

# Package ‘dbparser’

October 13, 2022

**Title** 'DrugBank' Database XML Parser

**Version** 1.2.0

**Description** This tool is for parsing the 'DrugBank' XML database <<https://www.drugbank.ca/>>. The parsed data are then returned in a proper 'R' dataframe with the ability to save them in a given database.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Imports** DBI, dplyr, odbc, progress, purrr, readr, RMariaDB, RSQLite, tibble, tools, XML

**RoxygenNote** 7.1.0

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**URL** <https://docs.ropensci.org/dbparser/>,  
<https://github.com/ropensci/dbparser/>

**BugReports** <https://github.com/ropensci/dbparser/issues>

**Depends** R (>= 2.10)

**NeedsCompilation** no

**Author** Mohammed Ali [aut, cre],  
Ali Ezzat [aut],  
Hao Zhu [rev],  
Emma Mendelsohn [rev]

**Maintainer** Mohammed Ali <moh\_fcis@yahoo.com>

**Repository** CRAN

**Date/Publication** 2020-08-26 12:10:03 UTC

**R topics documented:**

articles . . . . .	3
attachments . . . . .	5
books . . . . .	8
cett . . . . .	10
cett_actions_doc . . . . .	12
cett_doc . . . . .	14
cett_ex_identity_doc . . . . .	17
cett_go_doc . . . . .	19
cett_poly_doc . . . . .	21
cett_poly_pfms_doc . . . . .	24
cett_poly_syn_doc . . . . .	26
dbparser . . . . .	28
drugs . . . . .	29
drug_affected_organisms . . . . .	31
drug_ahfs_codes . . . . .	33
drug_atc_codes . . . . .	35
drug_calc_prop . . . . .	36
drug_categories . . . . .	38
drug_classification . . . . .	40
drug_dosages . . . . .	42
drug_element . . . . .	44
drug_element_options . . . . .	46
drug_exp_prop . . . . .	47
drug_external_links . . . . .	49
drug_ex_identity . . . . .	51
drug_food_interactions . . . . .	53
drug_general_information . . . . .	54
drug_groups . . . . .	57
drug_interactions . . . . .	58
drug_intern_brand . . . . .	60
drug_manufacturers . . . . .	62
drug_mixtures . . . . .	64
drug_packagers . . . . .	66
drug_patents . . . . .	68
drug_pathway . . . . .	70
drug_pathway_drugs . . . . .	72
drug_pathway_enzyme . . . . .	74
drug_pdb_entries . . . . .	75
drug_pharmacology . . . . .	77
drug_prices . . . . .	80
drug_products . . . . .	81
drug_reactions . . . . .	84
drug_reactions_enzymes . . . . .	86
drug_salts . . . . .	88
drug_sequences . . . . .	90
drug_snp_adverse_reactions . . . . .	92

drug_snp_effects . . . . .	94
drug_syn . . . . .	96
get_drugbank_exported_date . . . . .	98
get_drugbank_metadata . . . . .	99
get_drugbank_version . . . . .	99
links . . . . .	100
read_drugbank_xml_db . . . . .	102
references . . . . .	103
run_all_parsers . . . . .	104

**Index****107**


---

articles	<i>Drugs/ Carriers/ Enzymes/ Targets/ Transporters articles element parser</i>
----------	--

---

**Description**

Return a list of articles that were used as references for drugs carriers

**Usage**

```

drugs_articles(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

```

carriers_articles(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

```

enzymes_articles(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

```

targets_articles(
  save_table = FALSE,

```

```

    save_csv = FALSE,
    csv_path = ".",
    override_csv = FALSE,
    database_connection = NULL
  )

transporters_articles(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 4 variables:

**ref-id** Identifier for the article being referenced. This is unique across all reference types (books, links, article, attachments).

**pubmed-id** The PubMed identifier for the article.

**citation** Article citation in a standard format.

**parent\_id** drug/carrier/target/enzyme/transporter id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other references: [attachments](#), [books](#), [links](#), [references\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

attachments	<i>Drugs/ Carriers/ Enzymes/ Targets/ Transporters attachments element parser</i>
-------------	---

---

**Description**

Return a list of attachment that were used as references for drugs carriers

**Usage**

```
drugs_attachments(
  save_table = FALSE,
  save_csv = FALSE,
```

```
    csv_path = ".",
    override_csv = FALSE,
    database_connection = NULL
  )

carriers_attachments(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

enzymes_attachments(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

targets_attachments(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

transporters_attachments(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

### Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

**Value**

a tibble with 4 variables:

**ref-id** Identifier for the article being referenced. This is unique across all reference types (books, links, article, attachments).

**title** The title of the attachment.

**url** The url to download the attachment from.

**parent\_id** drug/carrier/target/enzyme/transporter id

**read\_drugbank\_xml\_db**

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other references: [articles](#), [books](#), [links](#), [references\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
```

```
# return parsed tibble.  
# if the csv exist override it and return it.  
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)  
  
## End(Not run)
```

---

books	<i>Drugs/ Carriers/ Enzymes/ Targets/ Transporters books element parser</i>
-------	---

---

## Description

Return a list of text books that were used as references for drugs, carriers, enzymes, targets or transporters

## Usage

```
drugs_textbooks(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
carriers_textbooks(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
enzymes_textbooks(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
targets_textbooks(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```



```
)

transporters_textbooks(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

### Arguments

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 4 variables:

**ref-id** Identifier for the article being referenced. This is unique across all reference types (books, links, article, attachments).

**isbn** ISBN identifying the textbook.

**citation** A Textbook citation in a standard format.

**parent\_id** drug/ carrier/ target/ enzyme/ transporter id

### read\_drugbank\_xml\_db

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

### See Also

Other references: [articles](#), [attachments](#), [links](#), [references\(\)](#)

### Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()
```

```
# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

cett

*Run all CETT related parsers*

---

## Description

Run all parsers that retrieve carriers, enzymes, targets and transporters related information

## Usage

```
cett(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

**Arguments**

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

**Value**

a list of all drugs parsed tibbles

**read\_drugbank\_xml\_db**

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other collective\_parsers: [drugs\(\)](#), [run\\_all\\_parsers\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)
```

```
# save parsed tibble as csv if it does not exist in given location,  
# and return parsed tibble.  
# if the csv exist before read it and return its data.  
run_all_parsers(save_csv = TRUE, csv_path = TRUE)  
  
# save parsed tibble as csv if it does not exist in current location and  
# return parsed tibble.  
# if the csv exist override it and return it.  
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)  
  
## End(Not run)
```

---

cett\_actions\_doc

*Carriers/ Enzymes/ Targets/ Transporters Actions parsers*

---

## Description

Collection of related actions

## Usage

```
carriers_actions(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)  
  
enzymes_actions(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)  
  
targets_actions(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)  
  
transporters_actions(  
  save_table = FALSE,
```

```

    save_csv = FALSE,
    csv_path = ".",
    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

**save\_table**      boolean, save table in database if true.

**save\_csv**        boolean, save csv version of parsed tibble if true

**csv\_path**        location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv**    override existing csv, if any, in case it is true in the new parse operation

**database\_connection**  
DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 2 variables:

**action** describe related action

**parent\_id** carrier/ target/ enzyme/ transporter id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other cett: [cett\\_doc](#), [cett\\_ex\\_identity\\_doc](#), [cett\\_go\\_doc](#), [cett\\_poly\\_doc](#), [cett\\_poly\\_pfms\\_doc](#), [cett\\_poly\\_syn\\_doc](#)

### Examples

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

```

```
# save parsed tibble as csv if it does not exist in current location,  
# and return parsed tibble.  
# if the csv exist before read it and return its data.  
run_all_parsers(save_csv = TRUE)  
  
# save in database, save parsed tibble as csv,  
# if it does not exist in current location and return parsed tibble.  
# if the csv exist before read it and return its data.  
run_all_parsers(save_table = TRUE, save_csv = TRUE,  
database_connection = sqlite_con)  
  
# save parsed tibble as csv if it does not exist in given location,  
# and return parsed tibble.  
# if the csv exist before read it and return its data.  
run_all_parsers(save_csv = TRUE, csv_path = TRUE)  
  
# save parsed tibble as csv if it does not exist in current location and  
# return parsed tibble.  
# if the csv exist override it and return it.  
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)  
  
## End(Not run)
```

---

cett\_doc

*Carriers/ Enzymes/ Targets/ Transporters parsers*

---

## Description

Protein targets of drug action, enzymes that are inhibited/induced or involved in metabolism, and carrier or transporter proteins involved in movement of the drug across biological membranes.

## Usage

```
carriers(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)  
  
enzymes(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```

targets(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

transporters(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 6 variables (8 for enzymes):

**id** Universal Protein Resource (UniProt) Identifier for the record

**name** related name

**organism** Organism that the protein comes from.

**known\_action** Whether the pharmacological action of the drug is due to this target interaction.

**inhibition-strength** Whether the strength of enzyme inhibition is strong, moderate, or unknown.  
**Only applies to enzymes**

**induction-strength** Whether the strength of enzyme induction is strong or unknown. **Only applies to enzymes**

**position** related position

**parent\_id** drugbank id

## read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

## See Also

Other cett: [cett\\_actions\\_doc](#), [cett\\_ex\\_identity\\_doc](#), [cett\\_go\\_doc](#), [cett\\_poly\\_doc](#), [cett\\_poly\\_pfms\\_doc](#), [cett\\_poly\\_syn\\_doc](#)

## Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```



---

cett\_ex\_identity\_doc *Carriers/ Enzymes/ Targets/ Transporters Polypeptide External Identifiers parsers*

---

### Description

Extract descriptions of identified polypeptide external identifiers for targets, enzymes, carriers, or transporters.

### Usage

```
carriers_polypep_ex_ident(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
enzymes_polypep_ex_ident(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
targets_polypep_ex_ident(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
transporters_polypep_ex_ident(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

### Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, save\_csv must be true  
**override\_csv** override existing csv, if any, in case it is true in the new parse operation  
**database\_connection** DBI connection object that holds a connection to user defined database. If save\_table is enabled without providing value for this function an error will be thrown.

## Value

a tibble with 3 variables:

**resource** Name of the source database.

**identifier** Identifier for this drug in the given resource.

**parent\_key** polypeptide id

## read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

## See Also

Other cett: [cett\\_actions\\_doc](#), [cett\\_doc](#), [cett\\_go\\_doc](#), [cett\\_poly\\_doc](#), [cett\\_poly\\_pfms\\_doc](#), [cett\\_poly\\_syn\\_doc](#)

## Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
```

```

database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)

```

---

cett_go_doc	<i>Carriers/ Enzymes/ Targets/ Transporters Polypeptide GO Classifier parsers</i>
-------------	---

---

## Description

Extract descriptions of identified polypeptide go classifier for targets, enzymes, carriers, or transporters.

## Usage

```

carriers_polypeptides_go(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

enzymes_polypeptides_go(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

targets_polypeptides_go(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

```
transporters_polypeptides_go(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

### Arguments

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 3 variables:

**category**

**description**

**parent\_key** polypeptide id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other cett: [cett\\_actions\\_doc](#), [cett\\_doc](#), [cett\\_ex\\_identity\\_doc](#), [cett\\_poly\\_doc](#), [cett\\_poly\\_pfms\\_doc](#), [cett\\_poly\\_syn\\_doc](#)

### Examples

```
## Not run:  
# the same parameters and usage will be applied for any parser  
# return only the parsed tibble  
run_all_parsers()  
  
# will throw an error, as database_connection is NULL
```

```

run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)

```

---

cett\_poly\_doc

*Carriers/ Enzymes/ Targets/ Transporters Polypeptide parsers*


---

## Description

Extract descriptions of identified polypeptide targets, enzymes, carriers, or transporters.

## Usage

```

carriers_polypeptides(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

enzymes_polypeptides(
  save_table = FALSE,
  save_csv = FALSE,

```

```

    csv_path = ".",
    override_csv = FALSE,
    database_connection = NULL
  )

  targets_polypeptides(
    save_table = FALSE,
    save_csv = FALSE,
    csv_path = ".",
    override_csv = FALSE,
    database_connection = NULL
  )

  transporters_polypeptides(
    save_table = FALSE,
    save_csv = FALSE,
    csv_path = ".",
    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

**save\_table**      boolean, save table in database if true.

**save\_csv**        boolean, save csv version of parsed tibble if true

**csv\_path**        location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv**    override existing csv, if any, in case it is true in the new parse operation

**database\_connection**  
DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 20 variables:

**id** **Universal Protein Resource (UniProt) identifier**

**source** Specifies whether the identified polypeptide ID is associated with any of the following UniProt knowledge bases: Swiss-Prot, which is manually annotated and reviewed, or TrEMBL, which is automatically annotated and not reviewed.

**name**

**general\_function** General summary of the physiological function of the polypeptide

**specific\_function** A more specific description of the polypeptide's physiological function within the cell.

**gene\_name** The short name commonly associated with the associated gene. Eg. PTGS1.

**locus** The specific chromosomal location or position of the gene's sequence on a chromosome.

**cellular\_location** The cellular location of the polypeptide.

**transmembrane\_regions** Areas of the polypeptide sequence that span a biological membrane.

**signal\_regions** Location of any signal peptides within the polypeptide sequence.

**theoretical\_pi** Theoretical isoelectric point.

**molecular\_weight** The molecular weight of the polypeptide.

**chromosome\_location** The chromosomal location of the polypeptide gene

**organism** The organism in which this polypeptide functions.

**organism\_ncbi\_taxonomy\_id**

**amino\_acid\_sequence** The amino acid sequence of the polypeptide

**amino\_acid\_format**

**gene\_sequence** The sequence of the associated gene.

**gene\_format**

**parent\_key** carrier/ target/ enzyme/ transporter id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other cett: [cett\\_actions\\_doc](#), [cett\\_doc](#), [cett\\_ex\\_identity\\_doc](#), [cett\\_go\\_doc](#), [cett\\_poly\\_pfms\\_doc](#), [cett\\_poly\\_syn\\_doc](#)

### Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
```

```
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

cett\_poly\_pfms\_doc      *Carriers/ Enzymes/ Targets/ Transporters Polypeptide PFAMS parsers*

---

## Description

Extract descriptions of identified polypeptide PFAMS targets, enzymes, carriers, or transporters.

## Usage

```
carriers_polypeptides_pfams(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

enzymes_polypeptides_pfams(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

targets_polypeptides_pfams(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```



```

)

transporters_polypeptides_pfams(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

### Arguments

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 3 variables:

**name** The sequence of the associated gene.

**identifier**

**parent\_key** polypeptide id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other cett: [cett\\_actions\\_doc](#), [cett\\_doc](#), [cett\\_ex\\_identity\\_doc](#), [cett\\_go\\_doc](#), [cett\\_poly\\_doc](#), [cett\\_poly\\_syn\\_doc](#)

### Examples

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

```

```
# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

cett\_poly\_syn\_doc      *Carriers/ Enzymes/ Targets/ Transporters Polypeptide Synonyms  
parsers*

---

## Description

Extract descriptions of identified polypeptide synonyms for targets, enzymes, carriers, or transporters.

## Usage

```
carriers_polypeptides_syn(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
enzymes_polypeptides_syn(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
targets_polypeptides_syn(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
transporters_polypeptides_syn(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

### Arguments

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 2 variables:

**synonym**

**parent\_key** polypeptide id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

## See Also

Other cett: [cett\\_actions\\_doc](#), [cett\\_doc](#), [cett\\_ex\\_identity\\_doc](#), [cett\\_go\\_doc](#), [cett\\_poly\\_doc](#), [cett\\_poly\\_pfms\\_doc](#)

## Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

dbparser

*dbparser: A package for reading and parsing DrugBank xml database.*

---

## Description

The main purpose of the ‘dbparser’ package is to parse [DrugBank](<https://www.drugbank.ca/>) database which is downloadable in XML format from [this link](<https://www.DrugBank.ca/releases/latest>).

## Details

The parsed data can then be explored and analyzed as desired by the user with the ability to save parsed data into desired database as well.

To achieve this purpose, ‘dbparser‘ package provides three main categories of functions:

- xml db reader,
- **DrugBank** elements parsers,
- and database related methods.

For more information kindly check the reference/index (<https://docs.ropensci.org/dbparser/reference/index.html>)

### xml db reader functions

Reads **DrugBank** xml database and build drug elements full tree in memory

### parsers functions

Each parser function is responsible of parsing certain drug element and returning its tibble with the ability to save it in a predefined database.

Check this tutorial (<https://docs.ropensci.org/dbparser/articles/dbparser.html>)

### database functions

To open a connection to given database in order to store parsed **DrugBank** elements database.

Check this tutorial ([https://docs.ropensci.org/dbparser/articles/Database\\_Saving.html](https://docs.ropensci.org/dbparser/articles/Database_Saving.html))

---

drugs	<i>Run all drug related parsers</i>
-------	-------------------------------------

---

## Description

Run all parsers that retrieve drugs related information

## Usage

```
drugs(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

**Arguments**

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

**Value**

a list of all drugs parsed tibbles

**read\_drugbank\_xml\_db**

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other collective\_parsers: `cett()`, `run_all_parsers()`

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)
```

```

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)

```

---

drug\_affected\_organisms

*Drug Affected Organism parser*

---

## Description

Organisms in which the drug may display activity; activity may depend on local susceptibility patterns and resistance.

## Usage

```

drug_affected_organisms(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

## Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

**Value**

a tibble with 2 variables:

**affected-organism** affected-organism name

**drugbank\_id** drugbank id

**read\_drugbank\_xml\_db**

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: `drug_ahfs_codes()`, `drug_atc_codes()`, `drug_calc_prop()`, `drug_categories()`, `drug_classification()`, `drug_dosages()`, `drug_ex_identity()`, `drug_exp_prop()`, `drug_external_links()`, `drug_food_interactions()`, `drug_general_information()`, `drug_groups()`, `drug_interactions()`, `drug_intern_brand()`, `drug_manufacturers()`, `drug_mixtures()`, `drug_packagers()`, `drug_patents()`, `drug_pdb_entries()`, `drug_pharmacology()`, `drug_prices()`, `drug_products()`, `drug_reactions_enzymes()`, `drug_reactions()`, `drug_salts()`, `drug_sequences()`, `drug_snp_adverse_reactions()`, `drug_snp_effects()`, `drug_syn()`

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)
```



```
# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_ahfs\_codes      *Drug ahfs-codes parser*

---

## Description

The American Hospital Formulary Service (AHFS) identifier for this drug.

## Usage

```
drug_ahfs_codes(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

## Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

## Value

a tibble with the following variables:

**ahfs-code**

*drugbank\_id* drugbank id

**read\_drugbank\_xml\_db**

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: `drug_affected_organisms()`, `drug_atc_codes()`, `drug_calc_prop()`, `drug_categories()`, `drug_classification()`, `drug_dosages()`, `drug_ex_identity()`, `drug_exp_prop()`, `drug_external_links()`, `drug_food_interactions()`, `drug_general_information()`, `drug_groups()`, `drug_interactions()`, `drug_intern_brand()`, `drug_manufacturers()`, `drug_mixtures()`, `drug_packagers()`, `drug_patents()`, `drug_pdb_entries()`, `drug_pharmacology()`, `drug_prices()`, `drug_products()`, `drug_reactions_enzymes()`, `drug_reactions()`, `drug_salts()`, `drug_sequences()`, `drug_snp_adverse_reactions()`, `drug_snp_effects()`, `drug_syn()`

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug_atc_codes	<i>Drug ATC Codes element parser</i>
----------------	--------------------------------------

---

### Description

The Anatomical Therapeutic Classification (ATC) code for the drug assigned by the World Health Organization Anatomical Chemical Classification System.

### Usage

```
drug_atc_codes(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

### Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

### Details

Each 'atc-code' row has one or more level. The atc-code and level> have a code the code assigned by the World Health Organization Anatomical Therapeutic Chemical Classification system.

### Value

a tibble with 10 variables

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

## Description

Drug properties that have been predicted by ChemAxon or ALOGPS based on the inputted chemical structure. Associated links below will redirect to descriptions of the specific term.

## Usage

```
drug_calc_prop(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

## Arguments

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

## Value

a tibble with 4 variables:

**kind** Name of the property.

**value** Predicted physicochemical properties; obtained by the use of prediction software such as ALOGPS and ChemAxon.

**source** Name of the software used to calculate this property, either ChemAxon or ALOGPS.

**drugbank\_id** drugbank id

## read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

**Description**

General categorizations of the drug.

**Usage**

```
drug_categories(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

**Arguments**

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

**Value**

a tibble with 2 variables:

**category** category name

**mesh-id** The Medical Subjects Headings (MeSH) identifier for the category.

**drugbank\_id** drugbank id

**read\_drugbank\_xml\_db**

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

## Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_classification    *Drug Classification parser*

---

## Description

A description of the hierarchical chemical classification of the drug; imported from ClassyFire.

## Usage

```
drug_classification(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
```



```

    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 9 variables:

**description**

**direct-parent**

**kingdom**

**superclass**

**class**

**subclass**

**alternative-parent** One or more alternative parents

**substituent** One or more substituents

**drugbank\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_dosages

*Drug Dosages parser*


---

**Description**

A list of the commercially available dosages of the drug.

**Usage**

```
drug_dosages(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
```

```

    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with the following variables:

**form** The pharmaceutical formulation by which the drug is introduced into the body

**route** The path by which the drug or product is taken into the body.

**strength** The amount of active drug ingredient provided in the dosage

**drugbank\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

### Examples

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

```

```

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)

```

---

drug\_element

*extracts the given drug elements and return data as list of tibbles.*

---

## Description

drug\_element returns list of tibbles of drugs selected elements.

## Usage

```

drug_element(
  elements_options = c("all"),
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

**Arguments**

elements_options	list, options of elements to be parsed. default is "all"
save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown. @return list of selected drug elements tibbles

**Details**

this functions extracts selected element of drug nodes in **DrugBank** xml database with the option to save it in a predefined database via passed database connection. it takes two optional arguments to save the returned tibble in the database save\_table and database\_connection. it must be called after [read\\_drugbank\\_xml\\_db](#) function like any other parser function. if [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

drug\_element\_options can be called to know the valid options for this method

**See Also**

Other common: [drug\\_element\\_options\(\)](#), [run\\_all\\_parsers\(\)](#)

**Examples**

```
## Not run:
# return only the parsed tibble
drug_element()

# will throw an error, as database_connection is NULL
drug_element(save_table = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist before read it and return its data.
drug_element(save_csv = TRUE)

sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
drug_element(save_table = TRUE, database_connection = sqlite_con)

# save in database, save parsed tibble as csv if it does not
# exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
drug_element(save_table = TRUE, save_csv = TRUE,
  database_connection = sqlite_con)
```

```
# save parsed tibble as csv if it does not exist in given location and
# return parsed tibble.
# if the csv exist before read it and return its data.
drug_element(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current
# location and return parsed tibble.
# if the csv exist override it and return it.
drug_element(save_csv = TRUE, csv_path = TRUE, override = TRUE)
drug_element(c("drug_ahfs_codes", "drug_carriers"), save_table = TRUE)
drug_element(save_table = FALSE)
drug_element(c("drug_ahfs_codes", "drug_carriers"))

## End(Not run)
```

---

drug\_element\_options *returns drug\_element valid options.*

---

### Description

returns drug\_element valid options.

### Usage

```
drug_element_options()
```

### Value

list of drug\_element valid options

### See Also

Other common: [drug\\_element\(\)](#), [run\\_all\\_parsers\(\)](#)

### Examples

```
## Not run:
drug_element_options()

## End(Not run)
```

---

`drug_exp_prop`*Drug Experimental Properties parser*

---

## Description

Drug properties that have been experimentally proven

## Usage

```
drug_exp_prop(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

## Arguments

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

## Value

a tibble with the following variables:

**kind** Name of the property.

**value** Drug properties that have been experimentally proven.

**source** Reference to the source of this experimental data.

***drugbank\_id*** drugbank id

The following experimental properties are provided:

**Water Solubility** The experimentally determined aqueous solubility of the molecule.

**Molecular Formula** Protein formula of Biotech drugs

**Molecular Weight** Protein weight of Biotech drugs.

**Melting Point** The experimentally determined temperature at which the drug molecule changes from solid to liquid at atmospheric temperature.

**Boiling Point** The experimentally determined temperature at which the drug molecule changes from liquid to gas at atmospheric temperature.

**Hydrophobicity** The ability of a molecule to repel water rather than absorb or dissolve water.

**Isoelectric Point** The pH value at which the net electric charge of a molecule is zero.

**caco2 Permeability** A continuous line of heterogenous human epithelial colorectal adenocarcinoma cells, CAC02 cells are employed as a model of human intestinal absorption of various drugs and compounds. CAC02 cell permeability is ultimately an assay to measure drug absorption.

**pKa** The experimentally determined pka value of the molecule

**logP** The experimentally determined partition coefficient (LogP) based on the ratio of solubility of the molecule in 1-octanol compared to water.

**logS** The intrinsic solubility of a given compound is the concentration in equilibrium with its solid phase that dissolves into solution, given as the natural logarithm (LogS) of the concentration.

**Radioactivity** The property to spontaneously emit particles (alpha, beta, neutron) or radiation (gamma, K capture), or both at the same time, from the decay of certain nuclides.

### read\_drugbank\_xml\_db

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: `drug_affected_organisms()`, `drug_ahfs_codes()`, `drug_atc_codes()`, `drug_calc_prop()`, `drug_categories()`, `drug_classification()`, `drug_dosages()`, `drug_ex_identity()`, `drug_external_links()`, `drug_food_interactions()`, `drug_general_information()`, `drug_groups()`, `drug_interactions()`, `drug_intern_brand()`, `drug_manufacturers()`, `drug_mixtures()`, `drug_packagers()`, `drug_patents()`, `drug_pdb_entries()`, `drug_pharmacology()`, `drug_prices()`, `drug_products()`, `drug_reactions_enzymes()`, `drug_reactions()`, `drug_salts()`, `drug_sequences()`, `drug_snp_adverse_reactions()`, `drug_snp_effects()`, `drug_syn()`

### Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
```



```
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_external\_links    *Drug External Links parser*

---

## Description

Links to other websites or databases providing information about this drug.

## Usage

```
drug_external_links(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

## Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

**Value**

a tibble with the following variables:

**resource** Name of the source website.

**identifier** Identifier for this drug in the given resource

**drugbank\_id** drugbank id

**read\_drugbank\_xml\_db**

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: `drug_affected_organisms()`, `drug_ahfs_codes()`, `drug_atc_codes()`, `drug_calc_prop()`, `drug_categories()`, `drug_classification()`, `drug_dosages()`, `drug_ex_identity()`, `drug_exp_prop()`, `drug_food_interactions()`, `drug_general_information()`, `drug_groups()`, `drug_interactions()`, `drug_intern_brand()`, `drug_manufacturers()`, `drug_mixtures()`, `drug_packagers()`, `drug_patents()`, `drug_pdb_entries()`, `drug_pharmacology()`, `drug_prices()`, `drug_products()`, `drug_reactions_enzymes()`, `drug_reactions()`, `drug_salts()`, `drug_sequences()`, `drug_snp_adverse_reactions()`, `drug_snp_effects()`, `drug_syn()`

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
```

```

# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)

```

---

drug\_ex\_identity      *Drug External Identifiers parser*

---

### Description

Identifiers used in other websites or databases providing information about this drug.

### Usage

```

drug_ex_identity(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

### Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

### Value

a tibble with the following variables:

**resource** Name of the source database.  
**identifier** Identifier for this drug in the given resource.  
**drugbank\_id** drugbank id

**read\_drugbank\_xml\_db**

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: `drug_affected_organisms()`, `drug_ahfs_codes()`, `drug_atc_codes()`, `drug_calc_prop()`, `drug_categories()`, `drug_classification()`, `drug_dosages()`, `drug_exp_prop()`, `drug_external_links()`, `drug_food_interactions()`, `drug_general_information()`, `drug_groups()`, `drug_interactions()`, `drug_intern_brand()`, `drug_manufacturers()`, `drug_mixtures()`, `drug_packagers()`, `drug_patents()`, `drug_pdb_entries()`, `drug_pharmacology()`, `drug_prices()`, `drug_products()`, `drug_reactions_enzymes()`, `drug_reactions()`, `drug_salts()`, `drug_sequences()`, `drug_snp_adverse_reactions()`, `drug_snp_effects()`, `drug_syn()`

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

`drug_food_interactions`*Drug Groups parser*

---

### Description

Food that may interact with this drug.

### Usage

```
drug_food_interactions(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

### Arguments

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

### Value

a tibble with the following variables:

#### **food-interaction**

*drugbank\_id* drugbank id

### **read\_drugbank\_xml\_db**

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: `drug_affected_organisms()`, `drug_ahfs_codes()`, `drug_atc_codes()`, `drug_calc_prop()`, `drug_categories()`, `drug_classification()`, `drug_dosages()`, `drug_ex_identity()`, `drug_exp_prop()`, `drug_external_links()`, `drug_general_information()`, `drug_groups()`, `drug_interactions()`, `drug_intern_brand()`, `drug_manufacturers()`, `drug_mixtures()`, `drug_packagers()`, `drug_patents()`, `drug_pdb_entries()`, `drug_pharmacology()`, `drug_prices()`, `drug_products()`, `drug_reactions_enzymes()`, `drug_reactions()`, `drug_salts()`, `drug_sequences()`, `drug_snp_adverse_reactions()`, `drug_snp_effects()`, `drug_syn()`

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_general\_information

*Drugs General Information parser*

---

**Description**

A description of the hierarchical chemical classification of the drug; imported from ClassyFire.

**Usage**

```
drug_general_information(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

**Arguments**

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

**Value**

a tibble with 15 variables:

**primary\_key** Drugbank id

**other\_keys** Other identifiers that may be associated with the drug

**type** Either small molecule, or biotech. Biotech is used for any drug that is derived from living systems or organisms, usually composed of high molecular weight mixtures of protein, while small molecule describes a low molecular weight organic compound.

**name**

**created** Date that this drug was first added to DrugBank.

**updated** Denotes when this drug was last updated in DrugBank.

**description** Descriptions of drug chemical properties, history and regulatory status.

**cas\_number** The Chemical Abstracts Service (CAS) registry number assigned to the drug.

**unii** Unique Ingredient Identifier (UNII) of this drug.

**average\_mass** The weighted average of the isotopic masses of the drug

**state** One of solid, liquid, or gas

**monoisotopic\_mass** The mass of the most abundant isotope of the drug

**synthesis\_reference** Citation for synthesis of the drug molecule.

**fda\_label** Contains a URL for accessing the uploaded United States Food and Drug Administration (FDA) Monograph for this drug.

**msds** Contains a URL for accessing the Material Safety Data Sheet (MSDS) for this drug.

### read\_drugbank\_xml\_db

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: `drug_affected_organisms()`, `drug_ahfs_codes()`, `drug_atc_codes()`, `drug_calc_prop()`, `drug_categories()`, `drug_classification()`, `drug_dosages()`, `drug_ex_identity()`, `drug_exp_prop()`, `drug_external_links()`, `drug_food_interactions()`, `drug_groups()`, `drug_interactions()`, `drug_intern_brand()`, `drug_manufacturers()`, `drug_mixtures()`, `drug_packagers()`, `drug_patents()`, `drug_pdb_entries()`, `drug_pharmacology()`, `drug_prices()`, `drug_products()`, `drug_reactions_enzymes()`, `drug_reactions()`, `drug_salts()`, `drug_sequences()`, `drug_snp_adverse_reactions()`, `drug_snp_effects()`, `drug_syn()`

### Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
```



```
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

 drug\_groups

*Drug Groups parser*


---

## Description

Groups that this drug belongs to. May include any of: approved, vet\_approved, nutraceutical, illicit, withdrawn, investigational, and experimental.

## Usage

```
drug_groups(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

## Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

## Value

a tibble with 2 variables:

### group

*drugbank\_id* drugbank id

## read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

## Description

Drug-drug interactions detailing drugs that, when administered concomitantly with the drug of interest, will affect its activity or result in adverse effects. These interactions may be synergistic or antagonistic depending on the physiological effects and mechanism of action of each drug.

## Usage

```
drug_interactions(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

## Arguments

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

## Value

a tibble with the following variables:

**drugbank-id** Drugbank ID of the interacting drug.

**name** Name of the interacting drug.

**description** Textual description of the physiological consequences of the drug interaction

**drugbank\_id** parent drugbank id

## read\_drugbank\_xml\_db

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

**Description**

The proprietary names used by the manufacturers for commercially available forms of the drug, focusing on brand names for products that are available in countries other than Canada and the Unites States.

**Usage**

```
drug_intern_brand(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

**Arguments**

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

**Value**

a tibble with 4 variables:

**brand** The proprietary, well-known name for given to this drug by a manufacturer.

**company** The company or manufacturer that uses this name.

**drugbank\_id** drugbank id

**read\_drugbank\_xml\_db**

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#),

```
drug_pdb_entries(), drug_pharmacology(), drug_prices(), drug_products(), drug_reactions_enzymes(),
drug_reactions(), drug_salts(), drug_sequences(), drug_snp_adverse_reactions(), drug_snp_effects(),
drug_syn()
```

### Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_manufacturers      *Drug Manufacturers parser*

---

### Description

A list of companies that are manufacturing the commercially available forms of this drug that are available in Canada and the Unites States.

**Usage**

```
drug_manufacturers(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

**Arguments**

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

**Value**

a tibble with the following variables:

**generic** A list of companies that are manufacturing the generic form of the drug.

**url** A link to the companies that are manufacturing the drug.

**drugbank\_id** drugbank id

**read\_drugbank\_xml\_db**

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_mixtures

*Drug Mixtures parser*


---

**Description**

All commercially available products in which this drug is available in combination with other drug molecules

**Usage**

```
drug_mixtures(
  save_table = FALSE,
  save_csv = FALSE,
```



```

  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 4 variables:

**name** The proprietary name provided by the manufacturer for this combination product.

**ingredients** A list of ingredients, separated by addition symbols

**supplemental-ingredients** List of additional active ingredients which are not clinically relevant to the main indication of the product, separated by addition symbols.

**drugbank\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

**Examples**

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)

```

---

drug\_packagers

*Drug Packagers parser*


---

**Description**

A list of companies that are packaging the drug for re-distribution.

**Usage**

```

drug_packagers(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",

```

```

    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 2 variables:

**name**

**url** A link to any companies that are packaging the drug for re-distribution.

**drugbank\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

### Examples

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL

```

```

run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)

```

---

drug_patents	<i>Drug Patents parser A property right issued by the United States Patent and Trademark Office (USPTO) to an inventor for a limited time, in exchange for public disclosure of the invention when the patent is granted. Drugs may be issued multiple patents.</i>
--------------	---

---

## Description

Drug Patents parser A property right issued by the United States Patent and Trademark Office (USPTO) to an inventor for a limited time, in exchange for public disclosure of the invention when the patent is granted. Drugs may be issued multiple patents.

## Usage

```

drug_patents(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

**Arguments**

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

**Value**

a tibble with the following variables:

**number** The patent number(s) associated with the drug.

**country** The country that issued the patent rights.

**approved** The date that the patent request was filed.

**expires** The date that the patent rights expire.

**pediatric-extension** Indicates whether or not a pediatric extension has been approved for the patent. Granted pediatric extensions provide an additional 6 months of market protection.

**drugbank\_id** drugbank id

**read\_drugbank\_xml\_db**

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
```

```
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_pathway

*Drug Pathway parser*

---

## Description

Metabolic, disease, and biological pathways that the drug is involved in, as identified by the Small Molecule Protein Database (SMPDB).

## Usage

```
drug_pathway(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

## Arguments

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

## Value

a tibble with the following variables:

**smpdb-id** Small Molecule Pathway Database identifier for this pathway.

**name** Pathway name

**category** Pathway category

**drugbank\_id** drugbank id

## read\_drugbank\_xml\_db

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

## See Also

Other pathway: `drug_pathway_drugs()`, `drug_pathway_enzyme()`

## Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)
```

```

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)

```

---

drug\_pathway\_drugs      *Drug Pathway Drugs parser*

---

## Description

Drugs involved in this pathway.

## Usage

```

drug_pathway_drugs(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

## Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.



**Value**

a tibble with pathway drugsproperties

**read\_drugbank\_xml\_db**

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other pathway: [drug\\_pathway\\_enzyme\(\)](#), [drug\\_pathway\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_pathway\_enzyme     *Drug Pathway Enzymes parser*

---

### Description

Enzymes involved in this pathway.

### Usage

```
drug_pathway_enzyme(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

### Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

### Value

a tibble with pathway properties

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other pathway: [drug\\_pathway\\_drugs\(\)](#), [drug\\_pathway\(\)](#)

## Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_pdb\_entries

*Drug pdb-entries parser*

---

## Description

Protein Data Bank (PDB) identifiers for this drug.

## Usage

```
drug_pdb_entries(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
```

```

    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

### Value

a tibble with the following variables:

#### **pdb-entry**

*drugbank\_id* drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

### Examples

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

```

```

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)

```

---

drug\_pharmacology      *Drug Pharmacology parser*

---

## Description

Describes the use, mechanism of action, pharmacokinetics, pharmacodynamics, and physiological or biochemical effects in the body.

## Usage

```

drug_pharmacology(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)

```

## Arguments

save\_table      boolean, save table in database if true.  
save\_csv        boolean, save csv version of parsed tibble if true

csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

## Value

a tibble with the following variables:

**indication** The approved conditions, diseases, or states for which a drug can safely and effectively be used. An indication is considered to be FDA-approved when it has any of the following designations: NDA, ANDA, BLA, or OTC. May also include indications in other countries, such as Canada (through Health Canada) or in Europe (through the European Medicines Agency).

**pharmacodynamics** A description of how the drug modifies or affects the organism it is being used in. May include effects in the body that are desired (enzyme or protein targets for example) and undesired (also known as “side effects”). This is in contrast to pharmacokinetics, which describes how the body modifies the drug being used.

**mechanism\_of\_action** A component of pharmacodynamics that describes the biochemical interaction through which a drug produces its intended effect. May include the exact molecular protein or enzyme targets and/or a description of the physiological effects produced.

**toxicity** Any adverse reaction, or side effect, that may or may not occur with use of the drug. May be attributed to a number of effects including: an enhanced therapeutic effect, rare anaphylactic reactions, interactions with other medications, or unanticipated binding of the molecule at different sites within the body.

**metabolism** A description of the chemical degradation of the drug molecule within the body; most commonly by enzymes from the Cytochrome P450 (CYP) system in the liver.

**absorption** A description of the movement of the drug from the site of administration into the bloodstream or target tissue. Common pharmacokinetic metrics used to evaluate absorption include Area Under the Curve (AUC), bioavailability (F), maximum concentration (C<sub>max</sub>), and time to maximum concentration (T<sub>max</sub>).

**half-life** The period of time it takes for the amount of drug in the body to be reduced by one half. Provides a description of how quickly the drug is being eliminated and how much is available in the bloodstream.

**protein-binding** A description of the drug’s affinity for plasma proteins and the proportion of the drug that is bound to them when in circulation within the body.

**route\_of\_elimination** A description of the pathway that is used to excrete the drug from the body. Common pharmacokinetic parameters used to evaluate excretion include elimination half life, renal clearance, and tracking of radiolabelled compounds through the renal and GI system.

**volume\_of\_distribution** The V<sub>d</sub> of a drug represents the degree to which it is distributed into body tissue compared to the plasma.

**clearance** A pharmacokinetic measurement of the rate of removal of the drug from plasma, expressed as mL/min; reflects the rate of elimination of the drug.

**drugbank\_id** drugbank id

## read\_drugbank\_xml\_db

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

## See Also

Other drugs: `drug_affected_organisms()`, `drug_ahfs_codes()`, `drug_atc_codes()`, `drug_calc_prop()`, `drug_categories()`, `drug_classification()`, `drug_dosages()`, `drug_ex_identity()`, `drug_exp_prop()`, `drug_external_links()`, `drug_food_interactions()`, `drug_general_information()`, `drug_groups()`, `drug_interactions()`, `drug_intern_brand()`, `drug_manufacturers()`, `drug_mixtures()`, `drug_packagers()`, `drug_patents()`, `drug_pdb_entries()`, `drug_prices()`, `drug_products()`, `drug_reactions_enzymes()`, `drug_reactions()`, `drug_salts()`, `drug_sequences()`, `drug_snp_adverse_reactions()`, `drug_snp_effects()`, `drug_syn()`

## Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

 drug\_prices

*Drug Prices Parsers*


---

### Description

Unit drug prices

### Usage

```
drug_prices(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

### Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 5 variables:

**description**

**cost** Drug price per unit

**currency** Currency of price, example: US.

**unit**

**parent\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.



**See Also**

Other drugs: `drug_affected_organisms()`, `drug_ahfs_codes()`, `drug_atc_codes()`, `drug_calc_prop()`, `drug_categories()`, `drug_classification()`, `drug_dosages()`, `drug_ex_identity()`, `drug_exp_prop()`, `drug_external_links()`, `drug_food_interactions()`, `drug_general_information()`, `drug_groups()`, `drug_interactions()`, `drug_intern_brand()`, `drug_manufacturers()`, `drug_mixtures()`, `drug_packagers()`, `drug_patents()`, `drug_pdb_entries()`, `drug_pharmacology()`, `drug_products()`, `drug_reactions_enzymes()`, `drug_reactions()`, `drug_salts()`, `drug_sequences()`, `drug_snp_adverse_reactions()`, `drug_snp_effects()`, `drug_syn()`

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

**Description**

A list of commercially available products in Canada and the United States that contain the drug.

**Usage**

```
drug_products(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

**Arguments**

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

**Value**

a tibble with 32 variables:

**name** The proprietary name(s) provided by the manufacturer for any commercially available products containing this drug.

**labeller** The corporation responsible for labelling this product.

**ndc-id** The National Drug Code (NDC) identifier of the drug

**ndc-product-code** The National Drug Code (NDC) product code from the FDA National Drug Code directory.

**dpd-id** Drug Product Database (DPD) identification number (a.k.a. DIN) from the Canadian Drug Product Database. Only present for drugs that are marketed in Canada

**ema-product-code** EMA product code from the European Medicines Agency Database. Only present for products that are authorised by central procedure for marketing in the European Union.

**ema-ma-number** EMA marketing authorisation number from the European Medicines Agency Database. Only present for products that are authorised by central procedure for marketing in the European Union.

**started-marketing-on** The starting date for market approval.

**ended-marketing-on** The ending date for market approval.

**dosage-form** The pharmaceutical formulation by which the drug is introduced into the body.

**strength** The amount of active drug ingredient provided in the dosage

**route** The path by which the drug or product is taken into the body

**fda-application-number** The New Drug Application [NDA] number assigned to this drug by the FDA.

**over-the-counter** A list of Over The Counter (OTC) forms of the drug.

**generic** Whether this product is a generic drug.

**approved** Indicates whether this drug has been approved by the regulating government.

**country** The country where this commercially available drug has been approved.

**source** Source of this product information. For example, a value of DPD indicates this information was retrieved from the Canadian Drug Product Database.

**standing** One of good, discordant, or deprecated. Distinguishes products with up to date ingredient information (good) from products with conflicting information (discordant) or products that have been removed from an active label (deprecated).

**standing-updated-on** The date on which the standing was last updated

**standing-reason** Explains the non-good standing of the product. One of: `ingredient_change`, `code_duplication`, `invalid`, or `removed`.

**jurisdiction-marketing-category** The marketing category of this product in its jurisdiction

**branded** Whether this product has a named brand

**prescription** Whether this product is only available with a prescription

**unapproved** Whether this product is not approved in its jurisdiction

**vaccine** Whether this product is a vaccine

**allergenic** Whether this product is used in allergenic testing

**cosmetic** Whether this product is a cosmetic, such as sunscreen

**kit** Whether this product is a kit composed of multiple distinct parts

**solo** Whether this product has only a single active ingredient

**available** Whether this product can be sold in its jurisdiction

**drugbank\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

## Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_reactions

*Drug Reactions Parsers*

---

## Description

Extract the sequential representation of the metabolic reactions that this drug molecule is involved in. Depending on available information, this may include metabolizing enzymes, reaction type, substrates, products, pharmacological activity of metabolites, and a structural representation of the biochemical reactions.

## Usage

```
drug_reactions(
```

```

    save_table = FALSE,
    save_csv = FALSE,
    csv_path = ".",
    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection**  
DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 5 variables:

**sequence** Reactions are displayed within a numerical sequence

**left\_drugbank\_name** The substrate of the reaction. Maybe a drug or a metabolite.

**right\_drugbank\_name** The product of the reaction. Maybe a drug or a metabolite.

**left\_drugbank\_id**

**right\_drugbank\_id**

**parent\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

## Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_reactions\_enzymes

*Drug Reactions Enzymes Parsers*

---

## Description

EEnzymes involved in metabolizing this drug

## Usage

```
drug_reactions_enzymes(
  save_table = FALSE,
  save_csv = FALSE,
```

```

    csv_path = ".",
    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 3 variables:

**name**

**uniprot-id**

**parent\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

### Examples

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

```

```
# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_salts

*Drug Salts parser*

---

## Description

Available salt forms of the drug. Ions such as hydrochloride, sodium, and sulfate are often added to the drug molecule to increase solubility, dissolution, or absorption.

## Usage

```
drug_salts(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```



**Arguments**

<code>save_table</code>	boolean, save table in database if true.
<code>save_csv</code>	boolean, save csv version of parsed tibble if true
<code>csv_path</code>	location to save csv files into it, default is current location, <code>save_csv</code> must be true
<code>override_csv</code>	override existing csv, if any, in case it is true in the new parse operation
<code>database_connection</code>	DBI connection object that holds a connection to user defined database. If <code>save_table</code> is enabled without providing value for this function an error will be thrown.

**Value**

a tibble with 1 variables:

**drugbank-id** DrugBank identifiers of the available salt form(s).

**name** Name of the available salt form(s)

**unii** Unique Ingredient Identifier (UNII) of the available salt form(s).

**cas-number** Chemical Abstracts Service (CAS) registry number assigned to the salt form(s) of the drug.

**inchikey** IUPAC International Chemical Identifier (InChi) key identifier for the available salt form(s).

**average-mass** Average molecular mass: the weighted average of the isotopic masses of the salt.

**monoisotopic-mass** The mass of the most abundant isotope of the salt

**smiles** The simplified molecular-input line-entry system (SMILES) is a line notation used for describing the structure of chemical species using short ASCII strings; calculated by ChemAxon.

**inchi** A prediction of the IUPAC International Chemical Identifier (InChI); imported by ChemAxon.

**formula** Indicates the simple numbers of each type of atom within the molecule; calculated by ChemAxon.

**drugbank\_id** parent drugbank id

**read\_drugbank\_xml\_db**

`read_drugbank_xml_db` function must be called first before any parser.

If `read_drugbank_xml_db` is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other drugs: `drug_affected_organisms()`, `drug_ahfs_codes()`, `drug_atc_codes()`, `drug_calc_prop()`, `drug_categories()`, `drug_classification()`, `drug_dosages()`, `drug_ex_identity()`, `drug_exp_prop()`, `drug_external_links()`, `drug_food_interactions()`, `drug_general_information()`, `drug_groups()`, `drug_interactions()`, `drug_intern_brand()`, `drug_manufacturers()`, `drug_mixtures()`, `drug_packagers()`, `drug_patents()`, `drug_pdb_entries()`, `drug_pharmacology()`, `drug_prices()`, `drug_products()`, `drug_reactions_enzymes()`, `drug_reactions()`, `drug_sequences()`, `drug_snp_adverse_reactions()`, `drug_snp_effects()`, `drug_syn()`

**Examples**

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)

```

---

drug\_sequences

*Drug Sequences parser*


---

**Description**

The amino acid sequence; provided if the drug is a peptide.

**Usage**

```

drug_sequences(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",

```

```

    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Details

Describes peptide sequences of biotech drugs

### Value

a tibble with the following variables:

**sequence** a textual representation of the sequence

**format** Currently, only the FASTA format is used

**drugbank\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

```
drug_snp_adverse_reactions
```

*Drug SNP Adverse Drug Reactions parser*

---

**Description**

The adverse drug reactions that may occur as a result of the listed single nucleotide polymorphisms (SNPs)

**Usage**

```
drug_snp_adverse_reactions(
  save_table = FALSE,
```

```

    save_csv = FALSE,
    csv_path = ".",
    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with the following variables:

**protein-name** Proteins involved in this SNP.

**gene-symbol** Genes involved in this SNP.

**uniprot-id** Universal Protein Resource (UniProt) identifiers for proteins involved in this pathway.

**rs-id** The SNP Database identifier for this single nucleotide polymorphism.

**allele** The alleles associated with the identified SNP.

**adverse-reaction**

**description**

**pubmed-id** Reference to PubMed article.

**drugbank\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_effects\(\)](#), [drug\\_syn\(\)](#)

## Examples

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

drug\_snp\_effects

*Drug SNP Effects parser*

---

## Description

A list of single nucleotide polymorphisms (SNPs) relevant to drug activity or metabolism, and the effects these may have on pharmacological activity. SNP effects in the patient may require close monitoring, an increase or decrease in dose, or a change in therapy.

## Usage

```
drug_snp_effects(
  save_table = FALSE,
```

```

    save_csv = FALSE,
    csv_path = ".",
    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with the following variables:

**protein-name** Proteins involved in this SNP.

**gene-symbol** Genes involved in this SNP.

**uniprot-id** Universal Protein Resource (UniProt) identifiers for proteins involved in this pathway.

**rs-id** The SNP Database identifier for this single nucleotide polymorphism.

**allele** The alleles associated with the identified SNP.

**defining-change**

**description** A written description of the SNP effects.

**pubmed-id** Reference to PubMed article.

**drugbank\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_syn\(\)](#)

**Examples**

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)

```

---

drug\_syn

*Drug Synonyms parser*


---

**Description**

Other names or identifiers that are associated with this drug.

**Usage**

```

drug_syn(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",

```



```

    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 3 variables:

**language** Names of the drug in languages other than English.

**coder** Organisation or source providing the synonym. For example, INN indicates the synonym is an International Nonproprietary Name, while IUPAC indicates the synonym is the nomenclature designated by the International Union of Pure and Applied Chemistry.

**drugbank\_id** drugbank id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other drugs: [drug\\_affected\\_organisms\(\)](#), [drug\\_ahfs\\_codes\(\)](#), [drug\\_atc\\_codes\(\)](#), [drug\\_calc\\_prop\(\)](#), [drug\\_categories\(\)](#), [drug\\_classification\(\)](#), [drug\\_dosages\(\)](#), [drug\\_ex\\_identity\(\)](#), [drug\\_exp\\_prop\(\)](#), [drug\\_external\\_links\(\)](#), [drug\\_food\\_interactions\(\)](#), [drug\\_general\\_information\(\)](#), [drug\\_groups\(\)](#), [drug\\_interactions\(\)](#), [drug\\_intern\\_brand\(\)](#), [drug\\_manufacturers\(\)](#), [drug\\_mixtures\(\)](#), [drug\\_packagers\(\)](#), [drug\\_patents\(\)](#), [drug\\_pdb\\_entries\(\)](#), [drug\\_pharmacology\(\)](#), [drug\\_prices\(\)](#), [drug\\_products\(\)](#), [drug\\_reactions\\_enzymes\(\)](#), [drug\\_reactions\(\)](#), [drug\\_salts\(\)](#), [drug\\_sequences\(\)](#), [drug\\_snp\\_adverse\\_reactions\(\)](#), [drug\\_snp\\_effects\(\)](#)

### Examples

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

```

```
# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

`get_drugbank_exported_date`

*Return uploaded drugbank database exported date*

---

### **Description**

`get_drugbank_exported_date` returns uploaded drugbank database exported date.

### **Usage**

```
get_drugbank_exported_date()
```

### **Value**

drugbank exported date

**Examples**

```
## Not run:  
get_drugbank_exported_date()  
  
## End(Not run)
```

---

*get\_drugbank\_metadata* *Return uploaded drugbank database metadata*

---

**Description**

*get\_drugbank\_metadata* returns uploaded drugbank database version and exported date.

**Usage**

```
get_drugbank_metadata()
```

**Value**

drugbank metadata

**Examples**

```
## Not run:  
get_drugbank_metadata()  
  
## End(Not run)
```

---

*get\_drugbank\_version* *Return uploaded drugbank database version*

---

**Description**

*get\_drugbank\_version* returns uploaded drugbank database version.

**Usage**

```
get_drugbank_version()
```

**Value**

drugbank version

**Examples**

```
## Not run:  
get_drugbank_version()  
  
## End(Not run)
```

---

links	<i>Drugs/ Carriers/ Enzymes/ Targets/ Transporters links element parser</i>
-------	---

---

**Description**

Return a list of websites that were used as references for Drugs/ Carriers/ Enzymes/ Targets/ Transporters

**Usage**

```
drugs_links(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
carriers_links(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
enzymes_links(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
targets_links(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,  
  database_connection = NULL  
)
```

```
transporters_links(  
  save_table = FALSE,  
  save_csv = FALSE,  
  csv_path = ".",  
  override_csv = FALSE,
```

```

    database_connection = NULL
  )

```

### Arguments

**save\_table** boolean, save table in database if true.

**save\_csv** boolean, save csv version of parsed tibble if true

**csv\_path** location to save csv files into it, default is current location, **save\_csv** must be true

**override\_csv** override existing csv, if any, in case it is true in the new parse operation

**database\_connection** DBI connection object that holds a connection to user defined database. If **save\_table** is enabled without providing value for this function an error will be thrown.

### Value

a tibble with 4 variables:

**ref-id** Name of the source website

**title** Identifier for this drug in the given resource

**url** The url of the website

**parent\_id** drug/ carrier/ target/ enzyme/ transporter id

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other references: [articles](#), [attachments](#), [books](#), [references\(\)](#)

### Examples

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,

```

```
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

read\_drugbank\_xml\_db *Reads **DrugBank** xml database and load it into memory.*

---

## Description

read\_drugbank\_xml\_db loads **DrugBank** xml database full tree into memory.

## Usage

```
read_drugbank_xml_db(drugbank_db_path)
```

## Arguments

drugbank\_db\_path  
**string**, full path for the **DrugBank** xml or zip file.

## Details

This functions reads **DrugBank** xml database and load it into memory for later processing. Hence; this method **must** be called before any other function in the package and it needs to be called one time only.

It takes one single mandatory argument which is the location of DrugBank db.

## Value

**TRUE** when the loading process into memory to be used by parser methods is completed successfully and **FALSE** otherwise.

**Examples**

```
## Not run:
read_drugbank_xml_db("db_full_path")
read_drugbank_xml_db(drugbank_db_path = "db_full_path")

## End(Not run)
```

---

references	<i>Drugs/ Carriers/ Enzymes/ Targets/ Transporters references element parser</i>
------------	--

---

**Description**

Return a list of all references for drugs, carriers, enzymes, targets or transporters

**Usage**

```
references(
  save_table = FALSE,
  save_csv = FALSE,
  csv_path = ".",
  override_csv = FALSE,
  database_connection = NULL
)
```

**Arguments**

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

**read\_drugbank\_xml\_db**

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

**See Also**

Other references: [articles](#), [attachments](#), [books](#), [links](#)

**Examples**

```
## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

---

run_all_parsers	<i>extracts the all drug elements and return data as list of tibbles.</i>
-----------------	---

---

**Description**

this functions extracts all element of drug nodes in **DrugBank** xml database with the option to save it in a predefined database via passed database connection. it takes two optional arguments to save the returned tibble in the database `save_table` and `database_connection`.

**Usage**

```
run_all_parsers(
  save_table = FALSE,
```



```

    save_csv = FALSE,
    csv_path = ".",
    override_csv = FALSE,
    database_connection = NULL
  )

```

### Arguments

save_table	boolean, save table in database if true.
save_csv	boolean, save csv version of parsed tibble if true
csv_path	location to save csv files into it, default is current location, save_csv must be true
override_csv	override existing csv, if any, in case it is true in the new parse operation
database_connection	DBI connection object that holds a connection to user defined database. If save_table is enabled without providing value for this function an error will be thrown.

### Value

all drug elements tibbles

### read\_drugbank\_xml\_db

[read\\_drugbank\\_xml\\_db](#) function must be called first before any parser.

If [read\\_drugbank\\_xml\\_db](#) is called before for any reason, so no need to call it again before calling this function.

### See Also

Other common: [drug\\_element\\_options\(\)](#), [drug\\_element\(\)](#)

Other collective\_parsers: [cett\(\)](#), [drugs\(\)](#)

### Examples

```

## Not run:
# the same parameters and usage will be applied for any parser
# return only the parsed tibble
run_all_parsers()

# will throw an error, as database_connection is NULL
run_all_parsers(save_table = TRUE)

# save in database in SQLite in memory database and return parsed tibble
sqlite_con <- DBI::dbConnect(RSQLite::SQLite())
run_all_parsers(save_table = TRUE, database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in current location,
# and return parsed tibble.

```

```
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE)

# save in database, save parsed tibble as csv,
# if it does not exist in current location and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_table = TRUE, save_csv = TRUE,
database_connection = sqlite_con)

# save parsed tibble as csv if it does not exist in given location,
# and return parsed tibble.
# if the csv exist before read it and return its data.
run_all_parsers(save_csv = TRUE, csv_path = TRUE)

# save parsed tibble as csv if it does not exist in current location and
# return parsed tibble.
# if the csv exist override it and return it.
run_all_parsers(save_csv = TRUE, csv_path = TRUE, override = TRUE)

## End(Not run)
```

# Index

- \* **DrugBank DB Loading**
    - read\_drugbank\_xml\_db, 102
  - \* **cett**
    - cett\_actions\_doc, 12
    - cett\_doc, 14
    - cett\_ex\_identity\_doc, 17
    - cett\_go\_doc, 19
    - cett\_poly\_doc, 21
    - cett\_poly\_pfms\_doc, 24
    - cett\_poly\_syn\_doc, 26
  - \* **collective\_parsers**
    - cett, 10
    - drugs, 29
    - run\_all\_parsers, 104
  - \* **common**
    - drug\_element, 44
    - drug\_element\_options, 46
    - run\_all\_parsers, 104
  - \* **drugs**
    - drug\_affected\_organisms, 31
    - drug\_ahfs\_codes, 33
    - drug\_atc\_codes, 35
    - drug\_calc\_prop, 36
    - drug\_categories, 38
    - drug\_classification, 40
    - drug\_dosages, 42
    - drug\_ex\_identity, 51
    - drug\_exp\_prop, 47
    - drug\_external\_links, 49
    - drug\_food\_interactions, 53
    - drug\_general\_information, 54
    - drug\_groups, 57
    - drug\_interactions, 58
    - drug\_intern\_brand, 60
    - drug\_manufacturers, 62
    - drug\_mixtures, 64
    - drug\_packagers, 66
    - drug\_patents, 68
    - drug\_pdb\_entries, 75
    - drug\_pharmacology, 77
    - drug\_prices, 80
    - drug\_products, 81
    - drug\_reactions, 84
    - drug\_reactions\_enzymes, 86
    - drug\_salts, 88
    - drug\_sequences, 90
    - drug\_snp\_adverse\_reactions, 92
    - drug\_snp\_effects, 94
    - drug\_syn, 96
  - \* **pathway**
    - drug\_pathway, 70
    - drug\_pathway\_drugs, 72
    - drug\_pathway\_enzyme, 74
  - \* **references**
    - articles, 3
    - attachments, 5
    - books, 8
    - links, 100
    - references, 103
- articles, 3, 7, 9, 101, 103
- attachments, 4, 5, 9, 101, 103
- books, 4, 7, 8, 101, 103
- carriers (cett\_doc), 14
- carriers\_actions (cett\_actions\_doc), 12
- carriers\_articles (articles), 3
- carriers\_attachments (attachments), 5
- carriers\_links (links), 100
- carriers\_polypep\_ex\_ident  
(cett\_ex\_identity\_doc), 17
- carriers\_polypeptides (cett\_poly\_doc),  
21
- carriers\_polypeptides\_go (cett\_go\_doc),  
19
- carriers\_polypeptides\_pfams  
(cett\_poly\_pfms\_doc), 24

- carriers\_polypeptides\_syn  
(cett\_poly\_syn\_doc), 26
- carriers\_textbooks (books), 8
- cett, 10, 30, 105
- cett\_actions\_doc, 12, 16, 18, 20, 23, 25, 28
- cett\_doc, 13, 14, 18, 20, 23, 25, 28
- cett\_ex\_identity\_doc, 13, 16, 17, 20, 23, 25, 28
- cett\_go\_doc, 13, 16, 18, 19, 23, 25, 28
- cett\_poly\_doc, 13, 16, 18, 20, 21, 25, 28
- cett\_poly\_pfms\_doc, 13, 16, 18, 20, 23, 24, 28
- cett\_poly\_syn\_doc, 13, 16, 18, 20, 23, 25, 26
- dbparser, 28
- drug\_affected\_organisms, 31, 34, 36, 38, 39, 41, 43, 48, 50, 52, 54, 56, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_ahfs\_codes, 32, 33, 36, 38, 39, 41, 43, 48, 50, 52, 54, 56, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_atc\_codes, 32, 34, 35, 38, 39, 41, 43, 48, 50, 52, 54, 56, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_calc\_prop, 32, 34, 36, 36, 39, 41, 43, 48, 50, 52, 54, 56, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_categories, 32, 34, 36, 38, 38, 41, 43, 48, 50, 52, 54, 56, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_classification, 32, 34, 36, 38, 39, 40, 43, 48, 50, 52, 54, 56, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_dosages, 32, 34, 36, 38, 39, 41, 42, 48, 50, 52, 54, 56, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_element, 44, 46, 105
- drug\_element\_options, 45, 46, 105
- drug\_ex\_identity, 32, 34, 36, 38, 39, 41, 43, 48, 50, 51, 54, 56, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_exp\_prop, 32, 34, 36, 38, 39, 41, 43, 47, 50, 52, 54, 56, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_external\_links, 32, 34, 36, 38, 39, 41, 43, 48, 49, 52, 54, 56, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_food\_interactions, 32, 34, 36, 38, 39, 41, 43, 48, 50, 52, 53, 56, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_general\_information, 32, 34, 36, 38, 39, 41, 43, 48, 50, 52, 54, 54, 58, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_groups, 32, 34, 36, 38, 39, 41, 43, 48, 50, 52, 54, 56, 57, 60, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_interactions, 32, 34, 36, 38, 39, 41, 43, 48, 50, 52, 54, 56, 58, 58, 61, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_intern\_brand, 32, 34, 36, 38, 39, 41, 43, 48, 50, 52, 54, 56, 58, 60, 60, 63, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_manufacturers, 32, 34, 36, 38, 39, 41, 43, 48, 50, 52, 54, 56, 58, 60, 61, 62, 65, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_mixtures, 32, 34, 36, 38, 39, 41, 43, 48, 50, 52, 54, 56, 58, 60, 61, 63, 64, 67, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_packagers, 32, 34, 36, 38, 39, 41, 43, 48, 50, 52, 54, 56, 58, 60, 61, 63, 65, 66, 69, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_patents, 32, 34, 36, 38, 39, 41, 43, 48, 50, 52, 54, 56, 58, 60, 61, 63, 65, 67, 68, 76, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97
- drug\_pathway, 70, 73, 74
- drug\_pathway\_drugs, 71, 72, 74
- drug\_pathway\_enzyme, 71, 73, 74
- drug\_pdb\_entries, 32, 34, 36, 38, 39, 41, 43,

- 48, 50, 52, 54, 56, 58, 60, 62, 63, 65,  
67, 69, 75, 79, 81, 83, 85, 87, 89, 91,  
93, 95, 97
- drug\_pharmacology, 32, 34, 36, 38, 39, 41,  
43, 48, 50, 52, 54, 56, 58, 60, 62, 63,  
65, 67, 69, 76, 77, 81, 83, 85, 87, 89,  
91, 93, 95, 97
- drug\_prices, 32, 34, 36, 38, 39, 41, 43, 48,  
50, 52, 54, 56, 58, 60, 62, 63, 65, 67,  
69, 76, 79, 80, 83, 85, 87, 89, 91, 93,  
95, 97
- drug\_products, 32, 34, 36, 38, 39, 41, 43, 48,  
50, 52, 54, 56, 58, 60, 62, 63, 65, 67,  
69, 76, 79, 81, 81, 85, 87, 89, 91, 93,  
95, 97
- drug\_reactions, 32, 34, 36, 38, 39, 41, 43,  
48, 50, 52, 54, 56, 58, 60, 62, 63, 65,  
67, 69, 76, 79, 81, 83, 84, 87, 89, 91,  
93, 95, 97
- drug\_reactions\_enzymes, 32, 34, 36, 38, 39,  
41, 43, 48, 50, 52, 54, 56, 58, 60, 62,  
63, 65, 67, 69, 76, 79, 81, 83, 85, 86,  
89, 91, 93, 95, 97
- drug\_salts, 32, 34, 36, 38, 39, 41, 43, 48, 50,  
52, 54, 56, 58, 60, 62, 63, 65, 67, 69,  
76, 79, 81, 83, 85, 87, 88, 91, 93, 95,  
97
- drug\_sequences, 32, 34, 36, 38, 39, 41, 43,  
48, 50, 52, 54, 56, 58, 60, 62, 63, 65,  
67, 69, 76, 79, 81, 83, 85, 87, 89, 90,  
93, 95, 97
- drug\_snp\_adverse\_reactions, 32, 34, 36,  
38, 39, 41, 43, 48, 50, 52, 54, 56, 58,  
60, 62, 63, 65, 67, 69, 76, 79, 81, 83,  
85, 87, 89, 91, 92, 95, 97
- drug\_snp\_effects, 32, 34, 36, 38, 39, 41, 43,  
48, 50, 52, 54, 56, 58, 60, 62, 63, 65,  
67, 69, 76, 79, 81, 83, 85, 87, 89, 91,  
93, 94, 97
- drug\_syn, 32, 34, 36, 38, 39, 41, 43, 48, 50,  
52, 54, 56, 58, 60, 62, 63, 65, 67, 69,  
76, 79, 81, 83, 85, 87, 89, 91, 93, 95,  
96
- drugs, 11, 29, 105
- drugs\_articles (articles), 3
- drugs\_attachments (attachments), 5
- drugs\_links (links), 100
- drugs\_textbooks (books), 8
- enzymes (cett\_doc), 14
- enzymes\_actions (cett\_actions\_doc), 12
- enzymes\_articles (articles), 3
- enzymes\_attachments (attachments), 5
- enzymes\_links (links), 100
- enzymes\_polypep\_ex\_ident  
(cett\_ex\_identity\_doc), 17
- enzymes\_polypeptides (cett\_poly\_doc), 21
- enzymes\_polypeptides\_go (cett\_go\_doc),  
19
- enzymes\_polypeptides\_pfams  
(cett\_poly\_pfms\_doc), 24
- enzymes\_polypeptides\_syn  
(cett\_poly\_syn\_doc), 26
- enzymes\_textbooks (books), 8
- get\_drugbank\_exported\_date, 98
- get\_drugbank\_metadata, 99
- get\_drugbank\_version, 99
- links, 4, 7, 9, 100, 103
- read\_drugbank\_xml\_db, 4, 7, 9, 11, 13, 16,  
18, 20, 23, 25, 27, 30, 32, 34, 35, 37,  
39, 41, 43, 45, 48, 50, 52, 53, 56, 57,  
59, 61, 63, 65, 67, 69, 71, 73, 74, 76,  
79, 80, 83, 85, 87, 89, 91, 93, 95, 97,  
101, 102, 103, 105
- references, 4, 7, 9, 101, 103
- run\_all\_parsers, 11, 30, 45, 46, 104
- targets (cett\_doc), 14
- targets\_actions (cett\_actions\_doc), 12
- targets\_articles (articles), 3
- targets\_attachments (attachments), 5
- targets\_links (links), 100
- targets\_polypep\_ex\_ident  
(cett\_ex\_identity\_doc), 17
- targets\_polypeptides (cett\_poly\_doc), 21
- targets\_polypeptides\_go (cett\_go\_doc),  
19
- targets\_polypeptides\_pfams  
(cett\_poly\_pfms\_doc), 24
- targets\_polypeptides\_syn  
(cett\_poly\_syn\_doc), 26
- targets\_textbooks (books), 8
- transporters (cett\_doc), 14
- transporters\_actions  
(cett\_actions\_doc), 12

transporters\_articles (articles), [3](#)  
transporters\_attachments (attachments),  
[5](#)  
transporters\_links (links), [100](#)  
transporters\_polypep\_ex\_ident  
(cett\_ex\_identity\_doc), [17](#)  
transporters\_polypeptides  
(cett\_poly\_doc), [21](#)  
transporters\_polypeptides\_go  
(cett\_go\_doc), [19](#)  
transporters\_polypeptides\_pfams  
(cett\_poly\_pfms\_doc), [24](#)  
transporters\_polypeptides\_syn  
(cett\_poly\_syn\_doc), [26](#)  
transporters\_textbooks (books), [8](#)