FALSE)
```

Arguments

`newCon`	if TRUE for the loading of the system information file

Details

Internal use.

See Also

clearBedCache, lsBedCache

`checkBedConn` *Check if there is a connection to a BED database*

Description

Check if there is a connection to a BED database

Usage

```
checkBedConn(verbose = FALSE)
```

Arguments

`verbose`	if TRUE print information about the BED connection (default: FALSE).

Value

- TRUE if the connection can be established
- Or FALSE if the connection cannot be established or the "System" node does not exist or does not have "BED" as name or any version recorded.

See Also

connectToBed

`checkBeIds` *Check biological entities (BE) identifiers*

Description

This function takes a vector of identifiers and verify if they can be found in the provided source database according to the BE type and the organism of interest. If an ID is in the DB but not linked directly nor indirectly to any entity then it is considered as not found.

Usage

```
checkBeIds(ids, be, source, organism, stopThr = 1, caseSensitive = FALSE)
```

Arguments

`ids`	a vector of identifiers to be checked
`be`	biological entity. See getBeIds. **Guessed if not provided**
`source`	source of the ids. See getBeIds. **Guessed if not provided**
`organism`	the organism of interest. See getBeIds. **Guessed if not provided**
`stopThr`	proportion of non-recognized IDs above which an error is thrown. Default: 1 ==> no check
`caseSensitive`	if FALSE (default) the case is not taken into account when checking ids.

Value

invisible(TRUE). Stop if too many (see stopThr parameter) ids are not found. Warning if any id is not found.

See Also

getBeIds, listBeIdSources, getAllBeIdSources

Examples

```
## Not run:
checkBeIds(
   ids=c("10", "100"), be="Gene", source="EntrezGene", organism="human"
)
checkBeIds(
   ids=c("10", "100"), be="Gene", source="Ens_gene", organism="human"
)

## End(Not run)
```

`cleanDubiousXRef` *Identify and remove dubious cross-references*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
cleanDubiousXRef(d, strict = TRUE)
```

Arguments

`d`	a cross-reference data.frame with 2 columns.
`strict`	if TRUE (default), the function returns only unambiguous mappings

Value

This function returns d without dubious cross-references. Issues are reported in attr(d, "issues").

`clearBedCache` *Clear the BED cache SQLite database*

Description

Clear the BED cache SQLite database

Usage

```
clearBedCache(queries = NULL, force = FALSE, hard = FALSE, verbose = FALSE)
```

Arguments

`queries`	a character vector of the names of queries to remove. If NULL all queries are removed.
`force`	if TRUE clear the BED cache table even if cache file is not found
`hard`	if TRUE remove everything in cache without checking file names
`verbose`	display some information during the process

See Also

lsBedCache

`compareBedInstances` *Compare 2 BED database instances*

Description

Compare 2 BED database instances

Usage

```
compareBedInstances(connections)
```

Arguments

`connections`	a numeric vector of length 1 or 2 providing connections from lsBedConnections to be compared.

Details

The current connection is restored when exiting this function.

Value

If only one connection is provided, the function returns a list with information about BEID and platforms available for the connection along with DB version information. If two connections are provided the same information as above is provided for the 2 connection named V1 and V2 in that order. In addition, differences observed between the 2 instances are reported for BEID and platforms.

`connectToBed` *Connect to a neo4j BED database*

Description

Connect to a neo4j BED database

Usage

```
connectToBed(
  url = NULL,
  username = NULL,
  password = NULL,
  connection = 1,
  remember = FALSE,
  useCache = NA,
  importPath = NULL,
  .opts = list()
)
```

Arguments

`url`	a character string. The host and the port are sufficient (e.g: "localhost:5454")
`username`	a character string
`password`	a character string
`connection`	the id of the connection already registered to use. By default the first registered connection is used.
`remember`	if TRUE connection information is saved localy in a file and used to automatically connect the next time. The default is set to FALSE. All the connections that have been saved can be listed with lsBedConnections and any of them can be forgotten with forgetBedConnection.
`useCache`	if TRUE the results of large queries can be saved locally in a file. The default is FALSE for policy reasons. But it is recommended to set it to TRUE to improve the speed of recurrent queries. If NA (default parameter) the value is taken from former connection if it exists or it is set to FALSE.
`importPath`	the path to the import folder for loading information in BED (used only when feeding the database ==> default: NULL)
`.opts`	a named list identifying the curl options for the handle (see `neo2R::startGraph()`).

Details

Be careful that you should reconnect to BED database each time the environment is reloaded. It is done automatically if `remember` is set to TRUE.

Information about how to get an instance of the BED 'Neo4j' database is provided here:

- `https://github.com/patzaw/BED#bed-database-instance-available-as-a-docker-image`
- `https://github.com/patzaw/BED#build-a-bed-database-instance`

Value

This function does not return any value. It prepares the BED environment to allow transparent DB calls.

See Also

checkBedConn, lsBedConnections, forgetBedConnection

`convBeIdLists` *Converts lists of BE IDs*

Description

Converts lists of BE IDs

Usage

```
convBeIdLists(idList, entity = FALSE, ...)
```

Arguments

`idList`	a list of IDs lists
`entity`	if TRUE returns BE instead of BEID (default: FALSE). BE CAREFUL, THIS INTERNAL ID IS NOT STABLE AND CANNOT BE USED AS A REFERENCE. This internal identifier is useful to avoid biases related to identifier redundancy. See <../doc/BED.html#3_managing_identifiers>
`...`	params for the convBeIds function

Value

A list of convBeIds ouput ids. Scope ("be", "source" "organism" and "entity" (see Arguments)) is provided as a named list in the "scope" attributes: `attr(x, "scope")`

See Also

convBeIds, convDfBeIds

Examples

```
## Not run:
convBeIdLists(
   idList=list(a=c("10", "100"), b=c("1000")),
   from="Gene",
   from.source="EntrezGene",
   from.org="human",
   to.source="Ens_gene"
)
```

```
## End(Not run)
```

`convBeIds` *Converts BE IDs*

Description

Converts BE IDs

Usage

```
convBeIds(
  ids,
  from,
  from.source,
  from.org,
  to,
  to.source,
  to.org,
  caseSensitive = FALSE,
  canonical = FALSE,
  prefFilter = FALSE,
  restricted = TRUE,
  recache = FALSE,
  limForCache = 2000
)
```

Arguments

`ids`	list of identifiers
`from`	a character corresponding to the biological entity or Probe. **Guessed if not provided**
`from.source`	a character corresponding to the ID source. **Guessed if not provided**
`from.org`	a character corresponding to the organism. **Guessed if not provided**
`to`	a character corresponding to the biological entity or Probe
`to.source`	a character corresponding to the ID source
`to.org`	a character corresponding to the organism
`caseSensitive`	if TRUE the case of provided symbols is taken into account during search. This option will only affect the conversion from "Symbol" (default: caseSensitive=FALSE). All the other conversion will be case sensitive.
`canonical`	if TRUE, only returns the canonical "Symbol". (default: FALSE)
`prefFilter`	boolean indicating if the results should be filter to keep only preferred BEID of BE when they exist (default: FALSE). If there are several preferred BEID of a BE, all are kept. If there are no preferred BEID of a BE, all non-preferred BEID are kept.

`restricted`	boolean indicating if the results should be restricted to current version of to BEID db. If FALSE former BEID are also returned: **Depending on history it can take a very long time to return a very large result!**
`recache`	a logical value indicating if the results should be taken from cache or recomputed
`limForCache`	if there are more ids than limForCache. Results are collected for all IDs (beyond provided ids) and cached for futur queries. If not, results are collected only for provided ids and not cached.

Value

a data.frame with the following columns:

- **from**: the input IDs
- **to**: the corresponding IDs in `to.source`
- **to.preferred**: boolean indicating if the to ID is a preferred ID for the corresponding entity.
- **to.entity**: the entity technical ID of the `to` IDs

This data.frame can be filtered in order to remove duplicated from/to.entity associations which can lead information bias. Scope ("be", "source" and "organism") is provided as a named list in the "scope" attributes: `attr(x, "scope")`

See Also

getBeIdConvTable, convBeIdLists, convDfBeIds

Examples

```
## Not run:
oriId <- c("10", "100")
convBeIds(
   ids=oriId,
   from="Gene",
   from.source="EntrezGene",
   from.org="human",
   to.source="Ens_gene"
)
convBeIds(
   ids=oriId,
   from="Gene",
   from.source="EntrezGene",
   from.org="human",
   to="Peptide",
   to.source="Ens_translation"
)
convBeIds(
   ids=oriId,
   from="Gene",
   from.source="EntrezGene",
   from.org="human",
   to="Peptide",
```

```
    to.source="Ens_translation",
    to.org="mouse"
)

## End(Not run)
```

`convDfBeIds` *Add BE ID conversion to a data frame*

Description

Add BE ID conversion to a data frame

Usage

```
convDfBeIds(df, idCol = NULL, entity = FALSE, ...)
```

Arguments

`df`	the data.frame to be converted
`idCol`	the column in which ID to convert are. If NULL (default) the row names are taken.
`entity`	if TRUE returns BE instead of BEID (default: FALSE). BE CAREFUL, THIS INTERNAL ID IS NOT STABLE AND CANNOT BE USED AS A REFERENCE. This internal identifier is useful to avoid biases related to identifier redundancy. See `../doc/BED.html#3_managing_identifiers`
`...`	params for the convBeIds function

Value

A data.frame with converted IDs. Scope ("be", "source", "organism" and "entity" (see Arguments)) is provided as a named list in the "scope" attributes: `attr(x, "scope")`.

See Also

convBeIds, convBeIdLists

Examples

```
## Not run:
toConv <- data.frame(a=1:2, b=3:4)
rownames(toConv) <- c("10", "100")
convDfBeIds(
    df=toConv,
    from="Gene",
    from.source="EntrezGene",
    from.org="human",
```

```
    to.source="Ens_gene"
)

## End(Not run)
```

dumpEnsCore *Feeding BED: Dump table from the Ensembl core database*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
dumpEnsCore(
  organism,
  release,
  gv,
  ddir,
  toDump = c("attrib_type", "gene_attrib", "transcript", "external_db", "gene",
   "translation", "external_synonym", "object_xref", "xref", "stable_id_event"),
  env = parent.frame(n = 1)
)
```

Arguments

`organism`	the organism to download (e.g. "Homo sapiens").
`release`	Ensembl release (e.g. "83")
`gv`	version of the genome (e.g. "38")
`ddir`	path to the directory where the data should be saved
`toDump`	the list of tables to download
`env`	the R environment in which to load the tables when downloaded

dumpNcbiDb *Feeding BED: Dump tables from the NCBI gene DATA*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
dumpNcbiDb(
  taxOfInt,
  reDumpThr,
  ddir,
 toLoad = c("gene_info", "gene2ensembl", "gene_group", "gene_orthologs", "gene_history",
    "gene2refseq"),
  env = parent.frame(n = 1),
  curDate
)
```

Arguments

`taxOfInt`	the organism to download (e.g. "9606").
`reDumpThr`	time difference threshold between 2 downloads
`ddir`	path to the directory where the data should be saved
`toLoad`	the list of tables to load
`env`	the R environment in which to load the tables when downloaded
`curDate`	current date as given by Sys.Date

`dumpNcbiTax` *Feeding BED: Dump tables with taxonomic information from NCBI*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
dumpNcbiTax(
  reDumpThr,
  ddir,
  toDump = c("names.dmp"),
  env = parent.frame(n = 1),
  curDate
)
```

Arguments

`reDumpThr`	time difference threshold between 2 downloads
`ddir`	path to the directory where the data should be saved
`toDump`	the list of tables to load
`env`	the R environment in which to load the tables when downloaded
`curDate`	current date as given by Sys.Date

---

dumpUniprotDb *Feeding BED: Dump and preprocess flat data files from Uniprot*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
dumpUniprotDb(
  taxOfInt,
  divOfInt,
  release,
  ddir,
  ftp = "ftp://ftp.expasy.org/databases/uniprot",
  env = parent.frame(n = 1)
)
```

Arguments

`taxOfInt`	the organism of interest (e.g., "9606" for human, "10090" for mouse or "10116" for rat)
`divOfInt`	the taxonomic division to which the organism belong (e.g., "human", "rodents", "mammals", "vertebrates")
`release`	the release of interest (check if already downloaded)
`ddir`	path to the directory where the data should be saved
`ftp`	location of the ftp site
`env`	the R environment in which to load the tables when built

---

exploreBe *Explore BE identifiers*

Description

This function uses visNetwork to draw all the identifiers corresponding to one BE (including ProbeID and BESymbol)

Usage

```
exploreBe(id, source, be, showBE = FALSE, showProbes = FALSE)
```

Arguments

`id`	one ID for the BE
`source`	the ID source database. **Guessed if not provided**
`be`	the type of BE. **Guessed if not provided**
`showBE`	boolean. If TRUE the Biological Entity corresponding to the id is shown. If id is isolated (not mapped to any other ID or symbol) BE is shown anyway.
`showProbes`	boolean. If TRUE, probes targeting any BEID are shown.

Examples

```
## Not run:
exploreBe("Gene", "100", "EntrezGene")

## End(Not run)
```

`exploreConvPath` *Explore the shortest convertion path between two identifiers*

Description

This function uses visNetwork to draw all the shortest convertion paths between two identifiers (including ProbeID).

Usage

```
exploreConvPath(
  from.id,
  to.id,
  from,
  from.source,
  to,
  to.source,
  edgeDirection = FALSE,
  verbose = FALSE
)
```

Arguments

`from.id`	the first identifier
`to.id`	the second identifier
`from`	the type of entity: `listBe()` or Probe. **Guessed if not provided**
`from.source`	the identifier source: database or platform. **Guessed if not provided**
`to`	the type of entity: `listBe()` or Probe. **Guessed if not provided**
`to.source`	the identifier source: database or platform. **Guessed if not provided**
`edgeDirection`	a logical value indicating if the direction of the edges should be drawn.
`verbose`	if TRUE the cypher query is shown

Examples

```
## Not run:
exploreConvPath(
   from.id="ENST00000413465",
   from="Transcript", from.source="Ens_transcript",
   to.id="ENSMUST00000108658",
   to="Transcript", to.source="Ens_transcript"
)

## End(Not run)
```

`filterByBEID` *Filter an object to keep only a set of BEIDs*

Description

Filter an object to keep only a set of BEIDs

Usage

```
filterByBEID(x, toKeep, ...)
```

Arguments

`x`	an object representing a collection of BEID (e.g. BEIDList)
`toKeep`	a vector of elements to keep
`...`	method specific parameters

`findBe` *Find Biological Entity*

Description

Find Biological Entity in BED based on their IDs, symbols and names

Usage

```
findBe(
  be = NULL,
  organism = NULL,
  ncharSymb = 4,
  ncharName = 8,
  restricted = TRUE,
  by = 20,
  exclude = c("BEDTech_gene", "BEDTech_transcript")
)
```

Arguments

`be`	optional. If provided the search is focused on provided BEs.
`organism`	optional. If provided the search is focused on provided organisms.
`ncharSymb`	The minimum number of characters in searched to consider incomplete symbol matches.
`ncharName`	The minimum number of characters in searched to consider incomplete name matches.
`restricted`	boolean indicating if the results should be restricted to current version of to BEID db. If FALSE former BEID are also returned: **Depending on history it can take a very long time to return a very large result!**
`by`	number of found items to be converted into relevant IDs.
`exclude`	database to exclude from possible selection. Used to filter out technical database names such as "BEDTech_gene" and "BEDTech_transcript" used to manage orphan IDs (not linked to any gene based on information taken from sources)

Value

A data frame with the following fields:

- **found**: the element found in BED corresponding to the searched term
- **be**: the type of the element
- **source**: the source of the element
- **organism**: the related organism
- **entity**: the related entity internal ID
- **ebe**: the BE of the related entity
- **canonical**: if the symbol is canonical
- **Relevant ID**: the seeked element id
- **Symbol**: the symbol(s) of the corresponding gene(s)
- **Name**: the symbol(s) of the corresponding gene(s)

Scope ("be", "source" and "organism") is provided as a named list in the "scope" attributes: ‘attr(x, "scope")“

`findBeids` *Find Biological Entity identifiers*

Description

Find Biological Entity identifiers

Usage

```
findBeids(toGene = TRUE, ...)
```

Arguments

`toGene`	focus on gene entities (default=TRUE): matches from other BE are converted to genes.
`...`	parameters for beidsServer

Value

NULL if not any result, or a data.frame with the selected values and the following column:

- **value**: the BE identifier
- **preferred**: preferred identifier for the same BE in the same scope
- **be**: the type of biological entity
- **source**: the source of the identifier
- **organism**: the organism of the BE
- **canonical** (if toGene==TRUE): canonical gene product? (if known)
- **symbol**: the symbol of the identifier (if any)

`firstCommonUpstreamBe` *First common upstream BE*

Description

Returns the first common Biological Entity (BE) upstream a set of BE.

Usage

```
firstCommonUpstreamBe(beList = listBe(), uniqueOrg = TRUE)
```

Arguments

`beList`	a character vector containing BE
`uniqueOrg`	a logical value indicating if as single organism is under focus. If false "Gene" is returned.

Details

This function is used to identified the level at which different BE should be compared. Peptides and transcripts should be compared at the level of transcripts whereas transcripts and objects should be compared at the level of genes. BE from different organism should be compared at the level of genes using homologs.

See Also

listBe

Examples

```
## Not run:
firstCommonUpstreamBe(c("Object", "Transcript"))
firstCommonUpstreamBe(c("Peptide", "Transcript"))
firstCommonUpstreamBe(c("Peptide", "Transcript"), uniqueOrg=FALSE)

## End(Not run)
```

`focusOnScope` *Focus a BE related object on a specific identifier (BEID) scope*

Description

Focus a BE related object on a specific identifier (BEID) scope

Usage

```
focusOnScope(
  x,
  be,
  source,
  organism,
  scope,
  force,
  restricted,
  prefFilter,
  ...
)
```

Arguments

`x`	an object representing a collection of BEID (e.g. BEIDList)
`be`	the type of biological entity to focus on. Used if `is.null(scope)`
`source`	the source of BEID to focus on. Used if `is.null(scope)`
`organism`	the organism of BEID to focus on. Used if `is.null(scope)`
`scope`	a list with the following element: • **be** • **source** • **organism**
`force`	if TRUE the conversion is done even between identical scopes (default: FALSE)
`restricted`	if TRUE (default) the BEID are limited to current version of the source
`prefFilter`	if TRUE (default) the BEID are limited to prefered identifiers when they exist
`...`	method specific parameters for BEID conversion

Value

Depends on the class of x

`focusOnScope.BEIDList` *Convert a BEIDList object in a specific identifier (BEID) scope*

Description

Convert a BEIDList object in a specific identifier (BEID) scope

Usage

```
## S3 method for class 'BEIDList'
focusOnScope(
  x,
  be = NULL,
  source = NULL,
  organism = NULL,
  scope = NULL,
  force = FALSE,
  restricted = TRUE,
  prefFilter = TRUE,
  ...
)
```

Arguments

`x`	the BEIDList to be converted
`be`	the type of biological entity to focus on. If NULL (default), it's taken from `scope(x)`. Used if `is.null(scope)`
`source`	the source of BEID to focus on. If NULL (default), it's taken from `scope(x)`. Used if `is.null(scope)`
`organism`	the organism of BEID to focus on. If NULL (default), it's taken from `scope(x)`. Used if `is.null(scope)`
`scope`	a list with the following element: • **be** • **source** • **organism**
`force`	if TRUE the conversion is done even between identical scopes (default: FALSE)
`restricted`	if TRUE (default) the BEID are limited to current version of the source
`prefFilter`	if TRUE (default) the BEID are limited to prefered identifiers when they exist
`...`	additional parameters to the BEID conversion function

Value

A BEIDList

`forgetBedConnection` *Forget a BED connection*

Description

Forget a BED connection

Usage

```
forgetBedConnection(connection, save = FALSE)
```

Arguments

`connection`	the id of the connection to forget.
`save`	a logical. Should be set to TRUE to save the updated list of connections in the file space (default to FALSE to comply with CRAN policies).

See Also

lsBedConnections, checkBedConn, connectToBed

`genBePath` *Construct CQL sub-query to map 2 biological entity*

Description

Internal use

Usage

```
genBePath(from, to, onlyR = FALSE)
```

Arguments

`from`	one biological entity (BE)
`to`	one biological entity (BE)
`onlyR`	logical. If TRUE (default: FALSE) it returns only the names of the relationships and not the cypher sub-query

Value

A character value corresponding to the sub-query. Or, if onlyR, a character vector with the names of the relationships.

See Also

genProbePath, listBe

`geneIDsToAllScopes` *Find all GeneID, ObjectID, TranscriptID, PeptideID and ProbeID corresponding to a Gene in any organism*

Description

Find all GeneID, ObjectID, TranscriptID, PeptideID and ProbeID corresponding to a Gene in any organism

Usage

```
geneIDsToAllScopes(
  geneids,
  source,
  organism,
  entities = NULL,
  orthologs = TRUE,
  canonical_symbols = TRUE
)
```

Arguments

`geneids`	a character vector of gene identifiers
`source`	the source of gene identifiers. **Guessed if not provided**
`organism`	the gene organism. **Guessed if not provided**
`entities`	a numeric vector of gene entity. If NULL (default), geneids, source and organism arguments are used to identify genes. Be carefull when using entities as these identifiers are not stable.
`orthologs`	return identifiers from orthologs
`canonical_symbols`	return only canonical symbols (default: TRUE).

Value

A data.frame with the following fields:

- **value**: the identifier
- **preferred**: preferred identifier for the same BE in the same scope
- **be**: the type of BE
- **organism**: the BE organism
- **source**: the source of the identifier
- **canonical**: canonical gene product (logical)
- **symbol**: canonical symbol of the identifier
- **Gene_entity**: the gene entity input

- **GeneID** (optional): the gene ID input
- **Gene_source** (optional): the gene source input
- **Gene_organism** (optional): the gene organism input

`genProbePath` *Identify the biological entity (BE) targeted by probes and construct the CQL sub-query to map probes to the BE*

Description

Internal use

Usage

```
genProbePath(platform)
```

Arguments

`platform`	the platform of the probes

Value

A character value corresponding to the sub-query. The `attr(,"be")` correspond to the BE targeted by probes

See Also

genBePath, listPlatforms

`getAllBeIdSources` *List all the source databases of BE identifiers whatever the BE type*

Description

List all the source databases of BE identifiers whatever the BE type

Usage

```
getAllBeIdSources(recache = FALSE)
```

Arguments

`recache`	boolean indicating if the CQL query should be run even if the table is already in cache

Value

A data.frame indicating the BE related to the ID source (database).

See Also

listBeIdSources, listPlatforms

`getBeIdConvTable` *Get a conversion table between biological entity (BE) identifiers*

Description

Get a conversion table between biological entity (BE) identifiers

Usage

```
getBeIdConvTable(
  from,
  to = from,
  from.source,
  to.source,
  organism,
  caseSensitive = FALSE,
  canonical = FALSE,
  restricted = TRUE,
  entity = TRUE,
  verbose = FALSE,
  recache = FALSE,
  filter = NULL,
  limForCache = 100
)
```

Arguments

`from`	one BE or "Probe"
`to`	one BE or "Probe"
`from.source`	the from BE ID database if BE or the from probe platform if Probe
`to.source`	the to BE ID database if BE or the to probe platform if Probe
`organism`	organism name
`caseSensitive`	if TRUE the case of provided symbols is taken into account during the conversion and selection. This option will only affect the conversion from "Symbol" (default: caseSensitive=FALSE). All the other conversion will be case sensitive.
`canonical`	if TRUE, only returns the canonical "Symbol". (default: FALSE)

`restricted`	boolean indicating if the results should be restricted to current version of to BEID db. If FALSE former BEID are also returned: **Depending on history it can take a very long time to return a very large result!**
`entity`	boolean indicating if the technical ID of to BE should be returned
`verbose`	boolean indicating if the CQL query should be displayed
`recache`	boolean indicating if the CQL query should be run even if the table is already in cache
`filter`	character vector on which to filter from IDs. If NULL (default), the result is not filtered: all from IDs are taken into account.
`limForCache`	if there are more filter than limForCache results are collected for all IDs (beyond provided ids) and cached for futur queries. If not, results are collected only for provided ids and not cached.

Value

a data.frame mapping BE IDs with the following fields:

- **from**: the from BE ID
- **to**: the to BE ID
- **entity**: (optional) the technical ID of to BE

See Also

getHomTable, listBe, listPlatforms, listBeIdSources

Examples

```
## Not run:
getBeIdConvTable(
    from="Gene", from.source="EntrezGene",
    to.source="Ens_gene",
    organism="human"
)

## End(Not run)
```

`getBeIdDescription` *Get description of Biological Entity identifiers*

Description

This description can be used for annotating tables or graph based on BE IDs.

Usage

```
getBeIdDescription(ids, be, source, organism, ...)
```

Arguments

`ids`	list of identifiers
`be`	one BE. **Guessed if not provided**
`source`	the BE ID database. **Guessed if not provided**
`organism`	organism name. **Guessed if not provided**
`...`	further arguments for getBeIdNames and getBeIdSymbols functions

Value

a data.frame providing for each BE IDs (row.names are provided BE IDs):

- **id**: the BE ID
- **symbol**: the BE symbol
- **name**: the corresponding name

See Also

getBeIdNames, getBeIdSymbols

Examples

```
## Not run:
getBeIdDescription(
   ids=c("10", "100"),
   be="Gene",
   source="EntrezGene",
   organism="human"
)

## End(Not run)
```

`getBeIdNames` *Get names of Biological Entity identifiers*

Description

Get names of Biological Entity identifiers

Usage

```
getBeIdNames(ids, be, source, organism, limForCache = 4000, ...)
```

Arguments

`ids`	list of identifiers
`be`	one BE. **Guessed if not provided**
`source`	the BE ID database. **Guessed if not provided**
`organism`	organism name. **Guessed if not provided**
`limForCache`	if there are more ids than limForCache results are collected for all IDs (beyond provided ids) and cached for futur queries. If not, results are collected only for provided ids and not cached.
`...`	params for the getBeIdNameTable function

Value

a data.frame mapping BE IDs and names with the following fields:

- **id**: the BE ID
- **name**: the corresponding name
- **direct**: true if the name is directly related to the BE ID
- **entity**: (optional) the technical ID of to BE

See Also

getBeIdNameTable, getBeIdSymbols

Examples

```
## Not run:
getBeIdNames(
   ids=c("10", "100"),
   be="Gene",
   source="EntrezGene",
   organism="human"
)

## End(Not run)
```

`getBeIdNameTable` *Get a table of biological entity (BE) identifiers and names*

Description

Get a table of biological entity (BE) identifiers and names

Usage

```
getBeIdNameTable(
  be,
  source,
  organism,
  restricted,
  entity = TRUE,
  verbose = FALSE,
  recache = FALSE,
  filter = NULL
)
```

Arguments

`be`	one BE
`source`	the BE ID database
`organism`	organism name
`restricted`	boolean indicating if the results should be restricted to direct names
`entity`	boolean indicating if the technical ID of BE should be returned
`verbose`	boolean indicating if the CQL query should be displayed
`recache`	boolean indicating if the CQL query should be run even if the table is already in cache
`filter`	character vector on which to filter id. If NULL (default), the result is not filtered: all IDs are taken into account.

Value

a data.frame with the following fields:

- **id**: the from BE ID
- **name**: the BE name
- **direct**: false if the symbol is not directly associated to the BE ID
- **entity**: (optional) the technical ID of to BE

See Also

getBeIdNames, getBeIdSymbolTable

Examples

```
## Not run:
getBeIdNameTable(
   be="Gene",
   source="EntrezGene",
   organism="human"
)
```

```
## End(Not run)
```

---

getBeIds *Get biological entities identifiers*

Description

Get biological entities identifiers

Usage

```
getBeIds(
  be = c(listBe(), "Probe"),
  source,
  organism = NA,
  restricted,
  entity = TRUE,
  attributes = NULL,
  verbose = FALSE,
  recache = FALSE,
  filter = NULL,
  caseSensitive = FALSE,
  limForCache = 100,
  bef = NULL
)
```

Arguments

`be`	one BE or "Probe"
`source`	the BE ID database or "Symbol" if BE or the probe platform if Probe
`organism`	organism name
`restricted`	boolean indicating if the results should be restricted to current version of to BEID db. If FALSE former BEID are also returned.
`entity`	boolean indicating if the technical ID of BE should be returned
`attributes`	a character vector listing attributes that should be returned.
`verbose`	boolean indicating if the CQL query should be displayed
`recache`	boolean indicating if the CQL query should be run even if the table is already in cache
`filter`	character vector on which to filter id. If NULL (default), the result is not filtered: all IDs are taken into account.
`caseSensitive`	if TRUE the case of provided symbols is taken into account. This option will only affect "Symbol" source (default: caseSensitive=FALSE).
`limForCache`	if there are more filter than limForCache results are collected for all IDs (beyond provided ids) and cached for futur queries. If not, results are collected only for provided ids and not cached.
`bef`	For internal use only

Value

a data.frame mapping BE IDs with the following fields:

- **id**: the BE ID
- **BE**: IF entity is TRUE the technical ID of BE
- **db.version**: IF be is not "Probe" and source not "Symbol" the version of the DB
- **db.deprecated**: IF be is not "Probe" and source not "Symbol" a value if the BE ID is deprecated or FALSE if it's not
- **canonical**: IF source is "Symbol" TRUE if the symbol is canonical
- **organism**: IF be is "Probe" the organism of the targeted BE

If attributes are part of the query, additional columns for each of them. Scope ("be", "source" and "organism") is provided as a named list in the "scope" attributes: `attr(x, "scope")`

See Also

listPlatforms, listBeIdSources

Examples

```
## Not run:
beids <- getBeIds(be="Gene", source="EntrezGene", organism="human", restricted=TRUE)

## End(Not run)
```

`getBeIdSymbols` *Get symbols of Biological Entity identifiers*

Description

Get symbols of Biological Entity identifiers

Usage

```
getBeIdSymbols(ids, be, source, organism, limForCache = 4000, ...)
```

Arguments

`ids`	list of identifiers
`be`	one BE. **Guessed if not provided**
`source`	the BE ID database. **Guessed if not provided**
`organism`	organism name. **Guessed if not provided**
`limForCache`	if there are more ids than limForCache. Results are collected for all IDs (beyond provided ids) and cached for futur queries. If not, results are collected only for provided ids and not cached.
`...`	params for the getBeIdSymbolTable function

Value

a data.frame with the following fields:

- **id**: the from BE ID
- **symbol**: the BE symbol
- **canonical**: true if the symbol is canonical for the direct BE ID
- **direct**: false if the symbol is not directly associated to the BE ID
- **entity**: (optional) the technical ID of to BE

See Also

getBeIdSymbolTable, getBeIdNames

Examples

```
## Not run:
getBeIdSymbols(
   ids=c("10", "100"),
   be="Gene",
   source="EntrezGene",
   organism="human"
)

## End(Not run)
```

`getBeIdSymbolTable` *Get a table of biological entity (BE) identifiers and symbols*

Description

Get a table of biological entity (BE) identifiers and symbols

Usage

```
getBeIdSymbolTable(
  be,
  source,
  organism,
  restricted,
  entity = TRUE,
  verbose = FALSE,
  recache = FALSE,
  filter = NULL
)
```

Arguments

`be`	one BE
`source`	the BE ID database
`organism`	organism name
`restricted`	boolean indicating if the results should be restricted to direct symbols
`entity`	boolean indicating if the technical ID of BE should be returned
`verbose`	boolean indicating if the CQL query should be displayed
`recache`	boolean indicating if the CQL query should be run even if the table is already in cache
`filter`	character vector on which to filter id. If NULL (default), the result is not filtered: all IDs are taken into account.

Value

a data.frame with the following fields:

- **id**: the from BE ID
- **symbol**: the BE symbol
- **canonical**: true if the symbol is canonical for the direct BE ID
- **direct**: false if the symbol is not directly associated to the BE ID
- **entity**: (optional) the technical ID of to BE

See Also

getBeIdSymbols, getBeIdNameTable

Examples

```
## Not run:
getBeIdSymbolTable(
   be="Gene",
   source="EntrezGene",
   organism="human"
)

## End(Not run)
```

`getBeIdURL` *Get reference URLs for BE IDs*

Description

Get reference URLs for BE IDs

Usage

```
getBeIdURL(ids, databases)
```

Arguments

`ids`	the BE ID
`databases`	the databases from which each ID has been taken (if only one database is provided it is chosen for all ids)

Value

A character vector of the same length than ids corresponding to the relevant URLs. NA is returned is there is no URL corresponding to the provided database.

Examples

```
## Not run:
getBeIdURL(c("100", "ENSG00000145335"), c("EntrezGene", "Ens_gene"))

## End(Not run)
```

`getDirectOrigin` *Get the direct origin of BE identifiers*

Description

The origin is directly taken as provided by the original database. This function does not return indirect relationships.

Usage

```
getDirectOrigin(
  ids,
  sources = NULL,
  process = c("is_expressed_as", "is_translated_in", "codes_for")
)
```

Arguments

`ids`	list of product identifiers
`sources`	a character vector corresponding to the possible product ID sources. If NULL (default), all sources are considered
`process`	the production process among: "is_expressed_as", "is_translated_in", "codes_for".

Value

a data.frame with the following columns:

- **origin**: the origin BE identifiers
- **osource**: the origin database
- **product**: the product BE identifiers
- **psource**: the production database
- **canonical**: whether the production process is canonical or not

The process is also returned as an attribute of the data.frame.

See Also

getDirectOrigin, convBeIds

Examples

```
## Not run:
oriId <- c("XP_016868427", "NP_001308979")
res <- getDirectOrigin(
   ids=oriId,
   source="RefSeq_peptide",
   process="is_translated_in"
)
attr(res, "process")

## End(Not run)
```

`getDirectProduct` *Get the direct product of BE identifiers*

Description

The product is directly taken as provided by the original database. This function does not return indirect relationships.

Usage

```
getDirectProduct(
  ids,
  sources = NULL,
  process = c("is_expressed_as", "is_translated_in", "codes_for"),
  canonical = NA
)
```

Arguments

`ids`	list of origin identifiers
`sources`	a character vector corresponding to the possible origin ID sources. If NULL (default), all sources are considered
`process`	the production process among: "is_expressed_as", "is_translated_in", "codes_for".
`canonical`	If TRUE returns only canonical production process. If FALSE returns only non-canonical production processes. If NA (default) canonical information is taken into account.

Value

a data.frame with the following columns:

- **origin**: the origin BE identifiers
- **osource**: the origin database
- **product**: the product BE identifiers
- **psource**: the production database
- **canonical**: whether the production process is canonical or not

The process is also returned as an attribute of the data.frame.

See Also

getDirectOrigin, convBeIds

Examples

```
## Not run:
oriId <- c("10", "100")
res <- getDirectProduct(
   ids=oriId,
   source="EntrezGene",
   process="is_expressed_as",
   canonical=NA
)
attr(res, "process")

## End(Not run)
```

`getEnsemblGeneIds` *Feeding BED: Download Ensembl DB and load gene information in BED*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
getEnsemblGeneIds(organism, release, gv, ddir, dbCref, dbAss, canChromosomes)
```

Arguments

`organism`	character vector of 1 element corresponding to the organism of interest (e.g. "Homo sapiens")
`release`	the Ensembl release of interest (e.g. "83")
`gv`	the genome version (e.g. "38")
`ddir`	path to the directory where the data should be saved
`dbCref`	a named vector of characters providing cross-reference DB of interest. These DB are also used to find indirect ID associations.
`dbAss`	a named vector of characters providing associated DB of interest. Unlike the DB in dbCref parameter, these DB are not used for indirect ID associations: the IDs are only linked to Ensembl IDs.
`canChromosomes`	canonical chromosmomes to be considered as preferred ID (e.g. c(1:22, "X", "Y", "MT") for human)

`getEnsemblPeptideIds` *Feeding BED: Download Ensembl DB and load peptide information in BED*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
getEnsemblPeptideIds(organism, release, gv, ddir, dbCref, canChromosomes)
```

Arguments

`organism`	character vector of 1 element corresponding to the organism of interest (e.g. "Homo sapiens")
`release`	the Ensembl release of interest (e.g. "83")
`gv`	the genome version (e.g. "38")
`ddir`	path to the directory where the data should be saved
`dbCref`	a named vector of characters providing cross-reference DB of interest. These DB are also used to find indirect ID associations.
`canChromosomes`	canonical chromosmomes to be considered as preferred ID (e.g. c(1:22, "X", "Y", "MT") for human)

`getEnsemblTranscriptIds`

Feeding BED: Download Ensembl DB and load transcript information in BED

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
getEnsemblTranscriptIds(organism, release, gv, ddir, dbCref, canChromosomes)
```

Arguments

`organism`	character vector of 1 element corresponding to the organism of interest (e.g. "Homo sapiens")
`release`	the Ensembl release of interest (e.g. "83")
`gv`	the genome version (e.g. "38")
`ddir`	path to the directory where the data should be saved
`dbCref`	a named vector of characters providing cross-reference DB of interest. These DB are also used to find indirect ID associations.
`canChromosomes`	canonical chromosmomes to be considered as preferred ID (e.g. c(1:22, "X", "Y", "MT") for human)

`getGeneDescription` *Get description of genes corresponding to Biological Entity identifiers*

Description

This description can be used for annotating tables or graph based on BE IDs.

Usage

```
getGeneDescription(
  ids,
  be,
  source,
  organism,
 gsource = largestBeSource(be = "Gene", organism = organism, rel = "is_known_as",
   restricted = TRUE),
  limForCache = 2000
)
```

Arguments

`ids`	list of identifiers
`be`	one BE. **Guessed if not provided**
`source`	the BE ID database. **Guessed if not provided**
`organism`	organism name. **Guessed if not provided**
`gsource`	the source of the gene IDs to use. It's chosen automatically by default.
`limForCache`	The number of ids above which the description is gathered for all be IDs and cached for futur queries.

Value

a data.frame providing for each BE IDs (row.names are provided BE IDs):

- **id**: the BE ID
- **gsource**: the Gene ID the column name provides the source of the used identifier
- **symbol**: the associated gene symbols
- **name**: the associated gene names

See Also

getBeIdDescription, getBeIdNames, getBeIdSymbols

Examples

```
## Not run:
getGeneDescription(
   ids=c("1438_at", "1552335_at"),
   be="Probe",
   source="GPL570",
   organism="human"
)

## End(Not run)
```

`getHomTable` *Get gene homologs between 2 organisms*

Description

Get gene homologs between 2 organisms

Usage

```
getHomTable(
  from.org,
  to.org,
  from.source = "Ens_gene",
  to.source = from.source,
  restricted = TRUE,
  verbose = FALSE,
  recache = FALSE,
  filter = NULL,
  limForCache = 100
)
```

Arguments

`from.org`	organism name
`to.org`	organism name
`from.source`	the from gene ID database
`to.source`	the to gene ID database
`restricted`	boolean indicating if the results should be restricted to current version of to BEID db. If FALSE former BEID are also returned: **Depending on history it can take a very long time to return a very large result!**
`verbose`	boolean indicating if the CQL query should be displayed
`recache`	boolean indicating if the CQL query should be run even if the table is already in cache

`filter`	character vector on which to filter from IDs. If NULL (default), the result is not filtered: all from IDs are taken into account.
`limForCache`	if there are more filter than limForCache results are collected for all IDs (beyond provided ids) and cached for futur queries. If not, results are collected only for provided ids and not cached.

Value

a data.frame mapping gene IDs with the following fields:

- **from**: the from gene ID
- **to**: the to gene ID

See Also

getBeIdConvTable

Examples

```
## Not run:
getHomTable(
   from.org="human",
   to.org="mouse"
)

## End(Not run)
```

`getNcbiGeneTransPep` *Feeding BED: Download NCBI gene DATA and load gene, transcript and peptide information in BED*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
getNcbiGeneTransPep(organism, reDumpThr = 1e+05, ddir, curDate)
```

Arguments

`organism`	character vector of 1 element corresponding to the organism of interest (e.g. "Homo sapiens")
`reDumpThr`	time difference threshold between 2 downloads
`ddir`	path to the directory where the data should be saved
`curDate`	current date as given by Sys.Date

`getOrgNames` *Get organism names from taxonomy IDs*

Description

Get organism names from taxonomy IDs

Usage

```
getOrgNames(taxID = NULL)
```

Arguments

`taxID`	a vector of taxonomy IDs. If NULL (default) the function lists all taxonomy IDs available in the DB.

Value

A data.frame mapping taxonomy IDs to organism names with the following fields:

- **taxID**: the taxonomy ID
- **name**: the organism name
- **nameClass**: the class of the name

See Also

getTaxId, listOrganisms

Examples

```
## Not run:
getOrgNames(c("9606", "10090"))
getOrgNames("9606")

## End(Not run)
```

`getRelevantIds` *Get relevant IDs for a formerly identified BE in a context of interest*

Description

DEPRECATED: use searchBeid and geneIDsToAllScopes instead. This function is meant to be used with searchId in order to implement a dictonary of identifiers of interest. First the searchId function is used to search a term. Then the getRelevantIds function is used to find the corresponding IDs in a context of interest.

Usage

```
getRelevantIds(
  d,
  selected = 1,
  be = c(listBe(), "Probe"),
  source,
  organism,
  restricted = TRUE,
  simplify = TRUE,
  verbose = FALSE
)
```

Arguments

`d`	the data.frame returned by searchId.
`selected`	the rows of interest in d
`be`	the BE in the context of interest
`source`	the source of the identifier in the context of interest
`organism`	the organism in the context of interest
`restricted`	boolean indicating if the results should be restricted to current version of to BEID db. If FALSE former BEID are also returned: **Depending on history it can take a very long time to return a very large result!**
`simplify`	if TRUE (default) duplicated IDs are removed from the output
`verbose`	if TRUE, the CQL query is shown

Value

The d data.frame with a new column providing the relevant ID in the context of interest and without the gene field. Scope ("be", "source" and "organism") is provided as a named list in the "scope" attributes: `attr(x, "scope")`

See Also

searchId

`getTargetedBe` *Identify the biological entity (BE) targeted by probes*

Description

Identify the biological entity (BE) targeted by probes

Usage

```
getTargetedBe(platform)
```

Arguments

`platform`	the platform of the probes

Value

The BE targeted by the platform

See Also

listPlatforms

Examples

```
## Not run:
getTargetedBe("GPL570")

## End(Not run)
```

`getTaxId` *Get taxonomy ID of an organism name*

Description

Get taxonomy ID of an organism name

Usage

```
getTaxId(name)
```

Arguments

`name`	the name of the organism

Value

A vector of taxonomy ID

See Also

getOrgNames, listOrganisms

Examples

```
## Not run:
getTaxId("human")

## End(Not run)
```

`getUniprot` *Feeding BED: Download Uniprot information in BED*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
getUniprot(organism, taxDiv, release, ddir)
```

Arguments

`organism`	character vector of 1 element corresponding to the organism of interest (e.g. "Homo sapiens")
`taxDiv`	the taxonomic division to which the organism belong (e.g., "human", "rodents", "mammals", "vertebrates")
`release`	the release of interest (check if already downloaded)
`ddir`	path to the directory where the data should be saved

`guessIdScope` *Guess biological entity (BE), database source and organism of a vector of identifiers.*

Description

Guess biological entity (BE), database source and organism of a vector of identifiers.

Usage

```
guessIdScope(ids, be, source, organism, tcLim = 100)

guessIdOrigin(...)
```

Arguments

`ids`	a character vector of identifiers
`be`	one BE or "Probe". **Guessed if not provided**
`source`	the BE ID database or "Symbol" if BE or the probe platform if Probe. **Guessed if not provided**
`organism`	organism name. **Guessed if not provided**
`tcLim`	number of identifiers to check to guess origin for the whole set. Inf ==> no limit.
`...`	params for `guessIdScope`

Value

A list (NULL if no match):

- **be**: a character vector of length 1 providing the best BE guess (NA if inconsistent with user input: be, source or organism)
- **source**: a character vector of length 1 providing the best source guess (NA if inconsistent with user input: be, source or organism)
- **organism*$: a character vector of length 1 providing the best organism guess (NA if inconsistent with user input: be, source or organism)

The "details" attribute (‘attr(x, "details")“) is a data frame providing numbers supporting the guess

Functions

- `guessIdOrigin()`: Deprecated version of guessIdScope

Examples

```
## Not run:
guessIdScope(ids=c("10", "100"))

## End(Not run)
```

`identicalScopes` *Check if two objects have the same BEID scope*

Description

Check if two objects have the same BEID scope

Usage

```
identicalScopes(x, y)
```

Arguments

`x`	the object to test
`y`	the object to test

Value

A logical indicating if the 2 scopes are identical

`is.BEIDList` *Check if the provided object is a BEIDList*

Description

Check if the provided object is a BEIDList

Usage

```
is.BEIDList(x)
```

Arguments

`x`	the object to check

Value

A logical value

`largestBeSource` *Autoselect source of biological entity identifiers*

Description

The selection is based on direct identifiers

Usage

```
largestBeSource(
  be,
  organism,
  rel = NA,
  restricted = TRUE,
  exclude = c("BEDTech_gene", "BEDTech_transcript")
)
```

Arguments

`be`	the biological entity under focus
`organism`	the organism under focus
`rel`	a type of relationship to consider in the query (e.g. "is_member_of") in order to focus on specific information. If NA (default) all be are taken into account whatever their available relationships.
`restricted`	boolean indicating if the results should be restricted to current version of to BEID db. If FALSE former BEID are also taken into account.
`exclude`	database to exclude from possible selection. Used to filter out technical database names such as "BEDTech_gene" and "BEDTech_transcript" used to manage orphan IDs (not linked to any gene based on information taken from sources)

Value

The name of the selected source. The selected source will be the one providing the largest number of current identifiers.

See Also

listBeIdSources

Examples

```
## Not run:
largestBeSource(be="Gene", "Mus musculus")

## End(Not run)
```

`listBe` *Lists all the biological entities (BE) available in the BED database*

Description

Lists all the biological entities (BE) available in the BED database

Usage

```
listBe()
```

Value

A character vector of biological entities (BE)

See Also

listPlatforms, listBeIdSources, listOrganisms

`listBeIdSources` *Lists all the databases taken into account in the BED database for a biological entity (BE)*

Description

Lists all the databases taken into account in the BED database for a biological entity (BE)

Usage

```
listBeIdSources(
  be = listBe(),
  organism,
  direct = FALSE,
  rel = NA,
  restricted = FALSE,
  recache = FALSE,
  verbose = FALSE,
  exclude = c()
)
```

Arguments

`be`	the BE on which to focus
`organism`	the name of the organism to focus on.
`direct`	a logical value indicating if only "direct" BE identifiers should be considered
`rel`	a type of relationship to consider in the query (e.g. "is_member_of") in order to focus on specific information. If NA (default) all be are taken into account whatever their available relationships.
`restricted`	boolean indicating if the results should be restricted to current version of to BEID db. If FALSE former BEID are also returned. There is no impact if direct is set to TRUE.
`recache`	boolean indicating if the CQL query should be run even if the table is already in cache
`verbose`	boolean indicating if the CQL query should be shown.
`exclude`	database to exclude from possible selection. Used to filter out technical database names such as "BEDTech_gene" and "BEDTech_transcript" used to manage orphan IDs (not linked to any gene based on information taken from sources)

Value

A data.frame indicating the number of ID in each available database with the following fields:

- **database**: the database name
- **nbBe**: number of distinct entities
- **nbId**: number of identifiers
- **be**: the BE under focus

See Also

listBe, largestBeSource

Examples

```
## Not run:
listBeIdSources(be="Transcript", organism="mouse")

## End(Not run)
```

`listDBAttributes` *List all attributes provided by a BEDB*

Description

List all attributes provided by a BEDB

Usage

```
listDBAttributes(dbname)
```

Arguments

`dbname`	the name of the database

Value

A character vector of attribute names

`listOrganisms` *Lists all the organisms available in the BED database*

Description

Lists all the organisms available in the BED database

Usage

```
listOrganisms()
```

Value

A character vector of organism scientific names

See Also

listPlatforms, listBeIdSources, listBe, getTaxId, getOrgNames

---

`listPlatforms` *Lists all the probe platforms available in the BED database*

Description

Lists all the probe platforms available in the BED database

Usage

```
listPlatforms(be = c(NA, listBe()))
```

Arguments

`be`	a character vector of BE on which to focus. if NA (default) all the BE are considered.

Value

A data.frame mapping platforms to BE with the following fields:

- **name**: the platform nam
- **description**: platform description
- **focus**: Targeted BE

See Also

listBe, listBeIdSources, listOrganisms, getTargetedBe

Examples

```
## Not run:
listPlatforms(be="Gene")
listPlatforms()

## End(Not run)
```

`loadBE` *Feeding BED: Load biological entities in BED*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadBE(
  d,
  be = "Gene",
  dbname,
  version = NA,
  deprecated = NA,
  taxId = NA,
  onlyId = FALSE
)
```

Arguments

`d`	a data.frame with information about the entities to be loaded. It should contain the following fields: "id". If there is a boolean column named "preferred", the value is loaded.
`be`	a character corresponding to the BE type (default: "Gene")
`dbname`	the DB from which the BE ID are taken
`version`	the version of the DB from which the BE IDs are taken
`deprecated`	NA (default) or the date when the ID was deprecated
`taxId`	the taxonomy ID of the BE organism
`onlyId`	a logical. If TRUE, only an BEID is created and not the corresponding BE.

`loadBeAttribute` *Feeding BED: Load attributes for biological entities in BED*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadBeAttribute(d, be = "Gene", dbname, attribute)
```

Arguments

`d`	a data.frame providing for each BE ID ("id" column) an attribute value ("value" column). There can be several values for each id.
`be`	a character corresponding to the BE type (default: "Gene")
`dbname`	the DB from which the BE ID are taken
`attribute`	the name of the attribute to be loaded

`loadBedModel` *Feeding BED: Load BED data model in neo4j*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadBedModel()
```

`loadBedOtherIndexes` *Feeding BED: Load additional indexes in neo4j*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadBedOtherIndexes()
```

`loadBedResult` *Get a BED query result from cache*

Description

Internal use

Usage

```
loadBedResult(name)
```

Arguments

`name`	the name of the query

See Also

cacheBedCall, cacheBedResult

`loadBENames` *Feeding BED: Load names associated to BEIDs*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadBENames(d, be = "Gene", dbname)
```

Arguments

`d`	a data.frame with information about the names to be loaded. It should contain the following fields: "id", "name".
`be`	a character corresponding to the BE type (default: "Gene")
`dbname`	the DB of BEID

`loadBESymbols` *Feeding BED: Load symbols associated to BEIDs*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadBESymbols(d, be = "Gene", dbname)
```

Arguments

`d`	a data.frame with information about the symbols to be loaded. It should contain the following fields: "id", "symbol" and "canonical" (optional).
`be`	a character corresponding to the BE type (default: "Gene")
`dbname`	the DB of BEID

`loadBEVersion` *Feeding BED: Load biological entities in BED with information about DB version*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadBEVersion(d, be = "Gene", dbname, taxId = NA, onlyId = FALSE)
```

Arguments

`d`	a data.frame with information about the entities to be loaded. It should contain the following fields: "id", "version" and "deprecated".
`be`	a character corresponding to the BE type (default: "Gene")
`dbname`	the DB from which the BE ID are taken
`taxId`	the taxonomy ID of the BE organism
`onlyId`	a logical. If TRUE, only an BEID is created and not the corresponding BE.

`loadCodesFor` *Feeding BED: Load correspondance between genes and objects as coding events*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadCodesFor(d, gdb, odb)
```

Arguments

`d`	a data.frame with information about the coding events. It should contain the following fields: "gid" and "oid"
`gdb`	the DB of Gene IDs
`odb`	the DB of Object IDs

`loadCorrespondsTo` *Feeding BED: Load correspondances between BE IDs*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadCorrespondsTo(d, db1, db2, be = "Gene")
```

Arguments

`d`	a data.frame with information about the correspondances to be loaded. It should contain the following fields: "id1" and "id2".
`db1`	the DB of id1
`db2`	the DB of id2
`be`	a character corresponding to the BE type (default: "Gene")

`loadHistory` *Feeding BED: Load history of BEIDs*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadHistory(d, dbname, be = "Gene")
```

Arguments

`d`	a data.frame with information about the history. It should contain the following fields: "old" and "new".
`dbname`	the DB of BEID
`be`	a character corresponding to the BE type (default: "Gene")

`loadIsAssociatedTo` *Feeding BED: Load BE ID associations*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadIsAssociatedTo(d, db1, db2, be = "Gene")
```

Arguments

`d`	a data.frame with information about the associations to be loaded. It should contain the following fields: "id1" and "id2". At the end id1 is associated to id2 (this way and not the other).
`db1`	the DB of id1
`db2`	the DB of id2
`be`	a character corresponding to the BE type (default: "Gene")

Details

When associating one id1 to id2, the BE identified by id1 is deleted after that its production edges have been transferred to the BE identified by id2. After this operation all id "corresponding_to" id1 do not directly identify any BE as they are supposed to do. Thus, to run this function with id1 involved in "corresponds_to" edges.

`loadIsExpressedAs` *Feeding BED: Load correspondance between genes and transcripts as expression events*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadIsExpressedAs(d, gdb, tdb)
```

Arguments

`d`	a data.frame with information about the expression events. It should contain the following fields: "gid", "tid" and "canonical" (optional).
`gdb`	the DB of Gene IDs
`tdb`	the DB of Transcript IDs

`loadIsHomologOf` *Feeding BED: Load homology between BE IDs*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadIsHomologOf(d, db1, db2, be = "Gene")
```

Arguments

`d`	a data.frame with information about the homologies to be loaded. It should contain the following fields: "id1" and "id2".
`db1`	the DB of id1
`db2`	the DB of id2
`be`	a character corresponding to the BE type (default: "Gene")

`loadIsTranslatedIn` *Feeding BED: Load correspondance between transcripts and peptides as translation events*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadIsTranslatedIn(d, tdb, pdb)
```

Arguments

`d`	a data.frame with information about the translation events. It should contain the following fields: "tid", "pid" and "canonical" (optional).
`tdb`	the DB of Transcript IDs
`pdb`	the DB of Peptide IDs

`loadLuceneIndexes` *Feeding BED: Create Lucene indexes in neo4j*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadLuceneIndexes()
```

`loadNCBIEntrezGOFunctions` *Feeding BED: Load in BED GO functions associated to Entrez gene IDs from NCBI*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadNCBIEntrezGOFunctions(organism, reDumpThr = 1e+05, ddir, curDate)
```

Arguments

`organism`	character vector of 1 element corresponding to the organism of interest (e.g. "Homo sapiens")
`reDumpThr`	time difference threshold between 2 downloads
`ddir`	path to the directory where the data should be saved
`curDate`	current date as given by Sys.Date

`loadNcbiTax` *Feeding BED: Load taxonomic information from NCBI*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadNcbiTax(reDumpThr, ddir, orgOfInt = c("human", "rat", "mouse"), curDate)
```

Arguments

`reDumpThr`	time difference threshold between 2 downloads
`ddir`	path to the directory where the data should be saved
`orgOfInt`	organisms of interest: a character vector
`curDate`	current date as given by Sys.Date

`loadOrganisms` *Feeding BED: Load organisms in BED*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadOrganisms(d)
```

Arguments

`d`	a data.frame with 2 columns named "tax_id" and "name_txt" providing the taxonomic ID for each organism name

---

`loadPlf` *Feeding BED: Load a probes platform*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadPlf(name, description, be)
```

Arguments

`name`	the name of the platform
`description`	a description of the platform
`be`	the type of BE targeted by the platform

`loadProbes` *Feeding BED: Load probes targeting BE IDs*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
loadProbes(d, be = "Transcript", platform, dbname)
```

Arguments

`d`	a data.frame with information about the entities to be loaded. It should contain the following fields: "id" and "probeID".
`be`	a character corresponding to the BE targeted by the probes (default: "Transcript")
`platform`	the plateform gathering the probes
`dbname`	the DB from which the BE ID are taken

`lsBedCache` *List all the BED queries in cache and the total size of the cache*

Description

List all the BED queries in cache and the total size of the cache

Usage

```
lsBedCache(verbose = TRUE)
```

Arguments

`verbose`	if TRUE (default) prints a message displaying the total size of the cache

Value

A data.frame giving for each query (row names) its size in Bytes (column "size") and in human readable format (column "hr"). The attribute "Total" corresponds to the sum of all the file size.

See Also

clearBedCache

`lsBedConnections` *List all registered BED connection*

Description

List all registered BED connection

Usage

```
lsBedConnections()
```

See Also

connectToBed, forgetBedConnection, checkBedConn

`metadata` *Get object metadata*

Description

Get object metadata

Usage

```
metadata(x, ...)
```

Arguments

`x`	an object representing a collection of BEID (e.g. BEIDList)
`...`	method specific parameters

`metadata<-` *Set object metadata*

Description

Set object metadata

Usage

```
metadata(x) <- value
```

Arguments

`x`	an object representing a collection of BEID (e.g. BEIDList)
`value`	a data.frame with rownames or a column "**.lname**" all in names of l.

`registerBEDB` *Feeding BED: Register a database of biological entities in BED DB*

Description

Not exported to avoid unintended modifications of the DB.

Usage

```
registerBEDB(name, description = NA, currentVersion = NA, idURL = NA)
```

Arguments

`name`	of the database (e.g. "Ens_gene")
`description`	a short description of the database (e.g. "Ensembl gene")
`currentVersion`	the version taken into account in BED (e.g. 83)
`idURL`	the URL template to use to retrieve id information. A '%s' corresponding to the ID should be present in this character vector of length one.

`scope` *Get the BEID scope of an object*

Description

Get the BEID scope of an object

Usage

```
scope(x, ...)
```

Arguments

`x`	an object representing a collection of BEID (e.g. BEIDList)
`...`	method specific parameters

`scopes` *Get the BEID scopes of an object*

Description

Get the BEID scopes of an object

Usage

```
scopes(x, ...)
```

Arguments

`x`	an object representing a collection of BEID (e.g. BEIDList)
`...`	method specific parameters

Value

A tibble with 4 columns:

- be
- source
- organism
- Freq

`searchBeid` *Search a BEID*

Description

Search a BEID

Usage

```
searchBeid(x, clean_id_search = TRUE, clean_name_search = TRUE)
```

Arguments

`x`	a character value to search
`clean_id_search`	clean x to avoid error during ID search. Default: TRUE. Set it to false if you're sure of your lucene query.
`clean_name_search`	clean x to avoid error during ID search. Default: TRUE. Set it to false if you're sure of your lucene query.

Value

NULL if there is not any match or a data.frame with the following columns:

- **Value**: the matching term
- **From**: the type of the matched term (e.g. BESymbol, GeneID...)
- **BE**: the matching biological entity (BE)
- **BEID**: the BE identifier
- **Database**: the BEID reference database
- **Preferred**: TRUE if the BEID is considered as a preferred identifier
- **Symbol**: BEID canonical symbol
- **Name**: BEID name
- **Entity**: technical BE identifier
- **GeneID**: Corresponding gene identifier
- **Gene_DB**: Gene ID database
- **Preferred_gene**: TRUE if the GeneID is considered as a preferred identifier
- **Gene_symbol**: Gene symbol
- **Gene_name**: Gene name
- **Gene_entity**: technical gene identifier
- **Organism**: gene organism (scientific name)

`searchId` *Search identifier, symbol or name information*

Description

DEPRECATED: use searchBeid and geneIDsToAllScopes instead. This function is meant to be used with getRelevantIds in order to implement a dictonary of identifiers of interest. First the searchId function is used to search a term. Then the getRelevantIds function is used to find the corresponding ID in a context of interest.

Usage

```
searchId(
  searched,
  be = NULL,
  organism = NULL,
  ncharSymb = 4,
  ncharName = 8,
  verbose = FALSE
)
```

Arguments

`searched`	the searched term. Identifiers are searched by exact match. Symbols and names are also searched for partial match when searched is greater than ncharSymb and ncharName respectively.
`be`	optional. If provided the search is focused on provided BEs.
`organism`	optional. If provided the search is focused on provided organisms.
`ncharSymb`	The minimum number of characters in searched to consider incomplete symbol matches.
`ncharName`	The minimum number of characters in searched to consider incomplete name matches.
`verbose`	boolean indicating if the CQL queries should be displayed

Value

A data frame with the following fields:

- **found**: the element found in BED corresponding to the searched term
- **be**: the type of the element
- **source**: the source of the element
- **organism**: the related organism
- **entity**: the related entity internal ID
- **ebe**: the BE of the related entity
- **canonical**: if the symbol is canonical
- **gene**: list of the related genes BE internal ID

Exact matches are returned first folowed by the shortest elements.

See Also

getRelevantIds

`setBedVersion` *Feeding BED: Set the BED version*

Description

Not exported to avoid unintended modifications of the DB. This function is used when modifying the BED content.

Usage

```
setBedVersion(bedInstance, bedVersion)
```

Arguments

`bedInstance`	instance of BED to be set
`bedVersion`	version of BED to be set

`showBedDataModel` *Show the data model of BED*

Description

Show the shema of the BED data model.

Usage

```
showBedDataModel()
```

Index