# Package: sdmxdata (via r-universe)

June 5, 2026

**Title** Open Data SDMX API Client

**Version** 0.2.9000

**Description** The data and meta data from various statistical agencies such as OECD and Statistics Netherlands can be browsed and downloaded. The client uses the open data SDMX API of Statistics Netherlands.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Suggests** knitr, rmarkdown, tinytest

**Imports** httr2, data.table, jsonlite, R6

**URL** https://edwindj.github.io/sdmxdata/

**VignetteBuilder** knitr

**Depends** R (>= 4.2.0)

**LazyData** true

**Config/pak/sysreqs** libssl-dev

**Repository** https://edwindj.r-universe.dev

**Date/Publication** 2025-11-07 10:28:36 UTC

**RemoteUrl** https://github.com/edwindj/sdmxdata

**RemoteRef** HEAD

**RemoteSha** 5abeda0cdbe8fa6a26b6f13b6c54274bfc04bf95

## Contents

---

`as.data.frame.sdmx_v2_1_data_request`
*Transform a SDMX data request into a data.frame*

---

## Description

Retrieve data from an sdmx api and return it as a data.frame

## Usage

```
## S3 method for class 'sdmx_v2_1_data_request'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | An `sdmx_v2_1_data_request()` object |
| `row.names` | Not used |
| `optional` | Not used |
| `...` | reserved for future use |

## Value

a data.frame

## Examples

```
if (interactive()){
  dfs <- list_dataflows()

  flowRef <- paste0(dfs[2, c("agencyID", "id", "version")], sep=",")

  # json
```

```
  d <-
    sdmx_v2_1_data_request(flowRef = flowRef) |>
    as.data.frame()

}
```

---

`as.data.table.sdmx_v2_1_data_request` *Get data from a SDMX API*

---

## Description

Retrieve data from an sdmx api and return it as a data.frame

## Usage

```
## S3 method for class 'sdmx_v2_1_data_request'
as.data.table(x, keep.rownames = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | An httr2 request object |
| `keep.rownames` | logical, not used |
| `...` | reserved for future use |

## Value

data.table

## Examples

```
if (interactive()){
  dfs <- list_dataflows()

  flowRef <- paste0(dfs[2, c("agencyID", "id", "version")], sep=",")

  # json
  d <-
    sdmx_v2_1_data_request(flowRef = flowRef) |>
    as.data.frame()

}
```

## data_request_from_url *Create a data request from a URL*

### Description

This function creates a data request from a URL that contains the necessary parameters for an SDMX data request. Note that this is a raw request. Most users are better off using `get_observations_from_url()` instead.

### Usage

```
data_request_from_url(url, ..., verbose = FALSE)
```

### Arguments

| | |
|---|---|
| `url` | A URL that contains the parameters for the data request, including flowRef. |
| `...` | Additional parameters to be passed to the data request. |
| `verbose` | If `TRUE`, print information about the request. |

### Value

An object of class `sdmx_v2_1_data_request()` that can be used to retrieve data.

### Examples

```
# create a data request object from a URL
url <- "https://sdmx.oecd.org/public/rest/data/OECD.TAD.ARP,DSD_FISH_PROD@DF_FISH_AQUA,1.0/.A.._T.T?startPeriod

# it does not execute the request, it just creates the request object
req <- data_request_from_url(url, verbose = TRUE)

# this is the raw csv output of the SDMX data request
# most users are better of using `get_observations_from_url()`
d <- req |> as.data.frame()
head(d)
```

## download_to *Download the response of a request to a file*

### Description

Download the response of a request to a file

### Usage

```
download_to(x, destfile, ...)
```

### Arguments

| | |
|---|---|
| x | An httr2 request object |
| destfile | A character string with the path to the file to save the response |
| ... | saved for future use |

### Details

Download the response of a request to a file, but allows for different format specifications e.g. csv, json, xml

---

## endpoints *Endpoints with sdmx data*

---

### Description

A data.frame with several endpoints for sdmx data

### Usage

```
endpoints
```

### Format

A data.frame with columns:

- id - the id of the endpoint
- name - the name of agency of the endpoint
- url - the url of the endpoint
- language - the default language of the endpoint

### Examples

```
data("endpoints")
endpoints[, 1:3]

OECD <- get_endpoint("OECD")
OECD$verbose <- TRUE
list_oecd <- OECD |> list_dataflows(cache=TRUE)
```

get_data *Retrieve data from a SDMX data API*

## Description

Retrieve data from an SDMX data API. The function retrieves data from the SDMX data API and returns it as a data.frame.

## Usage

```
get_data(
  endpoint = NULL,
  agencyID,
  id,
  version = "latest",
  ref = NULL,
  startPeriod = NULL,
  endPeriod = NULL,
  filter_on = list(),
  ...,
  dim_contents = c("label", "both", "id"),
  obs_value_numeric = TRUE,
  raw = FALSE,
  language = NULL,
  cache = TRUE,
  pivot = NULL,
  verbose = getOption("sdmxdata.verbose", FALSE)
)
```

## Arguments

| | |
|---|---|
| endpoint | An endpoint, url or `httr2::request()` object. |
| agencyID | The agency ID |
| id | The id of the dataflow |
| version | The version of the dataflow, default "latest" |
| ref | The dataflow reference can be used in stead of `agencyID`, `id` and `version` |
| startPeriod | The start period for which the data should be returned, works only for dataflows with an explicit time dimension. |
| endPeriod | The end period for which the data should be returned, works only for dataflows with an explicit time dimension. |
| filter_on | A named list of filters to apply to the data, if not specified or `list()`, it is the default selection, set to `NULL` to select all. |
| ... | Additional parameters to pass to the request |
| dim_contents | The contents of the dimension columns, either "label", "id" or "both" |

| | |
|---|---|
| obs_value_numeric | Should the OBS_VALUE column be coerced to numeric? Default is TRUE |
| raw | If TRUE return the raw data.frame from the SDMX, otherwise the data.frame is processed |
| language | The language of the metadata |
| cache | if TRUE cache the accompanying meta data. |
| pivot | character The name of the column to pivot the data on. If NULL (default) the data is returned in a long format. |
| verbose | If TRUE print information about the request |

## Details

get_data is similar to get_observations(), but its differs in two aspects:

- it returns only data columns, no attribute columns
- it can pivot the observations, i.e. it can return the data in a wide format, which is useful feature when the data has multiple measures or when the data is to presented as a time series.

## Value

a data.frame

## See Also

Other retrieve data: get_observations(), get_observations_from_url()

---

# get_dataflow_structure

*Get information about a dataflow*

---

## Description

Get information and structure about a dataflow. The dataflow can be identified using agencyID, id and version or by using the ref argument.

## Usage

```
get_dataflow_structure(
  endpoint = NULL,
  agencyID = getOption("sdmxdata.agencyID", NULL),
  id,
  version = "latest",
  ref,
  language = getOption("sdmxdata.language"),
  cache = TRUE,
  verbose = getOption("sdmxdata.verbose", FALSE)
)
```

### Arguments

| | |
|---|---|
| endpoint | An endpoint or an object that can be coerced to an SDMXEndpoint |
| agencyID | The agency ID of the dataflow |
| id | The id of the dataflow |
| version | The version of the dataflow, defaults to "latest" |
| ref | A string with the flow reference, in the form of "agencyID:id(version)". Overrules the agencyID, id and version arguments. |
| language | The language of the metadata |
| cache | logical, if TRUE cache the metadata |
| verbose | print some information on the console |

### Details

For most use cases the agencyID and id are to be preferred, as the ref argument includes a specific version number of the dataflow. Not specifying a version number will default to the latest version. If it is desirable to pin the reference of the dataflow to a specific version, the ref argument can be used. The ref argument can also be found in list_dataflows().

### Value

a list with the dataflow information

---

## get_endpoint *Connect to a SDMX endpoint*

---

### Description

Connect to an SDMX endpoint

### Usage

```
get_endpoint(id, language = NULL, singleton = TRUE, verbose = FALSE)
```

### Arguments

| | |
|---|---|
| id | character, id of one of the available endpoints. |
| language | character, the language to use for the text used in the response. |
| singleton | logical, if TRUE return the same object if it already exists |
| verbose | logical, if TRUE print information about the dataflows. |

### Value

a SDMXEndpoint object

## See Also

Other SDMX endpoints: SDMXEndpoint, sdmx_endpoint()

## Examples

```
data("endpoints")
endpoints[, 1:3]

OECD <- get_endpoint("OECD")
OECD$verbose <- TRUE
list_oecd <- OECD |> list_dataflows(cache=TRUE)
```

---

# get_observations *Get observations from an SDMX provider*

---

## Description

Get observations from an SDMX provider. It retrieves the data from the SDMX provider and returns it as a data.frame.

## Usage

```
get_observations(
  endpoint = NULL,
  agencyID,
  id,
  version = "latest",
  ref = NULL,
  startPeriod = NULL,
  endPeriod = NULL,
  filter_on = list(),
  use_factor = TRUE,
  ...,
  language = NULL,
  as.data.table = FALSE,
  dim_contents = c("label", "both", "id"),
  attributes_contents = c("label", "id", "both"),
  obs_value_numeric = TRUE,
  raw = FALSE,
  drop_first_columns = !raw,
  cache = TRUE,
  verbose = getOption("sdmxdata.verbose", FALSE)
)
```

## Arguments

| | |
|---|---|
| endpoint | An endpoint, url or `httr2::request()` object. |
| agencyID | The agency ID |
| id | The id of the dataflow |
| version | The version of the dataflow, default "latest" |
| ref | The dataflow reference can be used in stead of `agencyID`, `id` and `version` |
| startPeriod | The start period for which the data should be returned, works only for dataflows with an explicit time dimension. |
| endPeriod | The end period for which the data should be returned, works only for dataflows with an explicit time dimension. |
| filter_on | A named list of filters to apply to the data, if not specified or `list()`, it is the default selection, set to `NULL` to select all. |
| use_factor | if `TRUE` then dimension and attributes columns are factors, otherwise character |
| ... | Additional parameters to pass to the request |
| language | The language of the metadata |
| as.data.table | If `TRUE` return a `data.table::data.table()`, otherwise a `data.frame()` |
| dim_contents | The contents of the dimension columns, either "label", "id" or "both" |
| attributes_contents | The contents of the attribute columns, either "label", "id" or "both" |
| obs_value_numeric | Should the OBS_VALUE column be coerced to numeric? Default is `TRUE` |
| raw | If `TRUE` return the raw data.frame from the SDMX, otherwise the data.frame is processed |
| drop_first_columns | Should the first columns be dropped? Default is `TRUE` (if not raw) |
| cache | if `TRUE` cache the accompanying meta data. |
| verbose | If `TRUE` print information about the request |

## Details

By default the dimension and attribute columns are recoded to factors with labels. This can be changed with the `dim_contents` and `attributes_contents` arguments.

## Value

`data.frame()` or `data.table::data.table()` depending on `as.data.table`

## See Also

Other retrieve data: `get_data()`, `get_observations_from_url()`

## Examples

```
dfs <- list_dataflows()
ref <- dfs$ref[4]

  dfs <- list_dataflows()
  ref <- dfs$ref[4]

  obs <- get_observations(
    ref = ref,
    filter_on = list(
      "Perioden" = c("2009JJ00")
    ),
    verbose = TRUE
  )

obs
```

---

`get_observations_from_url` *Get observations using a data-explorer url*

---

## Description

Retrieve observations using the url from the data-explorer. The url given will be parsed and translated into a `get_observations()` call, to get you started. The call will be printed to the console or can be retrieved by settings `return_query = TRUE`.

## Usage

```
get_observations_from_url(url, return_query = FALSE, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `url` | character, the REST 2.1 url retrieved from an external source |
| `return_query` | logical, if `TRUE` return the query object instead of the observation |
| `verbose` | logical, if `TRUE` print information about the query |

## Value

when `return_query = FALSE` a data.frame with the observations, i.e. same output as `get_observations()`. Otherwise a list with the query object and the parameters

## See Also

Other retrieve data: `get_data()`, `get_observations()`

## Examples

```
# retrieve an external url pointing to sdmx data

url <-
 "https://sdmx.oecd.org/public/rest/data/OECD.TAD.ARP,DSD_FISH_PROD@DF_FISH_AQUA,1.0/.A.._T.T?startPeriod=2010

obs <- get_observations_from_url(url)
head(obs)

query <- get_observations_from_url(url, return_query = TRUE)
query
```

---

`list_agencies` *list the agencies of an endpoint*

---

## Description

list the agencies of an endpoint

## Usage

```
list_agencies(
  endpoint = NULL,
  language = NULL,
  raw = FALSE,
  cache = TRUE,
  verbose = getOption("sdmxdata.verbose", FALSE)
)
```

## Arguments

| | |
|---|---|
| `endpoint` | an endpoint, url or an `httr2::request()` object. |
| `language` | the language to return the data in. |
| `raw` | if `TRUE` return the raw data. |
| `cache` | if `TRUE` cache the list of agencies. |
| `verbose` | if `TRUE` print information about the caching. |

`list_categoryschemes` *Get sdmx category schemes*

## Description

Get sdmx categoryschemes from a given endpoint.

## Usage

```
list_categoryschemes(
  endpoint = NULL,
  agencyID = NULL,
  ...,
  language = "nl",
  cache = TRUE,
  raw = FALSE,
  verbose = getOption("sdmxdata.verbose", FALSE)
)
```

## Arguments

| | |
|---|---|
| `endpoint` | an endpoint, url or `httr2::request()` object. |
| `agencyID` | A character string from a given agencyID. |
| `...` | saved for future use. |
| `language` | The language to use for the text used in the response. |
| `cache` | if `TRUE` cache the result. |
| `raw` | If `TRUE` return the raw data from the SDMX, otherwise the data is processed. |
| `verbose` | if `TRUE` print information about the dataflows. |

## Value

a data.frame with available dataflows

## Examples

```
dfs <- list_dataflows(verbose = TRUE)

head(dfs)
```

`list_dataflows` *List available sdmx dataflows*

## Description

List the sdmx dataflows from a given endpoint.

## Usage

```
list_dataflows(
  endpoint = NULL,
  agencyID = NULL,
  ...,
  language = NULL,
  cache = TRUE,
  raw = FALSE,
  verbose = getOption("sdmxdata.verbose", FALSE)
)
```

## Arguments

| | |
|---|---|
| `endpoint` | A character string or endpoint for a given endpoint. |
| `agencyID` | A character string from a given agencyID. |
| `...` | saved for future use. |
| `language` | The language to use for the text used in the response. |
| `cache` | if `TRUE` cache the list of dataflows |
| `raw` | If `TRUE` return the raw data from the SDMX, otherwise the data is processed. |
| `verbose` | if `TRUE` print information about the dataflows. |

## Value

a data.frame with available dataflows

## Examples

```
dfs <- list_dataflows(verbose = TRUE)

head(dfs)
```

sdmx_endpoint *Create a new SDMX endpoint*

## Description

Create an endpoint from an url or httr2 request object

## Usage

```
sdmx_endpoint(x, ...)

## S3 method for class 'character'
sdmx_endpoint(
  x,
  id = gsub("\\W+", "_", x),
  name = x,
  language = NULL,
  cache_dir = file.path(tempdir(), "sdmxdata", id),
  verbose = getOption("sdmxdata.verbose", FALSE),
  ...
)

## S3 method for class 'httr2_request'
sdmx_endpoint(x, ...)
```

## Arguments

| | |
|---|---|
| x | An endpoint or an httr2 request object |
| ... | saved for future use |
| id | character, the id of the endpoint |
| name | character, the name of the endpoint |
| language | character, the language to use for the text used in the response |
| cache_dir | character, the directory to cache the metadata in |
| verbose | logical, if TRUE print information on the connection |

## Value

a SDMXEndpoint object

## See Also

Other SDMX endpoints: SDMXEndpoint, get_endpoint()

sdmx_v2_1_data_request *Request SDMX data using the SDMX REST API v2.1*

## Description

sdmx_v2_1_data_request is a wrapper around the smdx rest api v2.1 and is used to retrieve data from the api.

## Usage

```
sdmx_v2_1_data_request(
  endpoint = NULL,
  resource = c("data", "metadata"),
  flowRef = NULL,
  key = NULL,
  providerRef = NULL,
  ...,
  startPeriod = NULL,
  endPeriod = NULL,
  updatedAfter = NULL,
  firstNObservations = NULL,
  lastNObservations = NULL,
  dimensionAtObservation = NULL,
  detail = c("full", "dataonly", "serieskeysonly", "nodata"),
  includeHistory = NULL,
  labels = c("both", "id")
)
```

## Arguments

| | |
|---|---|
| endpoint | An endpoint or an object that can be coerced to an endpoint |
| resource | The resource to request. Either "data" or "metadata" |
| flowRef | The flow reference to request, see details |
| key | The key to request, see details |
| providerRef | The provider reference to request, see details |
| ... | Additional parameters to pass to the request |
| startPeriod | The start period for which the data should be returned |
| endPeriod | The end period for which the data should be returned |
| updatedAfter | Only return data that has been updated after this date |
| firstNObservations | Only return the first n observations |
| lastNObservations | Only return the last n observations |

| | |
|---|---|
| dimensionAtObservation | The dimensions of the observations should be returned |
| detail | The detail of the data to return. Either "full", "dataonly", "serieskeysonly" or "nodata" |
| includeHistory | Include the different versions of the data |
| labels | Include the labels in the data, only valid when format is "csv" |

## Value

a modified `httr2::request()` object

---

`sdmx_v2_1_structure_request` *Retrieve sdmx structure*

---

## Description

Wrapper for the SDMX REST v2.1 interface for structured information

## Usage

```
sdmx_v2_1_structure_request(
  endpoint = NULL,
  resource = "dataflow",
  agencyID = NULL,
  resourceID = NULL,
  version = NULL,
  itemID = NULL,
  format = c("json", "xml"),
  language = NULL,
  ...,
  detail = c("full", "allstubs", "referencestubs", "allcompletestubs",
    "referencecompletestubs", "referencepartial"),
 references = c("none", "parents", "parentsandsiblings", "children", "descendants",
    "all")
)
```

## Arguments

| | |
|---|---|
| endpoint | An endpoint or an object that can be coerced to an endpoint |
| resource | The resource to request. One of "datastructure", "metadatastructure", "categoryscheme", "conceptscheme", "codelist", "hierarchicalcodelist", "organisationscheme", "agencyscheme", "dataproviderscheme", "dataconsumerscheme", "organisationunitscheme", "dataflow", "metadataflow", "reportingtaxonomy", "provisionagreement", "structureset", "process", "categorisation", "contentconstraint", "attachmentconstraint", "actualconstraint", "allowedconstraint", "structure", "transformationscheme", "rulesetscheme", "userdefinedoperatorscheme", "customtypescheme", "namepersonalisationscheme", "vtlmappingscheme" |

| | |
|---|---|
| agencyID | The agency ID to request |
| resourceID | The resource ID to request |
| version | The version to request |
| itemID | The item ID to request |
| format | The format to request. Either "json" or "xml" |
| language | The language to use for the text used in the response |
| ... | saved for future use |
| detail | The detail of the data to return. Either "full", "allstubs", "referencestubs", "allcompletestubs", "referencecompletestubs", "referencepartial" |
| references | The references to return. Either "none", "parents", "parentsandsiblings", "children", "descendants", "all" |

## Value

a modified `httr2::request()` object

---

`SDMXEndpoint` *Base class for SDMX endpoints*

---

## Description

Create a:

- a known endpoint from `endpoints` use `get_endpoint()`.
- unlisted SDMX 2.1 REST endpoint use `sdmx_endpoint()`.
- a new SDMX endpoint use `SDMXEndpoint$new()`.

Base class for SDMX endpoints

## Public fields

`req` `httr2::request()` object, can be adjusted to add headers or proxy settings

`version` character, the SDMX version of the endpoint

`id` character, the id of the endpoint

`name` character, the name of the endpoint

`url` character, the url of the endpoint

`verbose` logical, if `TRUE` print information the connection

`cache_dir` character, the directory to cache the metadata in

`language` character, the language to use for the text used in the response

## Methods

### Public methods:

- SDMXEndpoint$new()
- SDMXEndpoint$clone()

**Method** `new()`**:** create a new SDMXEndpoint object, see also `sdmx_endpoint()`

*Usage:*

```
SDMXEndpoint$new(
  url,
  id = gsub("\\W+", "_", url),
  name = url,
  language = NULL,
  cache_dir = file.path(tempdir(), "sdmxdata", id),
  verbose = getOption("sdmxdata.verbose", FALSE)
)
```

*Arguments:*

`url` character, the url of the endpoint

`id` character, the id of the endpoint

`name` character, the name of the endpoint

`language` character, the language to use for the text used in the response

`cache_dir` character, the directory to cache the metadata in

`verbose` logical, if `TRUE` print information on the connection

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

*Usage:*

```
SDMXEndpoint$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## See Also

Other SDMX endpoints: `get_endpoint()`, `sdmx_endpoint()`

# Index