Package 'ctxR'

October 8, 2025

Type Package

Title Utilities for Interacting with the 'CTX' APIs

Version 1.1.3

Description Access chemical, hazard, bioactivity, and exposure data from the Computational Toxicology and Exposure ('CTX') APIs <https://www.epa.gov/comptox-tools/computational-toxicology-and-exposure-apis>. 'ctxR' was developed to streamline the process of accessing the information available through the 'CTX' APIs without requiring prior knowledge of how to use APIs. Most data is also available on the CompTox Chemical Dashboard ('CCD') <https://comptox.epa.gov/dashboard/> and other resources found at the EPA Computational Toxicology and Exposure Online Resources <https://www.epa.gov/comptox-tools>.

License GPL (>= 3)

Imports cli, data.table, httr, jsonlite, purrr, rlang, stringr, tidyr, tibble, urltools, lifecycle

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Suggests countcolors, devtools, DT, ggplot2, gridExtra, htmlTable, httptest, kableExtra, knitr, prettydoc, rmarkdown, rmdformats, testthat (>= 3.0.0), XML

URL https://github.com/USEPA/ctxR, https://usepa.github.io/ctxR/

BugReports https://github.com/USEPA/ctxR/issues

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 3.5)

NeedsCompilation no

Author Paul Kruse [aut] (ORCID: <https://orcid.org/0000-0001-5516-9717>), Caroline Ring [aut] (ORCID: <https://orcid.org/0000-0002-0463-1251>),

Madison Feshuk [ctb, cre] (ORCID:
<https://orcid.org/0000-0002-1390-6405>),
Brianna Dirks [ctb] (ORCID: <https://orcid.org/0009-0003-3691-1543>),
Carter Thunes [ctb],
Jason Brown [ctb] (ORCID: <https://orcid.org/0009-0000-2294-641X>)

Maintainer Madison Feshuk <feshuk.madison@epa.gov>

Repository CRAN

Date/Publication 2025-10-08 15:00:08 UTC

Contents

`bioactivity_api_server`

Bioactivity API Server url

Description

A section of url used in Bioactivity API Endpoints

Usage

```
bioactivity_api_server
```

Format

An object of class `character` of length 1.

---

check_api_key *Check API key*

Description

Check API key

Usage

```
check_api_key(API_key = NULL, verbose = FALSE)
```

Arguments

`API_key`	User input API key during function call.
`verbose`	A logical indicating if some "progress report" should be given.

Value

Either the API key (input during function call or loaded) or NULL (neither input during function call nor loaded).

---

check_existence_by_dtxsid *Check existence by DTXSID*

Description

Check existence by DTXSID

Usage

```
check_existence_by_dtxsid(
  DTXSID = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating whether some "progress report" should be given.

Value

A data.table with information on whether the input DTXSID is valid, and where to find more information on the chemical when the DTXSID is valid.

Examples

```
# DTXSID for bpa
bpa <- check_existence_by_dtxsid('DTXSID7020182')
# False DTXSID
false_res <- check_existence_by_Dtxsid('DTXSID7020182f')
```

---

check_existence_by_dtxsid_batch

Check existence by DTXSID batch

Description

Check existence by DTXSID batch

Usage

```
check_existence_by_dtxsid_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSIDs
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request.
Server	The root address of the API endpoint
verbose	A logical indicating whether some "progress report" should be given.

Value

A data.table of information detailing valid and invalid DTXSIDs.

Examples

```
dtxsids <- c('DTXSID7020182F', 'DTXSID7020182', 'DTXSID0020232F')
existence <- check_existence_by_dtxsid_batch(DTXSID = dtxsids)
```

---

chemical_api_server *Chemical API Server url*

Description

A section of url used in Chemical API Endpoints

Usage

```
chemical_api_server
```

Format

An object of class `character` of length 1.

---

chemical_contains *Chemical contains*

Description

Chemical contains

Usage

```
chemical_contains(
  word = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE,
  top = NULL
)
```

Arguments

word	A character string of a chemical identifier or portion of a chemical identifier. Identifiers can be a chemical name, dtxsid, dtxcid, casrn, or inchikey.
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.
top	The number of results to return if there are multiple results available

Value

A data.frame of chemicals and related values matching the query parameters

Author(s)

Paul Kruse, Kristin Issacs

Examples

```
# Pull chemicals that contain substring
substring_chemicals <- chemical_contains(word = 'TXSID702018')
```

---

chemical_contains_batch

Chemical contains batch search

Description

Chemical contains batch search

Usage

```
chemical_contains_batch(
  word_list = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE,
  top = NULL
)
```

Arguments

word_list	A list of character strings of chemical names or portion of chemical names
API_key	User-specific API key
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some "progress report" should be given.
top	The number of results to return if there are multiple results available

Value

A named list of data.frames of chemicals and related values matching the query parameters. The data.frames under the 'valid' entry contain chemical information for successful requests while the data.frames under the 'invalid' entry contain data.frames with chemical name suggestions based on the input search values.

Author(s)

Paul Kruse, Kristin Issacs

Examples

```
# Pull chemicals that contain substrings
substring_chemicals <- chemical_contains_batch(word_list = c('TXDIS702018',
                                                             'DTXSID70201'))
```

`chemical_equal` *Chemical equal*

Description

Chemical equal

Usage

```
chemical_equal(
  word = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

`word`	A character string of a chemical identifier or portion of a chemical identifier. Identifiers can be a chemical name, dtxsid, dtxcid, casrn, or inchikey.
`API_key`	The user-specific API key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A data.frame of chemicals and related values matching the query parameters

Author(s)

Paul Kruse, Kristin Issacs

Examples

```
# Pull chemicals with matching DTXSID
bpa_dtxsid <- chemical_equal(word = 'DTXSID7020182')
```

`chemical_equal_batch` *Chemical equal batch search*

Description

Chemical equal batch search

Usage

```
chemical_equal_batch(
  word_list = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

`word_list`	A list of character strings of chemical names or portion of chemical names, DTXSIDs, CASRNs, InChIKeys.
`API_key`	User-specific API key
`rate_limit`	Number of seconds to wait between each request
`verbose`	A logical indicating if some "progress report" should be given.

Value

A named list of data.tables of chemicals and related values matching the query parameters. The list contains two entries, 'valid' and 'invalid'; 'valid', contains a data.table of the results of the the searched chemical that were found in the databases; 'invalid' contains a data.table with 'suggestions' for each searched valued that did not return a chemical.

Author(s)

Paul Kruse, Kristin Issacs

Examples

```
# Pull chemicals that match input strings
bpa <- chemical_equal_batch(word_list = c('DTXSID7020182', 'DTXCID30182'))
```

chemical_starts_with *Chemical starts with*

Description

Chemical starts with

Usage

```
chemical_starts_with(
  word = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE,
  top = NULL
)
```

Arguments

word	A character string of a chemical identifier or portion of a chemical identifier. Identifiers can be a chemical name, dtxsid, dtxcid, casrn, or inchikey.
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.
top	The number of results to return if there are multiple results available

Value

A data.frame of chemicals and related values matching the query parameters

Author(s)

Paul Kruse, Kristin Issacs

Examples

```
# Pull chemicals that start with a fragment DTXSID
dtxsid_fragment <- chemical_starts_with(word = 'DTXSID702018')
```

chemical_starts_with_batch *Chemical starts with batch search*

Description

Chemical starts with batch search

Usage

```
chemical_starts_with_batch(
  word_list = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE,
  top = NULL
)
```

Arguments

word_list	A list of character strings of chemical names or portion of chemical names
API_key	User-specific API key
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some "progress report" should be given.
top	The number of results to return if there are multiple results available

Value

A named list of data.frames of chemicals and related values matching the query parameters. The data.frames under the 'valid' entry contain chemical information for successful requests while the data.frames under the 'invalid' entry contain data.frames with chemical name suggestions based on the input search values.

Author(s)

Paul Kruse, Kristin Issacs

Examples

```
# Pull chemicals that start with given substrings
bpa_substrings <- chemical_starts_with_batch(word_list = c('DTXSID702018',
                                                           'DTXCID3018'))
```

ctxR_options *ctxR Options*

Description

ctxR stores options as a named list in R's global options, i.e. `getOption('ctxR')`. It currently stores two such options, one for CCTE credentialing and one to suppress private API information in the URLs printed to the screen when web queries are placed. For both of those, see `register_ctx_api_key()`.

Usage

```
set_ctxR_option(...)

has_ctxR_options()

has_ctxR_option(option)
```

Arguments

`...`	a named listing of options to set
`option`	a specific option to query, e.g. `display_api_key`

Value

- `set_ctxR_option()` does not have a return value but has the side effect of setting options used by other functions.
- `has_ctxR_option()` returns a Boolean.
- `has_ctxR_options()` returns a Boolean.

See Also

`register_ctx_api_key()`

Examples

```
# Set ctxR options
set_ctxR_option('display_api_key' = FALSE)


# Check if there are options registered to 'ctxR'
has_ctxR_options()


# Check if a specific option is registered for 'ctxR'
has_ctxR_option('display_api_key')
```

exposure_api_server *Exposure API Server url*

Description

A section of url used in Exposure API Endpoints

Usage

```
exposure_api_server
```

Format

An object of class `character` of length 1.

get_aggregate_records_by_dtxsid *Get aggregate records by DTXSID*

Description

Get aggregate records by DTXSID

Usage

```
get_aggregate_records_by_dtxsid(
  DTXSID = NULL,
  API_key = NULL,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame of aggregate record data by DTXSID.

Examples

```
#Pull aggregate records for BPA by DTXSID
bpa_agg_records <- get_aggregate_records_by_dtxsid(DTXSID = 'DTXSID7020182')
```

`get_aggregate_records_by_dtxsid_batch` *Get Aggregate Records by DTXSID via batch*

Description

Get Aggregate Records by DTXSID via batch

Usage

```
get_aggregate_records_by_dtxsid_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

`DTXSID`	Chemical identifier DTXSID
`API_key`	The user-specific API key
`rate_limit`	Number of seconds to wait between each request
`Server`	The root address for the API endpoint
`verbose`	A logicial indicating if some "progress report" should be given.

Value

A list of data.frames containing aggregate records data for each input DTXSID.

Examples

```
# Retrieve aggregate records data for BPA and Caffeine
get_aggregate_records_by_dtxsid_batch(DTXSID = c('DTXSID0020232',
  'DTXSID7020182'))
```

`get_aggregate_records_by_medium` *Get aggregate records by medium*

Description

Get aggregate records by medium

Usage

```
get_aggregate_records_by_medium(
  Medium = NULL,
  API_key = NULL,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  pageNumber = 1,
  verbose = FALSE
)
```

Arguments

`Medium`	The mmdb medium of exposure.
`API_key`	The user-specific API key
`Server`	The root address for the API endpoint
`pageNumber`	Parameter for how to return data records.
`verbose`	A logical indicating if some "progress report" should be given.

Value

A list of search parameters and data of aggregate record data by medium.

Examples

```
#Pull aggregate records for BPA by medium
bpa_agg_records <- get_aggregate_records_by_medium(Medium = 'surface water')
```

`get_aggregate_records_by_medium_batch` *Get Aggregate Records by medium via batch*

Description

Get Aggregate Records by medium via batch

Usage

```
get_aggregate_records_by_medium_batch(
  Medium = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

`Medium`	The MMDB medium of exposure
`API_key`	The user-specific API key
`rate_limit`	Number of seconds to wait between each request
`Server`	The root address for the API endpoint
`verbose`	A logicial indicating if some "progress report" should be given.

Value

A list of data.frames containing aggregate records data for each input medium.

Examples

```
# Retrieve aggregate records data for 'surface water' and 'soil'
  get_aggregate_records_by_medium_batch(Medium = c('surface water', 'soil'))
```

get_all_assays *Retrieve all assays*

Description

Retrieve all assays

Usage

```
get_all_assays(
  API_key = NULL,
  Server = bioactivity_api_server,
  verbose = FALSE
)
```

Arguments

API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame containing all the assays and associated information

Examples

```
# Retrieve all assays
assays <- get_all_assays()
```

get_all_list_types *Get all list types*

Description

Get all list types

Usage

```
get_all_list_types(
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

API_key	The user-specific API key.
Server	The root address for the API endpoint.
verbose	A logical indicating if some "progress report" should be given.

Value

A character list of types of lists.

Examples

```
get_all_list_types()
```

---

get_all_public_chemical_lists

Get all public chemical lists

Description

Get all public chemical lists

Usage

```
get_all_public_chemical_lists(
  Projection = "",
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

Projection	Optional parameter controlling return type. It takes values chemicallistall' and 'chemicallistname' with the former as the default value.
API_key	The user-specific api key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame containing information on all public chemical lists available from the CTX chemical api.

Examples

```
# Pull all chemical lists
all_lists <- get_all_public_chemical_lists()
```

`get_annotation_by_aeid` *Retrieve annotations for AEID*

Description

Retrieve annotations for AEID

Usage

```
get_annotation_by_aeid(
  AEID = NULL,
  API_key = NULL,
  Server = bioactivity_api_server,
  verbose = FALSE
)
```

Arguments

`AEID`	The assay endpoint identifier AEID
`API_key`	The user-specific API key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A data.frame containing the annotated assays corresponding to the input AEID parameter

Examples

```
# Retrieve annotation for an assay
annotation <- get_annotation_by_aeid(AEID = 159)
```

get_annotation_by_aeid_batch *Retrieve annotations for AEID batch*

Description

Retrieve annotations for AEID batch

Usage

```
get_annotation_by_aeid_batch(
  AEID = NULL,
  API_key = NULL,
  Server = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

AEID	A list of AEID identifiers
API_key	The user-specific API key
Server	The root address for the API endpoint
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames containing annotation information for the assays with AEID matching the input parameter.

Examples

```
# Get annotations for multiple aeids
aeid_annotations <- get_annotation_by_aeid_batch(AEID = c(159, 160))
```

get_bioactivity_details *Retrieve bioactivity data from DTXSID, AEID, SPID, or m4id*

Description

Retrieve bioactivity data from DTXSID, AEID, SPID, or m4id

Usage

```
get_bioactivity_details(
  DTXSID = NULL,
  AEID = NULL,
  SPID = NULL,
  m4id = NULL,
  API_key = NULL,
  Server = bioactivity_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
AEID	The assay endpoint identifier AEID
SPID	The ChemSpider chemical input
m4id	The chemical identifier m4id
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame containing bioactivity information for the chemical or assay endpoint with identifier matching the input parameter.

Examples

```
# Pull BPA bioactivity details
bpa <- get_bioactivity_details(DTXSID = 'DTXSID7020182')
# Pull assay bioactivity details
assay <- get_bioactivity_details(AEID = 159)
```

`get_bioactivity_details_batch` *Retrieve bioactivity data from DTXSID or AEID batch*

Description

Retrieve bioactivity data from DTXSID or AEID batch

Usage

```
get_bioactivity_details_batch(
  DTXSID = NULL,
  AEID = NULL,
  SPID = NULL,
  m4id = NULL,
  API_key = NULL,
  Server = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

`DTXSID`	A list of chemical identifier DTXSIDs.
`AEID`	A list of assay endpoint identifiers AEIDs.
`SPID`	A list of ChemSpider chemical inputs
`m4id`	A list of chemical identifier m4ids
`API_key`	The user-specific API key.
`Server`	The root address for the API endpoint
`rate_limit`	Number of seconds to wait between each request
`verbose`	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames containing bioactivity information for the chemicals with DTXSID or assays with AEID matching the input parameter.

Examples

```
# Pull bioactivity details for multiple chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
batch_bioactivity <- get_bioactivity_details_batch(DTXSID = dtxsid)
# Pull bioactivity details for multiple assays
batch_bioactivity <- get_bioactivity_details_batch(AEID = c(159, 160))
```

`get_bioactivity_endpoint_status`
Bioactivity API Endpoint status

Description

Bioactivity API Endpoint status

Usage

```
get_bioactivity_endpoint_status()
```

Value

Status of Bioactivity API Endpoints

Examples

```
status <- get_bioactivity_endpoint_status()
print(status)
```

`get_bioactivity_summary`
Retrieve bioactivity summary for AEID

Description

Retrieve bioactivity summary for AEID

Usage

```
get_bioactivity_summary(
  AEID = NULL,
  API_key = NULL,
  Server = bioactivity_api_server,
  verbose = FALSE
)
```

Arguments

`AEID`	The assay endpoint indentifier AEID
`API_key`	The user-specific API key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A data.frame containing summary information corresponding to the input AEID

Examples

```
# Pull an assay bioactivity summary
aeid_1386 <- get_bioactivity_summary(AEID = 1386)
```

`get_bioactivity_summary_batch` *Retrieve bioactivity summary data from AEID batch*

Description

Retrieve bioactivity summary data from AEID batch

Usage

```
get_bioactivity_summary_batch(
  AEID = NULL,
  API_key = NULL,
  Server = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

`AEID`	A list of AEID identifiers
`API_key`	The user-specific API key.
`Server`	The root address for the API endpoint
`rate_limit`	Number of seconds to wait between each request
`verbose`	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames containing bioactivity summary information for the assays with AEID matching the input parameter.

Examples

```
# Get bioactivity summary for multiple aeids
aeids <- get_bioactivity_summary_batch(AEID = c(159, 160))
```

`get_biomonitoring_data` *Get Biomonitoring data*

Description

Get Biomonitoring data

Usage

```
get_biomonitoring_data(
  DTXSID = NULL,
  API_key = NULL,
  Projection = "",
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

`DTXSID`	The chemical identifier DTXSID
`API_key`	The user-specific API key
`Projection`	Optional parameter controlling return type.
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A data.frame containing general use keywords.

Examples

```
# Get biomonitoring data for Caffeine
get_biomonitoring_data('DTXSID0020232')
```

`get_biomonitoring_data_batch` *Get biomonitoring data via batch*

Description

Get biomonitoring data via batch

Usage

```
get_biomonitoring_data_batch(
  DTXSID = NULL,
  API_key = NULL,
  Projection = "",
  rate_limit = 0L,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

`DTXSID`	Chemical identifier DTXSID
`API_key`	The user-specific API key
`Projection`	Optional parameter controlling return type.
`rate_limit`	Number of seconds to wait between requests
`Server`	The root address for the API endpoint.
`verbose`	A logical indicating if some "progress report" should be given.

Value

A list of data.frames of biomonitoring data corresponding to the input DTXSIDs.

Examples

```
# Retrieve biomonitoring data for BPA and Caffeine
get_biomonitoring_data_batch(DTXSID = c('DTXSID7020182', 'DTXSID0020232'))
```

get_cancer_hazard *Get cancer hazard*

Description

Get cancer hazard

Usage

```
get_cancer_hazard(
  DTXSID = NULL,
  API_key = NULL,
  Server = hazard_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame of cancer hazard data related to the input DTXSID.

Examples

```
# Pull cancer hazard data for BPA
bpa_cancer <- get_cancer_hazard(DTXSID = 'DTXSID7020182')
```

get_cancer_hazard_batch *Get cancer hazard batch*

Description

Get cancer hazard batch

Usage

```
get_cancer_hazard_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = hazard_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSIDs
API_key	The user-specific API key.
rate_limit	Number of seconds to wait between requests
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table containing cancer hazard and related data for each input DTXSID.

Examples

```
# Pull cancer hazard data for multiples chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
dtxsid_cancer_hazard <- get_cancer_hazard_batch(DTXSID = dtxsid)
```

---

get_chemicals_in_list *Get chemicals in a given chemical list*

Description

Get chemicals in a given chemical list

Usage

```
get_chemicals_in_list(
  list_name = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

list_name	The name of the list of chemicals
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame of the chemical list

Examples

```
# Retrieve chemicals contained in chemical list 'CCL4'
ccl4_chemicals <- get_chemicals_in_list(list_name = 'CCL4')
```

---

get_chemicals_in_list_batch

Get chemicals in a given chemical list batch

Description

Get chemicals in a given chemical list batch

Usage

```
get_chemicals_in_list_batch(
  list_names = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

list_names	A list of names of chemical lists.
API_key	The user-specific API key.
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames each containing chemicals in the corresponding chemical lists.

Examples

```
# Pull chemicals in lists for multiple lists
chemicals_in_lists <- get_chemicals_in_list_batch(list_names = c('CCL4', 'NATADB'))
```

`get_chemicals_in_list_contain` *Get chemicals in a list specified by contained characters*

Description

Get chemicals in a list specified by contained characters

Usage

```
get_chemicals_in_list_contain(
  list_name = NULL,
  word = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

`list_name`	The name of the list to search
`word`	The contained characters to match chemicals in the given list
`API_key`	The user-specific api key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A list of DTXSIDs matching the list and search word criteria

Examples

```
bis_biosolids_2021 <- get_chemicals_in_list_contain(list_name = 'BIOSOLIDS2021',
                                                    word = 'Bis')
```

`get_chemicals_in_list_contain_batch` *Get chemicals in a list specified by characters contained batch search*

Description

Get chemicals in a list specified by characters contained batch search

Usage

```
get_chemicals_in_list_contain_batch(
  list_names = NULL,
  words = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

`list_names`	The names of the lists to search.
`words`	The search words, one for each list.
`API_key`	The user-specific API key.
`rate_limit`	Number of seconds to wait between each request.
`verbose`	A logical indicating if some "progress report" should be given.

Value

A named list of lists, with names corresponding to search terms and lists corresponding to DTXSIDs associated to the search terms

Examples

```
# Search `CCL4` for chemicals containing with 'Bis' and `BIOSOLIDS2021` for
# chemicals containing with 'Zyle'.
bis_and_zyle <- get_chemicals_in_list_contain_batch(list_names = c('CCL4',
                                                                   'BIOSOLIDS2021'),
                                                    words = c('Bis', 'Zyle'))
```

`get_chemicals_in_list_exact` *Get chemicals in a list specified by exact characters*

Description

Get chemicals in a list specified by exact characters

Usage

```
get_chemicals_in_list_exact(
  list_name = NULL,
  word = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

`list_name`	The name of the list to search
`word`	The exact characters to match chemicals in the given list
`API_key`	The user-specific api key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A list of DTXSIDs matching the list and search word criteria

Examples

```
bis_biosolids_2021 <- get_chemicals_in_list_exact(list_name = 'BIOSOLIDS2021',
                                                  word = 'Bisphenol A')
```

`get_chemicals_in_list_exact_batch` *Get chemicals in a list specified by exact characters batch search*

Description

Get chemicals in a list specified by exact characters batch search

Usage

```
get_chemicals_in_list_exact_batch(
  list_names = NULL,
  words = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

`list_names`	The names of the lists to search.
`words`	The search words, one for each list.
`API_key`	The user-specific API key.
`rate_limit`	Number of seconds to wait between each request.
`verbose`	A logical indicating if some "progress report" should be given.

Value

A named list of lists, with names corresponding to search terms and lists corresponding to DTXSIDs associated to the search terms

Examples

```
# Search `CCL4` for chemicals exactly matching with 'Bisphenol A' and
# `BIOSOLIDS2021` for chemicals exactly matching with 'Bisphenol A'.
bisphenol_a <- get_chemicals_in_list_exact_batch(list_names = c('CCL4',
                                                                'BIOSOLIDS2021'),
                                                 words = rep('Bisphenol A', 2))
```

`get_chemicals_in_list_start` *Get chemicals in a list specified by starting characters*

Description

Get chemicals in a list specified by starting characters

Usage

```
get_chemicals_in_list_start(
  list_name = NULL,
  word = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

`list_name`	The name of the list to search
`word`	The starting characters to match chemicals in the given list
`API_key`	The user-specific api key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A list of DTXSIDs matching the list and search word criteria

Examples

```
bis_biosolids_2021 <- get_chemicals_in_list_start(list_name = 'BIOSOLIDS2021',
                                                  word = 'Bi')
```

`get_chemicals_in_list_start_batch` *Get chemicals in a list specified by starting characters batch search*

Description

Get chemicals in a list specified by starting characters batch search

Usage

```
get_chemicals_in_list_start_batch(
  list_names = NULL,
  words = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

`list_names`	The names of the lists to search.
`words`	The search words, one for each list.
`API_key`	The user-specific API key.
`rate_limit`	Number of seconds to wait between each request.
`verbose`	A logical indicating if some "progress report" should be given.

Value

A named list of lists, with names corresponding to search terms and lists corresponding to DTXSIDs associated to the search terms

Examples

```
# Search `CCL4` for chemicals starting with 'Bis' and `BIOSOLIDS2021` for
# chemicals starting with 'Tri'.
bis_and_tri <- get_chemicals_in_list_start_batch(list_names = c('CCL4',
                                                                'BIOSOLIDS2021'),
                                                 words = c('Bis', 'Tri'))
```

`get_chemical_by_property_range` *Get chemicals by property and its value range*

Description

Get chemicals by property and its value range

Usage

```
get_chemical_by_property_range(
  start = NULL,
  end = NULL,
  property = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

`start`	A numeric value, the beginning of the range
`end`	A numeric value, the end of the range
`property`	A string, the property in question
`API_key`	The user-specific API key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A data.frame containing chemical information for chemicals matching the search criteria.

Examples

```
# Pull chemicals with a given property in a set range
density <- get_chemical_by_property_range(start = 1.311, end = 1.313,
                                          property = 'Density')
```

`get_chemical_by_property_range_batch` *Retrieve chemicals by property and value range in batch search*

Description

Retrieve chemicals by property and value range in batch search

Usage

```
get_chemical_by_property_range_batch(
  start_list = NULL,
  end_list = NULL,
  property_list = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

`start_list`	Numeric values, the beginning of the range
`end_list`	Numeric values, the end of the range
`property_list`	Strings, the properties being queried
`API_key`	The user-specific API key
`rate_limit`	Number of seconds to wait between each request
`verbose`	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames containing chemical information for the chemicals matching the search criteria.

Examples

```
# Pull chemicals by property ranges
prop_ranges <- get_chemical_by_property_range_batch(start_list = c(1.311,
                                                                   211.99),
                                                    end_list = c(1.313,
                                                                 212.01),
                                                    property_list = c('Density',
                                                                      'Boiling Point'))
```

get_chemical_details *Retrieve chemical details from DTXSID of DTXCID*

Description

Retrieve chemical details from DTXSID of DTXCID

Usage

```
get_chemical_details(
  DTXSID = NULL,
  DTXCID = NULL,
  Projection = "chemicaldetailstandard",
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
DTXCID	The chemical identifier DTXCID
Projection	The format and chemical detail data returned. Allowed values are 'chemicaldetailall', 'chemicaldetailstandard', 'chemicalidentifier', 'chemicalstructure', 'ntatoolkit', 'ccdchemicaldetails', 'compact'. If left empty or there is a mismatch, the default format will be 'chemicaldetailstandard'.
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table containing chemical information for the chemical with DTXSID matching the input parameter.

Examples

```
# Pull chemical details for BPA
bpa <- get_chemical_details(DTXSID = 'DTXSID7020182')
```

get_chemical_details_batch *Retrieve chemical details from DTXSID of DTXCID in batch search*

Description

Retrieve chemical details from DTXSID of DTXCID in batch search

Usage

```
get_chemical_details_batch(
  DTXSID = NULL,
  DTXCID = NULL,
  Projection = "chemicaldetailstandard",
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
DTXCID	The chemical identifier DTXCID
Projection	The format and chemical detail data returned. Allowed values are 'chemicaldetailall', 'chemicaldetailstandard', chemicalidentifier', 'chemicalstructure', 'ntatoolkit', ccdchemicaldetails', 'compact'. If left empty or there is a mismatch, the default format will be 'chemicaldetailstandard'.
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table (DTXSID) or a named list of data.tables (DTXCID) containing chemical information for the chemicals with DTXSID or DTXCID matching the input parameter.

Examples

```
# Pull chemical details for multiple chemicals by dtxsid
dtxsids <- c('DTXSID7020182', 'DTXSID2021315')
dtxsid_details <- get_chemical_details_batch(DTXSID = dtxsid)
# Pull chemical details for multiple chemicals by dtxcid
dtxcids <- c('DTXCID30182', 'DTXCID001315')
dtxcid_details <- get_chemical_details_batch(DTXCID = dtxcids)
```

get_chemical_endpoint_status *Chemical API Endpoint status*

Description

Chemical API Endpoint status

Usage

```
get_chemical_endpoint_status()
```

Value

Status of Chemical API Endpoints

Examples

```
status <- get_chemical_endpoint_status()
print(status)
```

get_chemical_image *Get image file by DTXSID or DTXCID*

Description

Get image file by DTXSID or DTXCID

Usage

```
get_chemical_image(
  DTXSID = NULL,
  DTXCID = NULL,
  gsid = NULL,
  SMILES = NULL,
  format = "",
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	Chemical identifier DTXSID
DTXCID	Chemical identifier DTXCID
gsid	DSSTox Generic Substance Identifier
SMILES	Chemical identifier SMILES
format	The image type, either "png" or "svg". If left blank, will default to "png".
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some “progress report” should be given.

Value

A Large array of three dimensions representing an image. For displaying this, one may use `png::writePNG()` or `countcolors::plotArrayAsImage()` among many such functions.

Examples

```
# Pull chemical image for BPA by dtxsid
bpa_image_matrix <- get_chemical_image(DTXSID = 'DTXSID7020182')
if (requireNamespace("countcolors", quietly = TRUE)){
  countcolors::plotArrayAsImage(bpa_image_matrix)
}
# Pull chemical image for BPA by dtxcid
bpa_image_matrix <- get_chemical_image(DTXCID = 'DTXCID30182')
if (requireNamespace("countcolors", quietly = TRUE)){
  countcolors::plotArrayAsImage(bpa_image_matrix)
}
```

---

get_chemical_image_batch

Get image file by DTXSID or DTXCID batch

Description

Get image file by DTXSID or DTXCID batch

Usage

```
get_chemical_image_batch(
  DTXSID = NULL,
  DTXCID = NULL,
  SMILES = NULL,
  format = "",
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

DTXSID	A list of chemical identifier DTXSIDs.
DTXCID	A list of chemical identifier DTXCIDs.
SMILES	A list of chemical identifier SMILES.
format	The image type, either "png" or "svg". If left blank, will default to "png".
API_key	The user-specific API key.
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some “progress report” should be given.

Value

A named list of Large arrays of three dimensions representing an image. For displaying an image, one may use `png::writePNG()` or `countcolors::plotArrayAsImage()` among many such functions.

Examples

```
# Pull images for multiple chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
images <- get_chemical_image_batch(DTXSID = dtxsid)
if (requireNamespace("countcolors", quietly = TRUE)){
  countcolors::plotArrayAsImage(images[[1]])
  countcolors::plotArrayAsImage(images[[2]])
}
```

---

get_chemical_lists_by_type

Get chemical lists by type

Description

Get chemical lists by type

Usage

```
get_chemical_lists_by_type(
  type = NULL,
  Projection = "",
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

type	The type of list. This is a case sensitive parameter and returns lists only for values of "federal", "international", "state", and "other".
Projection	Optional parameter controlling return type. It takes values 'chemicallistall' and 'chemicallistname' with the former as the default value.
API_key	The user-specified API key.
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame containing information about lists that meet the search criteria.

Examples

```
# Pull chemical lists by type
federal <- get_chemical_lists_by_type(type = 'Federal')
```

---

get_chemical_lists_by_type_batch

Get chemical lists by type batch search

Description

Get chemical lists by type batch search

Usage

```
get_chemical_lists_by_type_batch(
  type_list = NULL,
  Projection = "",
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

type_list	A list of list types. This is a case sensitive parameter and returns lists only for values of "federal", "international", "state", and "other".
Projection	Optional parameter controlling return type. It takes values 'chemicallistall' and 'chemicallistname' with the former as the default value.
API_key	The user-specified API key.
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames containing information about lists that meet the search criteria.

Examples

```
# Pull chemical lists by type
federal_state <- get_chemical_lists_by_type_batch(type_list = c('federal',
                                                                'state'))
```

`get_chemical_mol` *Get mol file by DTXSID or DTXCID*

Description

Get mol file by DTXSID or DTXCID

Usage

```
get_chemical_mol(
  DTXSID = NULL,
  DTXCID = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

`DTXSID`	Chemical identifier DTXSID
`DTXCID`	Chemical identifier DTXCID
`API_key`	The user-specific API key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A character string giving a mol file representation

Examples

```
# Pull mol file for BPA by dtxsid
bpa_mol <- get_chemical_mol(DTXSID = 'DTXSID7020182')
# Pull mol file for BPA by dtxcid
bpa_mol <- get_chemical_mol(DTXCID = 'DTXCID30182')
```

get_chemical_mol_batch *Get mol file by DTXSID or DTXCID batch*

Description

Get mol file by DTXSID or DTXCID batch

Usage

```
get_chemical_mol_batch(
  DTXSID = NULL,
  DTXCID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

DTXSID	A list of the chemical identifier DTXSIDs.
DTXCID	A list of the chemical identifier DTXCIDs.
API_key	The user-specific API key.
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of character strings giving a mol file representations of the given input chemicals.

Examples

```
# Pull mol files for multiple chemicals by DTXSID
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
mol_files <- get_chemical_mol_batch(DTXSID = dtxsid)
# Pull mol files for multiple chemicals by DTXCID
dtxcid <- c('DTXCID30182', 'DTXCID001315')
mol_files <- get_chemical_mol_batch(DTXCID = dtxcid)
```

get_chemical_mrv *Get mrv file by DTXSID or DTXCID*

Description

Get mrv file by DTXSID or DTXCID

Usage

```
get_chemical_mrv(
  DTXSID = NULL,
  DTXCID = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
DTXCID	The chemical identifier DTXCID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

XML file format for representing a mrv file.

Examples

```
# Pull mrv file for BPA by dtxsid
bpa_mrv <- get_chemical_mrv(DTXSID = 'DTXSID7020182')
# Pull mrv file for BPA by dtxcid
bpa_mrv <- getchemical_mrv(DTXCID = 'DTXCID30182')
```

get_chemical_mrv_batch *Ger mrv file by DTXSID or DTXCID batch*

Description

Ger mrv file by DTXSID or DTXCID batch

Usage

```
get_chemical_mrv_batch(
  DTXSID = NULL,
  DTXCID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

DTXSID	A list of the chemical identifier DTXSIDs.
DTXCID	A list of the chemical identifier DTXCIDs.
API_key	The user-specific API key.
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of XML file format for representing a mrv file for each chemicals.

Examples

```
# Pull mrv files for multiple chemicals by DTXSID
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
mrv_files <- get_chemical_mrv_batch(DTXSID = dtxsid)
# Pull mrv files for multiple chemicals by DTXCID
dtxcid <- c('DTXCID30182', 'DTXCID001315')
mrv_files <- get_chemical_mrv_batch(DTXCID = dtxcid)
```

get_chemical_synonym *Get chemical synonym*

Description

Get chemical synonym

Usage

```
get_chemical_synonym(
  DTXSID = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of synonym information for the input DTXSID

Examples

```
# Pull synonyms for BPA
bpa_synonym <- get_chemical_synonym(DTXSID = 'DTXSID7020182')
```

get_chemical_synonym_batch *Get chemical synonym batch*

Description

Get chemical synonym batch

Usage

```
get_chemical_synonym_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

DTXSID	A list of chemical identifier DTXSIDs
API_key	The user-specific API key.
rate_limit	The number of seconds to wait between requests.
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of lists containing synonym information for each input DTXSID.

Examples

```
# Pull synonyms for multiple chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
batch_synonyms <- get_chemical_synonym_batch(DTXSID = dtxsid)
```

---

get_chemical_weight_fraction
Get Chemical Weight Fractions

Description

Get Chemical Weight Fractions

Usage

```
get_chemical_weight_fraction(
  DTXSID = NULL,
  API_key = NULL,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame containing chemical weight fraction data.

Examples

```
# Get chemical weight fraction data for Caffeine
get_chemical_weight_fraction('DTXSID0020232')
```

---

get_chemical_weight_fraction_batch

Get chemical weight fraction via batch

Description

Get chemical weight fraction via batch

Usage

```
get_chemical_weight_fraction_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

DTXSID	Chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between requests
Server	The root address for the API endpoint.
verbose	A logical indicating if some "progress report" should be given.

Value

A list of data.frames of hemical weight fraction data corresponding to the input DTXSIDs.

Examples

```
# Retrieve chemical weight fraction data for BPA and Caffeine
get_chemical_weight_fraction_batch(DTXSID = c('DTXSID7020182',
                                              'DTXSID0020232'))
```

`get_chem_info` *Retrieve chemical information*

Description

Retrieve chemical information

Usage

```
get_chem_info(
  DTXSID = NULL,
  type = "",
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

`DTXSID`	The chemical identifier DTXSID
`type`	This specifies whether to only grab predicted or experimental results. If not specified, it will grab all details. The allowable input values are "predicted" or "experimental".
`API_key`	The user-specific API Key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some “progress report” should be given.

Value

A data.frame containing chemical information for the chemical with DTXSID matching the input parameter.

Examples

```
# Pull chemical information for BPA
bpa <- get_chem_info(DTXSID = 'DTXSID7020182')
```

get_chem_info_batch *Retrieve chemical information in batch search*

Description

Retrieve chemical information in batch search

Usage

```
get_chem_info_batch(
  DTXSID = NULL,
  type = "",
  API_key = NULL,
  rate_limit = 0L,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	A vector of chemical identifier DTXSIDs
type	A vector of type used in get_chem_info(). This specifies whether to only grab predicted or experimental results. If not specified, it will grab all details. The allowable input values are "", predicted", or "experimental".
API_key	The user-specific API key.
rate_limit	Number of seconds to wait between each request
Server	The root address for the API endpoint
verbose	A logical indicating if some “progress report” should be given.

Value

A data.table containing chemical information for the chemicals with DTXSID matching the input parameter.

Examples

```
# Pull chemical info for multiple chemicals
chem_info <- get_chem_info_batch(DTXSID = c('DTXSID7020182',
                                            'DTXSID2021315'))
```

get_chem_props_exp *Get experimental physical-chemical property data*

Description

Get experimental physical-chemical property data

Usage

```
get_chem_props_exp(
  DTXSID = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame of experimental physchem property data

Examples

```
# Get experimental physchem properties for BPA
bpa_exp_props <- get_chem_props_exp(DTXSID = 'DTXSID7020182')
```

get_chem_props_exp_batch *Get experimental physical-chemical property data via batch*

Description

Get experimental physical-chemical property data via batch

Usage

```
get_chem_props_exp_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table of experimental physchem property data

Examples

```
# Get experimental physchem properties for BPA and Caffeine
chem_props_exp <- get_chem_props_exp(DTXSID = c('DTXSID7020182',
                                                'DTXSID0020232'))
```

---

get_chem_props_pred *Get predicted physical-chemical property data*

Description

Get predicted physical-chemical property data

Usage

```
get_chem_props_pred(
  DTXSID = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame of predicted physchem property data

Examples

```
# Get predicted physchem properties for BPA
bpa_pred_props <- get_chem_props_pred(DTXSID = 'DTXSID7020182')
```

---

get_chem_props_pred_batch

Get predicted physical-chemical property data via batch

Description

Get predicted physical-chemical property data via batch

Usage

```
get_chem_props_pred_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table of predicted physchem property data

Examples

```
# Get predicted physchem properties for BPA and Caffeine
chem_props_pred <- get_chem_props_pred(DTXSID = c('DTXSID7020182',
                                                  'DTXSID0020232'))
```

`get_chem_props_summary` *Get Summary information on chemical properties*

Description

Get Summary information on chemical properties

Usage

```
get_chem_props_summary(
  DTXSID = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

`DTXSID`	The chemical identifier DTXSID.
`API_key`	The user-specific API key.
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A data.frame of summary data for chemical properties.

Examples

```
# Get summary data for BPA
bpa_props_summary <- get_chem_props_summary(DTXSID = 'DTXSID7020182')
```

get_demographic_exposure_prediction

Get demographic exposure prediction data

Description

Get demographic exposure prediction data

Usage

```
get_demographic_exposure_prediction(
  DTXSID = NULL,
  API_key = NULL,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table of demographic exposure prediction data.

Examples

```
# Pull general exposure prediction data for BPA
bpa <- get_demographic_exposure_prediction(DTXSID = 'DTXSID7020182')
```

get_demographic_exposure_prediction_batch

Retrieve demographic exposure predictions for chemicals via batch

Description

Retrieve demographic exposure predictions for chemicals via batch

Usage

```
get_demographic_exposure_prediction_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	Chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames, each containing demographic exposure prediction data for each input DTXSID.

Examples

```
# Pull demographic exposure prediction data for multiple chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
exp_demo <- get_demographic_exposure_prediction_batch(DTXSID = dtxsid)
```

---

get_exposure_endpoint_status

Exposure API Endpoint status

Description

Exposure API Endpoint status

Usage

```
get_exposure_endpoint_status()
```

Value

Status of Exposure API Endpoints

Examples

```
status <- get_exposure_endpoint_status()
print(status)
```

`get_exposure_functional_use`

Retrieve exposure related functional use data

Description

Retrieve exposure related functional use data

Usage

```
get_exposure_functional_use(
  DTXSID = NULL,
  API_key = NULL,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

`DTXSID`	Chemical identifier DTXSID
`API_key`	The user-specific API key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A data.frame of functional use data.

Examples

```
# Pull functional use data for BPA
bpa <- get_exposure_functional_use(DTXSID = 'DTXSID7020182')
```

`get_exposure_functional_use_batch` *Retrieve exposure related functional use data batch*

Description

Retrieve exposure related functional use data batch

Usage

```
get_exposure_functional_use_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

`DTXSID`	Chemical identifier DTXSID
`API_key`	The user-specific API key
`rate_limit`	Number of seconds to wait between each request
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames, each containing exposure functional use data for each input DTXSID.

Examples

```
# Pull exposure functional use data for multiple chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
dtxsid_func_use <- get_exposure_functional_use_batch(DTXSID = dtxsid)
```

get_exposure_functional_use_category *Retrieve functional use categories*

Description

Retrieve functional use categories

Usage

```
get_exposure_functional_use_category(
  API_key = NULL,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame of functional use categories.

Examples

```
# Pull functional use category data for BPA
functional_use_categories <- get_exposure_functional_use_category()
```

get_exposure_functional_use_probability *Retrieve probability of exposure for functional use category*

Description

Retrieve probability of exposure for functional use category

Usage

```
get_exposure_functional_use_probability(
  DTXSID = NULL,
  API_key = NULL,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame with probabilities corresponding to various routes of exposure related to functional use.

Examples

```
# Pull functional use probability data for BPA
bpa <- get_exposure_functional_use_probability(DTXSID = 'DTXSID7020182')
```

---

get_exposure_functional_use_probability_batch

Retrieve exposure functional use probability data batch

Description

Retrieve exposure functional use probability data batch

Usage

```
get_exposure_functional_use_probability_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	Chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table, with each row containing exposure functional use probability data for each input DTXSID. NA values are filled in for categories that have probability of 0

Examples

```
# Pull exposure functional use probability data for multiple chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
dtxsid_func_use_prob <- get_exposure_functional_use_batch(DTXSID = dtxsid)
```

---

get_exposure_list_presence_tags

Retrieve list presence tags

Description

Retrieve list presence tags

Usage

```
get_exposure_list_presence_tags(
  API_key = NULL,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame with all the list presence tags and associated data.

Examples

```
# Pull list presence tags
tags <- get_exposure_list_presence_tags()
```

get_exposure_list_presence_tags_by_dtxsid

Retrieve document data and list presence tags for a chemical

Description

Retrieve document data and list presence tags for a chemical

Usage

```
get_exposure_list_presence_tags_by_dtxsid(
  DTXSID = NULL,
  API_key = NULL,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame of document information and list presence tags

Examples

```
# Pull list presence tags for BPA
bpa <- get_exposure_list_presence_tags(DTXSID = 'DTXSID7020182')
```

get_exposure_list_presence_tags_by_dtxsid_batch

Retrieve document data and list presence tags for chemicals batch

Description

Retrieve document data and list presence tags for chemicals batch

Usage

```
get_exposure_list_presence_tags_by_dtxsid_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	Chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames, each containing exposure list presence tags use data for each input DTXSID.

Examples

```
# Pull exposure functional use data for multiple chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
exp_list_tags <- get_exposure_list_presence_tags_by_dtxsid_batch(DTXSID = dtxsid)
```

---

get_exposure_product_data

Retrieve product data for exposure purposes

Description

Retrieve product data for exposure purposes

Usage

```
get_exposure_product_data(
  DTXSID = NULL,
  API_key = NULL,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame with product information relating to exposure to the given chemical

Examples

```
# Pull exposure product data for BPA
bpa <- get_exposure_product_data(DTXSID = 'DTXSID7020182')
```

---

get_exposure_product_data_batch

Retrieve product data for exposure purposes batch

Description

Retrieve product data for exposure purposes batch

Usage

```
get_exposure_product_data_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	Chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames, each containing exposure product data for each input DTXSID.

Examples

```
# Pull exposure functional use data for multiple chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
dtxsid_product_data <- get_exposure_product_data_batch(DTXSID = dtxsid)
```

`get_exposure_product_data_puc` *Retrieve product use categories related to exposure*

Description

Retrieve product use categories related to exposure

Usage

```
get_exposure_product_data_puc(
  API_key = NULL,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

`API_key`	The user-specific API key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A data.frame consisting of all the product use categories

Examples

```
# Pull product data use categories for BPA
puc_categories <- get_exposure_product_data_puc()
```

get_fate_by_dtxsid *Get fate by DTXSID*

Description

Get fate by DTXSID

Usage

```
get_fate_by_dtxsid(
  DTXSID = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame containing chemical information for the chemical with DTXSID matching the input parameter.

Examples

```
# Pull chemical fate data for BPA
bpa <- get_fate_by_dtxsid(DTXSID = 'DTXSID7020182')
```

get_fate_by_dtxsid_batch *Retrieve chemical fate data in batch search*

Description

Retrieve chemical fate data in batch search

Usage

```
get_fate_by_dtxsid_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	A vector of chemicals identifier DTXSIDs
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table containing chemical fate information for the chemicals with DTXSID matching the input parameter.

Examples

```
# Pull chemical fate by dtxsids
chemical_fates <- get_fate_by_dtxsid_batch(DTXSID = c('DTXSID7020182',
                                                      'DTXSID2021315'))
```

---

get_general_exposure_prediction

Get general exposure prediction data

Description

Get general exposure prediction data

Usage

```
get_general_exposure_prediction(
  DTXSID = NULL,
  API_key = NULL,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table of general exposure prediction data or NULL if data is missing.

Examples

```
# Pull general exposure prediction data for BPA
bpa <- get_general_exposure_prediction(DTXSID = 'DTXSID7020182')
```

---

get_general_exposure_prediction_batch

Retrieve general exposure predictions for chemicals via batch

---

Description

Retrieve general exposure predictions for chemicals via batch

Usage

```
get_general_exposure_prediction_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	Chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
Server	The root address for the API endpoint
verbose	A logical indicating if some “progress report” should be given.

Value

A named list of data.frames, each containing general exposure prediction data for each input DTXSID.

Examples

```
# Pull general exposure prediction data for multiple chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
exp_pred <- get_general_exposure_prediction_batch(DTXSID = dtxsid)
```

`get_general_use_keywords` *Get General Use Keywords*

Description

Get General Use Keywords

Usage

```
get_general_use_keywords(
  DTXSID = NULL,
  API_key = NULL,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

`DTXSID`	The chemical identifier DTXSID
`API_key`	The user-specific API key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A data.frame containing general use keywords.

Examples

```
# Get general use keywords for Caffeine
get_general_use_keywords('DTXSID0020232')
```

`get_general_use_keywords_batch` *Get general use keywords via batch*

Description

Get general use keywords via batch

Usage

```
get_general_use_keywords_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

`DTXSID`	Chemical identifier DTXSID
`API_key`	The user-specific API key
`rate_limit`	Number of seconds to wait between requests
`Server`	The root address for the API endpoint.
`verbose`	A logical indicating if some "progress report" should be given.

Value

A list of data.frames of general use keywords corresponding to the input DTXSIDs.

Examples

```
# Retrieve general use keywords for BPA and Caffeine
get_general_use_keywords_batch(DTXSID = c('DTXSID7020182', 'DTXSID0020232'))
```

get_genetox_details *Get genetox details*

Description

Get genetox details

Usage

```
get_genetox_details(
  DTXSID = NULL,
  API_key = NULL,
  Server = hazard_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame of genetox detail data related to the input DTXSID.

Examples

```
# Pull genetox details for BPA
bpa_genetox_details <- get_genetox_details(DTXSID = 'DTXSID7020182')
```

get_genetox_details_batch *Get genetox details batch*

Description

Get genetox details batch

Usage

```
get_genetox_details_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = hazard_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSIDs
API_key	The user-specific API key.
rate_limit	Number of seconds to wait between requests
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table of genetox detail data for each input DTXSID.

Examples

```
# Pull genetox details data for multiples chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
dtxsid_genetox_details_hazard <- get_genetox_details_batch(DTXSID = dtxsid)
```

---

get_genetox_summary *Get genetox summary*

Description

Get genetox summary

Usage

```
get_genetox_summary(
  DTXSID = NULL,
  API_key = NULL,
  Server = hazard_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame of genetox summary data related to the input DTXSID.

Examples

```
# Pull genetox summary for BPA
bpa_genetox_summary <- get_genetox_summary(DTXSID = 'DTXSID7020182')
```

---

get_genetox_summary_batch

Get genetox summary batch

Description

Get genetox summary batch

Usage

```
get_genetox_summary_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = hazard_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSIDs
API_key	The user-specific API key.
rate_limit	Number of seconds to wait between requests
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table of genetox summary data for each input DTXSID.

Examples

```
# Pull genetox summary data for multiples chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
dtxsid_genetox_summary_hazard <- get_genetox_summary_batch(DTXSID = dtxsid)
```

---

get_hazard_by_dtxsid *Get hazard data by DTXSID*

Description

The function was deprecated due to updates and restructing of the CTX APIs.

Usage

```
get_hazard_by_dtxsid(
  DTXSID = NULL,
  API_key = NULL,
  Server = hazard_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame containing chemical (human and eco) hazard data

Examples

```
# Pull hazard data for BPA
bpa <- get_hazard_by_dtxsid(DTXSID = 'DTXSID7020182')
```

`get_hazard_by_dtxsid_batch` *Get hazard data by DTXSID batch*

Description

The function was deprecated due to updates and restructing of the CTX APIs.

Usage

```
get_hazard_by_dtxsid_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = hazard_api_server,
  verbose = FALSE
)
```

Arguments

`DTXSID`	A list of chemical identifier DTXSIDs
`API_key`	The user-specific API key
`rate_limit`	Number of seconds to wait between each request
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A data.table containing chemical (human and eco) hazard data for each input chemical.

Examples

```
# Pull hazard data for multiple chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
batch_hazard <- get_hazard_by_dtxsid_batch(DTXSID = dtxsid)
```

get_hazard_endpoint_status *Hazard API Endpoint status*

Description

Hazard API Endpoint status

Usage

```
get_hazard_endpoint_status()
```

Value

Status of Hazard API Endpoints

Examples

```
status <- get_hazard_endpoint_status()
print(status)
```

get_httk_data *Get httk data*

Description

Get httk data

Usage

```
get_httk_data(
  DTXSID = NULL,
  API_key = NULL,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.table of httk data for the given input chemical.

Examples

```
# Pull httk data for BPA
bpa_httk <- get_httk_data(DTXSID = 'DTXSID7020182')
```

`get_httk_data_batch` *Retrieve httk data via batch search*

Description

Retrieve httk data via batch search

Usage

```
get_httk_data_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = exposure_api_server,
  verbose = FALSE
)
```

Arguments

`DTXSID`	The chemical identifier DTXSID
`API_key`	The user-specific API key
`rate_limit`	Number of seconds to wait between each request
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A named list of httk data corresponding to the input chemicals

Examples

```
# Retrieve information for BPA and Caffeine
dtxsids <- c('DTXSID7020182', 'DTXSID0020232')
httk_data <- get_httk_data_batch(DTXSID = dtxsids)
```

get_inchi *Get InChI*

Description

Get InChI

Usage

```
get_inchi(
  name = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

name	Chemical name
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A string giving the associated inchi string.

Examples

```
bpa_inchi <- get_inchi(name = "Bisphenol A")
```

get_inchikey *Get InChIKey*

Description

Get InChIKey

Usage

```
get_inchikey(
  name = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

name	Chemical name
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A string giving the associated InChIKey.

Examples

```
bpa_inchikey <- get_inchikey(name = "Bisphenol A")
```

---

get_lists_containing_chemical

Get chemical lists containing given chemical

Description

Get chemical lists containing given chemical

Usage

```
get_lists_containing_chemical(
  DTXSID = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A list of names of chemical lists that contain the given chemical

Examples

```
# Pull chemical lists containing BPA
bpa_lists <- get_lists_containing_chemical(DTXSID = 'DTXSID7020182')
```

get_lists_containing_chemical_batch *Get chemical lists containing given chemical batch*

Description

Get chemical lists containing given chemical batch

Usage

```
get_lists_containing_chemical_batch(
  chemical_list = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

chemical_list	A list of the chemical identifier DTXSIDs.
API_key	The user-specific API key.
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of chemical lists that contain the given chemicals.

Examples

```
# Pull lists containing chemicals for multiple chemicals
lists <- get_lists_containing_chemical_batch(chemical_list = c('DTXSID7020182',
                                                               'DTXSID2021315'))
```

get_medium_categories *Retrieve MMDB medium categories*

Description

Retrieve MMDB medium categories

Usage

```
get_medium_categories(
  API_key = NULL,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame of harmonized medium categories from MMDB and relevant descriptions.

Examples

```
# Retrieve medium categories and descriptions
get_medium_categories()
```

---

get_msready_by_dtxcid *Get msready by DTXCID*

Description

Get msready by DTXCID

Usage

```
get_msready_by_dtxcid(
  DTXCID = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

DTXCID	The chemical identifier DTXCID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some “progress report” should be given.

Value

A character list of DTXSIDs with DTXCIDs matching the search criteria

Examples

```
# Pull chemicals with matching DTXCID
dtxcid_msready <- get_msready_by_dtxcid(DTXSID = 'DTXCID30182')
```

`get_msready_by_dtxcid_batch`

Get msready by DTXCID batch search

Description

Get msready by DTXCID batch search

Usage

```
get_msready_by_dtxcid_batch(
  DTXCID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

`DTXCID`	A list of chemical identifier DTXCIDs
`API_key`	A user-specific API key
`rate_limit`	Number of seconds to wait between each request
`verbose`	A logical indicating if some "progress report" should be given.

Value

A named list of character lists of DTXSIDs with DTXCIDs matching the search criteria

Examples

```
# Pull msready chemicals matching specific DTXCID
dtxcid_msready <- get_msready_by_dtxcid_batch(DTXCID = c('DTXCID30182',
                                                         'DTXCID001315'))
```

get_msready_by_formula

Get msready by formula

Description

Get msready by formula

Usage

```
get_msready_by_formula(
  formula = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

formula	A string denoting the input chemical formula
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A character list of DTXSIDs with chemical formulas matching the search criteria

Examples

```
# Pull chemicals that match input formula
mass_formula <- get_msready_by_formula(formula = 'C16H24N2O5S')
```

get_msready_by_formula_batch

Get msready by formula batch search

Description

Get msready by formula batch search

Usage

```
get_msready_by_formula_batch(
  formula_list = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

formula_list	A list of strings denoting the input chemicals formulas
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of character lists of DTXSIDs with chemical formulas matching the search criteria

Examples

```
# Pull msready data for several chemical formulas
msready_data <- get_msready_by_formula_batch(formula_list = c('C16H24N2O5S',
                                                              'C15H16O2'))
```

---

get_msready_by_mass *Get msready by mass*

Description

Get msready by mass

Usage

```
get_msready_by_mass(
  start = NULL,
  end = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

start	The starting value for mass range
end	The ending value for mass range
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A list of DTXSIDs with msready mass falling within the given range.

Examples

```
# Pull chemicals with msready mass in given range
mass_range <- get_msready_by_mass(start = 200.9, end = 200.95)
```

---

get_msready_by_mass_batch

Get ms ready by mass batch search

Description

Get ms ready by mass batch search

Usage

```
get_msready_by_mass_batch(
  start_list = NULL,
  end_list = NULL,
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

start_list	A numeric list of starting values for mass range
end_list	A numeric list of ending values for mass range
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of character lists with DTXSIDs with msready masses falling within the given ranges.

Examples

```
# Pull msready chemicals by mass ranges
msready_data <- get_msready_by_mass_batch(start_list = c(200.9, 200.95),
                                          end_list = c(200.95, 201.00))
```

`get_msready_by_mass_with_error_batch`
Get msready by mass and error offset

Description

Get msready by mass and error offset

Usage

```
get_msready_by_mass_with_error_batch(
  masses = NULL,
  error = NULL,
  API_key = NULL,
  rate_limit = 0,
  verbose = FALSE
)
```

Arguments

`masses`	A numeric list of masses.
`error`	The mass offset value.
`API_key`	The user-specific API key.
`rate_limit`	Number of seconds to wait between each request
`verbose`	A logical indicating if some "progress report" should be given.

Value

A list (of lists) of DTXSIDs, with a list returned for each input mass value.

Examples

```
#Pull chemicals by msready mass and error offset
msready_data <- get_msready_by_mass_with_error_batch(masses = c(226, 228),
                                                     error = 4)
```

get_production_volume *Get Production Volume*

Description

Get Production Volume

Usage

```
get_production_volume(
  DTXSID = NULL,
  API_key = NULL,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame containing production volume data.

Examples

```
# Get production volume data for Caffeine
get_production_volume('DTXSID0020232')
```

get_production_volume_batch *Get Production Volume data via batch*

Description

Get Production Volume data via batch

Usage

```
get_production_volume_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

DTXSID	Chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between requests
Server	The root address for the API endpoint.
verbose	A logical indicating if some "progress report" should be given.

Value

A list of data.frames of production volume data corresponding to the input DTXSIDs.

Examples

```
# Retrieve production volume data for BPA and Caffeine
get_production_volume_batch(DTXSID = c('DTXSID7020182', 'DTXSID0020232'))
```

---

get_product_use_categories_batch

Get Product Use categories via batch

Description

Get Product Use categories via batch

Usage

```
get_product_use_categories_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

DTXSID	Chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between requests
Server	The root address for the API endpoint.
verbose	A logical indicating if some "progress report" should be given.

Value

A list of data.frames of product use categories data corresponding to the input DTXSIDs.

Examples

```
# Retrieve product use categories for BPA and Caffeine
get_product_use_categories_batch(DTXSID = c('DTXSID7020182',
                                            'DTXSID0020232'))
```

---

get_product_use_category

Get Product Use Categories

Description

Get Product Use Categories

Usage

```
get_product_use_category(
  DTXSID = NULL,
  API_key = NULL,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame containing product use categories.

Examples

```
# Get product use categories for Caffeine
get_product_use_category('DTXSID0020232')
```

`get_public_chemical_list_by_name` *Get chemical list by name*

Description

Get chemical list by name

Usage

```
get_public_chemical_list_by_name(
  list_name = NULL,
  Projection = "",
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

`list_name`	The name of the list of chemicals
`Projection`	Optional parameter controlling return type. It takes values chemicallistall' and 'chemicallistname' with the former as the default value.
`API_key`	The user-specific API key
`Server`	The root address for the API endpoint
`verbose`	A logical indicating if some "progress report" should be given.

Value

A data.frame containing information about the chemical list. Note, this is not the chemical list itself. To access the chemicals in the list, use `get_chemicals_in_list`.

See Also

`get_chemicals_in_list`

Examples

```
# Pull chemical list by list name
ccl4 <- get_public_chemical_list_by_name(list_name = 'CCL4')
```

`get_public_chemical_list_by_name_batch` *Get chemical list by name batch*

Description

Get chemical list by name batch

Usage

```
get_public_chemical_list_by_name_batch(
  name_list = NULL,
  Projection = "",
  API_key = NULL,
  rate_limit = 0L,
  verbose = FALSE
)
```

Arguments

`name_list`	A list of chemical list names.
`Projection`	Optional parameter controlling return type. It takes values 'chemicallistall' and 'chemicallistname' with the former as the default value.
`API_key`	The user-specific API key.
`rate_limit`	Number of seconds to wait between each request
`verbose`	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames containing information about the chemical lists. Note, these are not the chemical lists themselves. To access the chemicals in a given list, use `get_chemicals_in_list`.

See Also

`get_chemicals_in_list`

Examples

```
# Pull chemical list information by list names
list_info <- get_public_chemical_list_by_name_batch(name_list = c('CCL4',
                                                                  'NATADB'))
```

get_reported_functional_use *Get Reported Functional Use*

Description

Get Reported Functional Use

Usage

```
get_reported_functional_use(
  DTXSID = NULL,
  API_key = NULL,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame containing reported functional use data.

Examples

```
# Get reported functional use data for Caffeine
get_reported_functional_use('DTXSID0020232')
```

get_reported_functional_use_batch *Get reported functional use via batch*

Description

Get reported functional use via batch

Usage

```
get_reported_functional_use_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

DTXSID	Chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between requests
Server	The root address for the API endpoint.
verbose	A logical indicating if some "progress report" should be given.

Value

A list of data.frames of reported functional use corresponding to the input DTXSIDs.

Examples

```
# Retrieve reported functional use for BPA and Caffeine
get_reported_functional_use_batch(DTXSID = c('DTXSID7020182',
                                             'DTXSID0020232'))
```

---

get_single_sample_records_by_dtxsid

Get single sample records by DTXSID

Description

Get single sample records by DTXSID

Usage

```
get_single_sample_records_by_dtxsid(
  DTXSID = NULL,
  API_key = NULL,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame of single sample record data by DTXSID.

Examples

```
#Pull single sample records for BPA by DTXSID
bpa_sample_records <- get_single_sample_records_by_dtxsid(DTXSID = 'DTXSID7020182')
```

---

get_single_sample_records_by_dtxsid_batch

Get Single Sample Records by DTXSID via batch

Description

Get Single Sample Records by DTXSID via batch

Usage

```
get_single_sample_records_by_dtxsid_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

DTXSID	Chemical identifier DTXSID
API_key	The user-specific API key
rate_limit	Number of seconds to wait between each request
Server	The root address for the API endpoint
verbose	A logicial indicating if some "progress report" should be given.

Value

A list of data.frames containing single sample records data for each input DTXSID.

Examples

```
# Retrieve single sample records data for BPA and Caffeine
get_single_sample_records_by_dtxsid_batch(DTXSID = c('DTXSID0020232',
  'DTXSID7020182'))
```

`get_single_sample_records_by_medium`

Get single sample records by medium

Description

Get single sample records by medium

Usage

```
get_single_sample_records_by_medium(
  Medium = NULL,
  API_key = NULL,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  pageNumber = 1,
  verbose = FALSE
)
```

Arguments

`Medium`	The mmdb medium of exposure.
`API_key`	The user-specific API key
`Server`	The root address for the API endpoint
`pageNumber`	Parameter for how to return data records.
`verbose`	A logical indicating if some "progress report" should be given.

Value

A list of search parameters and data of single sample record data by medium.

Examples

```
#Pull single records for BPA by medium
bpa_sample_records <- get_single_sample_records_by_medium(Medium = 'surface water')
```

`get_single_sample_records_by_medium_batch` *Get Single Sample Records by medium via batch*

Description

Get Single Sample Records by medium via batch

Usage

```
get_single_sample_records_by_medium_batch(
  Medium = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = "https://comptox.epa.gov/ctx-api/exposure",
  verbose = FALSE
)
```

Arguments

`Medium`	The MMDB medium of exposure
`API_key`	The user-specific API key
`rate_limit`	Number of seconds to wait between each request
`Server`	The root address for the API endpoint
`verbose`	A logicial indicating if some "progress report" should be given.

Value

A list of data.frames containing single sample records data for each input medium.

Examples

```
# Retrieve single sample records data for 'surface water' and 'soil'
get_single_sample_records_by_medium_batch(Medium = c('surface water', 'soil'))
```

get_skin_eye_hazard *Get skin and eye hazard*

Description

Get skin and eye hazard

Usage

```
get_skin_eye_hazard(
  DTXSID = NULL,
  API_key = NULL,
  Server = hazard_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSID
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A data.frame containing skin and eye hazard data.

Examples

```
# Pull skin and eye hazard data for BPA
bpa_skin_eye <- get_skin_eye_hazard_batch(DTXSID = 'DTXSID7020182')
```

get_skin_eye_hazard_batch *Get skin and eye hazard batch*

Description

Get skin and eye hazard batch

Usage

```
get_skin_eye_hazard_batch(
  DTXSID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  Server = hazard_api_server,
  verbose = FALSE
)
```

Arguments

DTXSID	The chemical identifier DTXSIDs
API_key	The user-specific API key.
rate_limit	Number of seconds to wait between each request
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A named list of data.frames containing skin and eye hazard data for each input DTXSID.

Examples

```
# Pull skin eye hazard data for multiples chemicals
dtxsid <- c('DTXSID7020182', 'DTXSID2021315')
dtxsid_skin_eye_hazard <- get_skin_eye_hazard_batch(DTXSID = dtxsid)
```

---

get_smiles *Get Smiles*

Description

Get Smiles

Usage

```
get_smiles(
  name = NULL,
  API_key = NULL,
  Server = chemical_api_server,
  verbose = FALSE
)
```

Arguments

name	Chemical name
API_key	The user-specific API key
Server	The root address for the API endpoint
verbose	A logical indicating if some "progress report" should be given.

Value

A string giving a SMILES string for the input chemical.

Examples

```
bpa_smiles <- get_smiles(name = "Bisphenol A")
```

---

hazard_api_server *Hazard API Server url*

Description

A section of url used in Hazard API Endpoints

Usage

```
hazard_api_server
```

Format

An object of class character of length 1.

---

register_ctx_api_key *Register CTX API Key for ctxR*

Description

This page contains documentation tools related to enabling CTX API services in R.

Usage

```
showing_key()

ctxR_show_api_key()

ctxR_hide_api_key()

register_ctx_api_key(key, write = FALSE)

## S3 method for class 'ctx_credentials'
print(...)

ctx_key()

has_ctx_key()
```

Arguments

key	an API key
write	if TRUE, stores the secrets provided in the .Renviron file
...	a dumped formal argument to the generic print method

Details

To obtain an API key and enable services, go to https://www.epa.gov/comptox-tools/computational-toxicology-and
This documentation shows you how to obtain an API key to allow access to the CTX APIs.

To tell ctxR about your API key, use register_ctx_api_key(), e.g. register_ctx_api_key(key = 'grbwigbwoginrowgbwibgdibdvinrginiwgo') (that's a fake key). This will set your API key for the current session, but if you restart R, you'll need to do it again. You can set it permanently by setting write = TRUEm see the examples. If you set it permanently it will be stored in a local file, and that will be accessed by ctxR persistently across sessions.

Users should be aware that the API key, a string of garbled characters/numbers/symbols, is a PRIVATE key - it uniquely identifies and authenticates you to CTX's services. If anyone gets your API key, they can use it to masquerade as you to CTX. To mitigate against users inadvertently sharing their keys, by default ctxR never displays a user's key in messages displayed to the console.

Users should be aware that ctxR has no mechanism with which to safeguard the private key once registered with R. That is to say, once you register your API key, any R function will have access to it. As a consequence, ctxR will not know if another function, potentially from a compromised package, accesses the key and uploads it to a third party. For this reason, when using ctxR we recommend a heightened sense of security and self-awareness: only use trusted packages, do not save the API keys in script files, etc.

Value

- showing_key returns a Boolean.
- ctxR_show_api_key() has no return value but has the side effect of changing the display settings of the API key.

- `ctxR_hide_api_key()` has no return value but has the side effect of changing the display settings of the API key.
- `register_ctx_api_key()` has no return value but has the side effect of storing the API key.
- `print.ctx_credentials()` has no return value and is an S3 method for printing the `ctx_credentials` class.
- `ctx_key()` returns a string, either the stored API key or `NA_character_`.
- `has_ctx_key()` returns a Boolean.

Examples

```
# Check if API key is showing
showing_key()


# Toggle API key to display
ctxR_show_api_key()


# Toggle API key to be hidden
ctxR_hide_api_key()


# Register key for this session
register_ctx_api_key(key = 'YOUR API KEY')
# Register key over sessions
register_ctx_api_key(key = 'YOUR API KEY', write = TRUE)


# Print function for ctx_credentials class
print.ctx_credentials()


# Display ctx API key
ctx_key()


# Check whether API key is registered
has_ctx_key()
```

Index