

# Package 'ctxR'

July 4, 2024

**Type** Package

**Title** Utilities for Interacting with the 'CTX' APIs

**Version** 1.0.0

**Description** Access chemical, hazard, bioactivity, and exposure data from the Computational Toxicology and Exposure ('CTX') APIs  
<<https://api-ccte.epa.gov/docs/>>. '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

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.1

**Suggests** countcolors, devtools, DT, ggplot2, gridExtra, httpptest, knitr, prettydoc, rmarkdown, testthat (>= 3.0.0), XML

**URL** <https://github.com/USEPA/ctxR>

**BugReports** <https://github.com/USEPA/ctxR/issues>

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Depends** R (>= 2.10)

**NeedsCompilation** no

**Author** Paul Kruse [aut, cre] (<<https://orcid.org/0000-0001-5516-9717>>),  
Caroline Ring [aut] (<<https://orcid.org/0000-0002-0463-1251>>),  
Madison Feshuk [ctb] (<<https://orcid.org/0000-0002-1390-6405>>),  
Carter Thunes [ctb],  
Jason Brown [ctb]

**Maintainer** Paul Kruse <kruse.paul@epa.gov>

**Repository** CRAN

**Date/Publication** 2024-07-04 16:40:02 UTC

## Contents

bioactivity_api_server . . . . .	4
chemical_api_server . . . . .	4
chemical_contains . . . . .	5
chemical_contains_batch . . . . .	5
chemical_equal . . . . .	6
chemical_equal_batch . . . . .	7
chemical_starts_with . . . . .	8
chemical_starts_with_batch . . . . .	9
create_data.table_chemical_details . . . . .	10
ctxR_options . . . . .	10
exposure_api_server . . . . .	11
get_all_assays . . . . .	12
get_all_public_chemical_lists . . . . .	12
get_annotation_by_aeid . . . . .	13
get_annotation_by_aeid_batch . . . . .	14
get_bioactivity_details . . . . .	15
get_bioactivity_details_batch . . . . .	16
get_bioactivity_endpoint_status . . . . .	17
get_bioactivity_summary . . . . .	17
get_bioactivity_summary_batch . . . . .	18
get_cancer_hazard . . . . .	19
get_cancer_hazard_batch . . . . .	19
get_cancer_hazard_batch_old . . . . .	20
get_chemicals_in_list . . . . .	21
get_chemicals_in_list_batch . . . . .	22
get_chemical_by_property_range . . . . .	22
get_chemical_by_property_range_batch . . . . .	23
get_chemical_details . . . . .	24
get_chemical_details_batch . . . . .	25
get_chemical_endpoint_status . . . . .	26
get_chemical_image . . . . .	27
get_chemical_image_batch . . . . .	28
get_chemical_lists_by_type . . . . .	29
get_chemical_lists_by_type_batch . . . . .	30
get_chemical_mol . . . . .	31
get_chemical_mol_batch . . . . .	32
get_chemical_mrv . . . . .	33
get_chemical_mrv_batch . . . . .	34
get_chemical_synonym . . . . .	35
get_chemical_synonym_batch . . . . .	35
get_chem_info . . . . .	36

get_chem_info_batch . . . . .	37
get_chem_info_batch_old . . . . .	38
get_ecotox_hazard_by_dtssid . . . . .	39
get_ecotox_hazard_by_dtssid_batch . . . . .	39
get_ecotox_hazard_by_dtssid_batch_old . . . . .	40
get_exposure_endpoint_status . . . . .	41
get_exposure_functional_use . . . . .	41
get_exposure_functional_use_batch . . . . .	42
get_exposure_functional_use_category . . . . .	43
get_exposure_functional_use_probability . . . . .	44
get_exposure_functional_use_probability_batch . . . . .	45
get_exposure_list_presence_tags . . . . .	46
get_exposure_list_presence_tags_by_dtssid . . . . .	46
get_exposure_list_presence_tags_by_dtssid_batch . . . . .	47
get_exposure_product_data . . . . .	48
get_exposure_product_data_batch . . . . .	49
get_exposure_product_data_puc . . . . .	50
get_fate_by_dtssid . . . . .	50
get_fate_by_dtssid_batch . . . . .	51
get_fate_by_dtssid_batch_old . . . . .	52
get_genetox_details . . . . .	53
get_genetox_details_batch . . . . .	53
get_genetox_details_batch_old . . . . .	54
get_genetox_summary . . . . .	55
get_genetox_summary_batch . . . . .	56
get_genetox_summary_batch_old . . . . .	57
get_hazard_by_dtssid . . . . .	57
get_hazard_by_dtssid_batch . . . . .	58
get_hazard_by_dtssid_batch_old . . . . .	59
get_hazard_endpoint_status . . . . .	60
get_human_hazard_by_dtssid . . . . .	60
get_human_hazard_by_dtssid_batch . . . . .	61
get_human_hazard_by_dtssid_batch_old . . . . .	62
get_inchi . . . . .	62
get_inchikey . . . . .	63
get_lists_containing_chemical . . . . .	64
get_lists_containing_chemical_batch . . . . .	64
get_msready_by_dtscid . . . . .	65
get_msready_by_dtscid_batch . . . . .	66
get_msready_by_formula . . . . .	67
get_msready_by_formula_batch . . . . .	67
get_msready_by_mass . . . . .	68
get_msready_by_mass_batch . . . . .	69
get_msready_by_mass_with_error_batch . . . . .	70
get_public_chemical_list_by_name . . . . .	71
get_public_chemical_list_by_name_batch . . . . .	72
get_skin_eye_hazard . . . . .	73
get_skin_eye_hazard_batch . . . . .	73

get_skin_eye_hazard_batch_old . . . . .	74
get_smiles . . . . .	75
hazard_api_server . . . . .	76
prepare_word . . . . .	76
register_ctx_api_key . . . . .	77

<b>Index</b>	<b>79</b>
--------------	-----------

---

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.

---

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 name or portion of a 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.
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

**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 = verbose,  
  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

**Examples**

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

---

chemical_equal	<i>Chemical equal</i>
----------------	-----------------------

---

**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 name or portion of a 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 data.frame of chemicals and related values matching the query parameters

**Examples**

```
# Pull chemicals with matching DTXSID
bpa_dt xsid <- 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.frames of chemicals and related values matching the query parameters

## 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 name or portion of a 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.
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

## Examples

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





---

```
create_data.table_chemical_details
```

*Create chemical details data.table helper function*

---

### Description

Create chemical details data.table helper function

### Usage

```
create_data.table_chemical_details(index = -1)
```

### Arguments

index            Determine which format should be used.

### Value

An empty data.table with columns matching the expected format of the get\_chemical\_details API call.

---

```
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')
```

---

<code>exposure_api_server</code>	<i>Exposure API Server url</i>
----------------------------------	--------------------------------

---

**Description**

A section of url used in Exposure API Endpoints

**Usage**

```
exposure_api_server
```

**Format**

An object of class character of length 1.

---

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\_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 auids
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 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 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\_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_cancer_hazard_batch_old  
  Get cancer hazard batch
```

---

**Description**

Get cancer hazard batch

**Usage**

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

**Arguments**

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

**Value**

A named list of data.frames, each containing cancer hazard and related data for each input 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**

<code>list_names</code>	A list of names of chemical lists.
<code>API_key</code>	The user-specific API key.
<code>rate_limit</code>	Number of seconds to wait between each request
<code>verbose</code>	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_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', 'nta-toolkit', 'ccdchemicaldetails'. 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'. 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 = dtxsids)
# 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,  
  SMILES = NULL,  
  format = "",  
  API_key = NULL,  
  Server = chemical_api_server,  
  verbose = FALSE  
)
```

### Arguments

DTXSID	Chemical identifier DTXSID
DTXCID	Chemical identifier DTXCID
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

<code>type_list</code>	A list of list types. This is a case sensitive parameter and returns lists only for values of "federal", "international", "state", and "other".
<code>Projection</code>	Optional parameter controlling return type. It takes values 'chemicallistall' and 'chemicallistname' with the former as the default value.
<code>API_key</code>	The user-specified API key.
<code>rate_limit</code>	Number of seconds to wait between each request
<code>verbose</code>	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	<i>Get mol file by DTXSID or DTXCID</i>
------------------	---

---

## 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	<i>Get mrv file by DTXSID or DTXCID</i>
------------------	---

---

## 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_chem_info	<i>Retrieve chemical information</i>
---------------	--------------------------------------

---

**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_info_batch_old
```

*Retrieve chemical information in batch search*

---

**Description**

Retrieve chemical information in batch search

**Usage**

```
get_chem_info_batch_old(
  DTxsID = NULL,
  type = "",
  API_key = NULL,
  rate_limit = 0L,
  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
verbose	A logical indicating if some “progress report” should be given.

**Value**

A named list of data.frames containing chemical information for the chemicals with DTxsID matching the input parameter.

---

`get_ecotox_hazard_by_dtksid`*Get ecotox hazard data by DTXSID*

---

**Description**

Get ecotox hazard data by DTXSID

**Usage**

```
get_ecotox_hazard_by_dtksid(  
  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 (ecotox) hazard data

**Examples**

```
# Pull ecotox hazard data for BPA  
bpa_ecotox <- get_ecotox_hazard_by_dtksid(DTXSID = 'DTXSID7020182')
```

---

`get_ecotox_hazard_by_dtksid_batch`*Get ecotox hazard data by DTXSID batch*

---

**Description**

Get ecotox hazard data by DTXSID batch

**Usage**

```
get_ecotox_hazard_by_dtksid_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 ecotox hazard data.

**Examples**

```
# Pull ecotox hazard data for multiples chemicals  
dtksid <- c('DTXSID7020182', 'DTXSID2021315')  
dtksid_ecotox_hazard <- get_ecotox_hazard_by_dtksid_batch(DTxsID = dtksid)
```

---

```
get_ecotox_hazard_by_dtksid_batch_old
```

*Get ecotox hazard data by DTxsID batch*

---

**Description**

Get ecotox hazard data by DTxsID batch

**Usage**

```
get_ecotox_hazard_by_dtksid_batch_old(  
  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        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 ecotox hazard data.

---

*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**

```
has_ctx_key() & is.na(ctx_key() == 'FAKE_KEY')
# 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

```
has_ctx_key() & is.na(ctx_key() == 'FAKE_KEY')  
# 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_dtksid_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_dtksid_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

```
has_ctx_key() & is.na(ctx_key()) == 'FAKE_KEY')
# 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

```
has_ctx_key() & is.na(ctx_key() == 'FAKE_KEY')  
# 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_dtksid
```

*Get fate by DTXSID*

---

### **Description**

Get fate by DTXSID

### **Usage**

```
get_fate_by_dtksid(  
  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_dtksid(DTXSID = 'DTXSID7020182')
```

---

```
get_fate_by_dtksid_batch
```

*Retrieve chemical fate data in batch search*

---

**Description**

Retrieve chemical fate data in batch search

**Usage**

```
get_fate_by_dtksid_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 dtksids
chemical_fates <- get_fate_by_dtksid_batch(DTxsID = c('DTxsID7020182',
                                                    'DTxsID2021315'))
```

---

get\_fate\_by\_dtksid\_batch\_old

*Retrieve chemical fate data in batch search*

---

**Description**

Retrieve chemical fate data in batch search

**Usage**

```
get_fate_by_dtksid_batch_old(
  DTxsID = NULL,
  API_key = NULL,
  rate_limit = 0L,
  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
verbose	A logical indicating if some “progress report” should be given.

**Value**

A named list of data.frames containing chemical fate information for the chemicals with DTxsID matching the input parameter.

---

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_details_batch_old  
  Get genetox details batch
```

---

**Description**

Get genetox details batch

**Usage**

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

**Arguments**

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

**Value**

A named list of data.frames of genetox detail data for each input 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\_genetox\_summary\_batch\_old  
*Get genetox summary batch*

---

**Description**

Get genetox summary batch

**Usage**

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

**Arguments**

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

**Value**

A named list of data.frames of genetox summary data for each input DTXSID.

---

get\_hazard\_by\_dtgsid *Get hazard data by DTGSID*

---

**Description**

Get hazard data by DTGSID

**Usage**

```
get_hazard_by_dtgsid(  
  DTGSID = 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_dtksid(DTXSID = 'DTXSID7020182')
```

---

get\_hazard\_by\_dtksid\_batch

*Get hazard data by DTXSID batch*

---

**Description**

Get hazard data by DTXSID batch

**Usage**

```
get_hazard_by_dtksid_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
dtksid <- c('DTXSID7020182', 'DTXSID2021315')
batch_hazard <- get_hazard_by_dtksid_batch(DTXSID = dtksid)
```

---

```
get_hazard_by_dtksid_batch_old
```

*Get hazard data by DTXSID batch*

---

## Description

Get hazard data by DTXSID batch

## Usage

```
get_hazard_by_dtksid_batch_old(  
  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	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 (human and eco) hazard data for each input chemical.

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\_human\_hazard\_by\_dtssid  
*Get human hazard data by DTXSID*

---

**Description**

Get human hazard data by DTXSID

**Usage**

```
get_human_hazard_by_dtssid(  
  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 hazard data

**Examples**

```
# Pull human hazard data for BPA
bpa_human <- get_human_hazard_by_dtssid(DTXSID = 'DTXSID7020182')
```

---

```
get_human_hazard_by_dtssid_batch
Get human hazard data by DTXSID batch
```

---

**Description**

Get human hazard data by DTXSID batch

**Usage**

```
get_human_hazard_by_dtssid_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 hazard data.

**Examples**

```
# Pull human hazard data for multiples chemicals
dtssid <- c('DTXSID7020182', 'DTXSID2021315')
dtssid_human_hazard <- get_human_hazard_by_dtssid_batch(DTXSID = dtssid)
```

---

get\_human\_hazard\_by\_dtxsid\_batch\_old

*Get human hazard data by DTXSID batch*

---

### Description

Get human hazard data by DTXSID batch

### Usage

```
get_human_hazard_by_dtxsid_batch_old(  
    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	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 human hazard data.

---

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\_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.



---

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\_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_skin_eye_hazard_batch_old  
Get skin and eye hazard batch
```

---

**Description**

Get skin and eye hazard batch

**Usage**

```
get_skin_eye_hazard_batch_old(  
  DTXSID = NULL,  
  API_key = NULL,  
  rate_limit = 0L,  
  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
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.

---

get_smiles	<i>Get Smiles</i>
------------	-------------------

---

**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	<i>Hazard API Server url</i>
-------------------	------------------------------

---

**Description**

A section of url used in Hazard API Endpoints

**Usage**

hazard\_api\_server

**Format**

An object of class character of length 1.

---

prepare_word	<i>Prepare url helper function</i>
--------------	------------------------------------

---

**Description**

Prepare url helper function

**Usage**

prepare\_word(word)

**Arguments**

word	A character string
------	--------------------

**Value**

A character string that is ready for use in http request

---

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 = 'grbwigbwoginrowgbwibgdibdviniwgo')` (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 = TRUE` 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

## \* datasets

- bioactivity\_api\_server, 4
  - chemical\_api\_server, 4
  - exposure\_api\_server, 11
  - hazard\_api\_server, 76
- bioactivity\_api\_server, 4
- chemical\_api\_server, 4
- chemical\_contains, 5
- chemical\_contains\_batch, 5
- chemical\_equal, 6
- chemical\_equal\_batch, 7
- chemical\_starts\_with, 8
- chemical\_starts\_with\_batch, 9
- create\_data.table\_chemical\_details, 10
- ctx\_key (register\_ctx\_api\_key), 77
- ctxR\_hide\_api\_key  
(register\_ctx\_api\_key), 77
- ctxR\_options, 10
- ctxR\_show\_api\_key  
(register\_ctx\_api\_key), 77
- exposure\_api\_server, 11
- get\_all\_assays, 12
- get\_all\_public\_chemical\_lists, 12
- get\_annotation\_by\_aeid, 13
- get\_annotation\_by\_aeid\_batch, 14
- get\_bioactivity\_details, 15
- get\_bioactivity\_details\_batch, 16
- get\_bioactivity\_endpoint\_status, 17
- get\_bioactivity\_summary, 17
- get\_bioactivity\_summary\_batch, 18
- get\_cancer\_hazard, 19
- get\_cancer\_hazard\_batch, 19
- get\_cancer\_hazard\_batch\_old, 20
- get\_chem\_info, 36
- get\_chem\_info\_batch, 37
- get\_chem\_info\_batch\_old, 38
- get\_chemical\_by\_property\_range, 22
- get\_chemical\_by\_property\_range\_batch, 23
- get\_chemical\_details, 24
- get\_chemical\_details\_batch, 25
- get\_chemical\_endpoint\_status, 26
- get\_chemical\_image, 27
- get\_chemical\_image\_batch, 28
- get\_chemical\_lists\_by\_type, 29
- get\_chemical\_lists\_by\_type\_batch, 30
- get\_chemical\_mol, 31
- get\_chemical\_mol\_batch, 32
- get\_chemical\_mrv, 33
- get\_chemical\_mrv\_batch, 34
- get\_chemical\_synonym, 35
- get\_chemical\_synonym\_batch, 35
- get\_chemicals\_in\_list, 21, 71, 72
- get\_chemicals\_in\_list\_batch, 22
- get\_ecotox\_hazard\_by\_dt\_xsid, 39
- get\_ecotox\_hazard\_by\_dt\_xsid\_batch, 39
- get\_ecotox\_hazard\_by\_dt\_xsid\_batch\_old, 40
- get\_exposure\_endpoint\_status, 41
- get\_exposure\_functional\_use, 41
- get\_exposure\_functional\_use\_batch, 42
- get\_exposure\_functional\_use\_category, 43
- get\_exposure\_functional\_use\_probability, 44
- get\_exposure\_functional\_use\_probability\_batch, 45
- get\_exposure\_list\_presence\_tags, 46
- get\_exposure\_list\_presence\_tags\_by\_dt\_xsid, 46
- get\_exposure\_list\_presence\_tags\_by\_dt\_xsid\_batch, 47
- get\_exposure\_product\_data, 48
- get\_exposure\_product\_data\_batch, 49
- get\_exposure\_product\_data\_puc, 50

get\_fate\_by\_dtssid, [50](#)  
get\_fate\_by\_dtssid\_batch, [51](#)  
get\_fate\_by\_dtssid\_batch\_old, [52](#)  
get\_genetox\_details, [53](#)  
get\_genetox\_details\_batch, [53](#)  
get\_genetox\_details\_batch\_old, [54](#)  
get\_genetox\_summary, [55](#)  
get\_genetox\_summary\_batch, [56](#)  
get\_genetox\_summary\_batch\_old, [57](#)  
get\_hazard\_by\_dtssid, [57](#)  
get\_hazard\_by\_dtssid\_batch, [58](#)  
get\_hazard\_by\_dtssid\_batch\_old, [59](#)  
get\_hazard\_endpoint\_status, [60](#)  
get\_human\_hazard\_by\_dtssid, [60](#)  
get\_human\_hazard\_by\_dtssid\_batch, [61](#)  
get\_human\_hazard\_by\_dtssid\_batch\_old,  
[62](#)  
get\_inchi, [62](#)  
get\_inchikey, [63](#)  
get\_lists\_containing\_chemical, [64](#)  
get\_lists\_containing\_chemical\_batch,  
[64](#)  
get\_msready\_by\_dtscid, [65](#)  
get\_msready\_by\_dtscid\_batch, [66](#)  
get\_msready\_by\_formula, [67](#)  
get\_msready\_by\_formula\_batch, [67](#)  
get\_msready\_by\_mass, [68](#)  
get\_msready\_by\_mass\_batch, [69](#)  
get\_msready\_by\_mass\_with\_error\_batch,  
[70](#)  
get\_public\_chemical\_list\_by\_name, [71](#)  
get\_public\_chemical\_list\_by\_name\_batch,  
[72](#)  
get\_skin\_eye\_hazard, [73](#)  
get\_skin\_eye\_hazard\_batch, [73](#)  
get\_skin\_eye\_hazard\_batch\_old, [74](#)  
get\_smiles, [75](#)

has\_ctx\_key (register\_ctx\_api\_key), [77](#)  
has\_ctxR\_option (ctxR\_options), [10](#)  
has\_ctxR\_options (ctxR\_options), [10](#)  
hazard\_api\_server, [76](#)

prepare\_word, [76](#)  
print.ctx\_credentials  
    (register\_ctx\_api\_key), [77](#)

register\_ctx\_api\_key, [77](#)  
register\_ctx\_api\_key(), [10](#), [11](#), [77](#)

set\_ctxR\_option (ctxR\_options), [10](#)  
showing\_key (register\_ctx\_api\_key), [77](#)