

Package ‘fedstatAPIr’

May 8, 2026

Title Unofficial API for Fedstat (Rosstat EMISS System) for Automatic and Efficient Data Queries

Version 1.0.3

Description An API for automatic data queries to the fedstat <<https://www.fedstat.ru>>, using a small set of functions with a common interface.

License MIT + file LICENSE

URL <https://github.com/DenchPokepon/fedstatAPIr>

BugReports <https://github.com/DenchPokepon/fedstatAPIr/issues>

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports htr, jsonlite, xml2, readsdmx, magrittr, utils, methods, data.table

Depends R (>= 2.10)

NeedsCompilation no

Author Denis Krylov [aut, cre],
Dmitry Kibalnikov [aut]

Maintainer Denis Krylov <deniskrylovvit@gmail.com>

Repository CRAN

Date/Publication 2023-03-30 08:30:02 UTC

Contents

fedstat_data_ids_filter	2
fedstat_data_load_with_filters	3
fedstat_get_data_ids	5
fedstat_indicators_names_database	7
fedstat_indicator_info	8
fedstat_parse_sdmx_to_table	8
fedstat_post_data_ids_filtered	10
parse_js1	11
parse_js2	12

fedstat_data_ids_filter

Filters data_ids based on filters that are given in JSON format

Description

Filters indicator data_ids with given filters taking into account possible filters specification errors and default filters.

filters should use filter_field_title in names and filter_value_title in values as they are presented on fedstat.ru. If for some reason the specified filters do not return the expected result, it is worth inspecting possible filter values in data_ids to see if the strings are defined correctly (e.g. encoding issues, mixing latin and cyrillic symbols)

filter_value_title currently supports the following special values:

1. asterix (*), it's alias for "select all possible filter values for this filter field"

Unspecified filters use asterix as a default (i.e. all possible filter values are selected and a warning is given)

Internally normalized filter_field_title and filter_value_title are used (all lowercase, removed extra whitespaces) to compare the equality of data_ids and filters

Usage

```
fedstat_data_ids_filter(data_ids, filters = list(), disable_warnings = FALSE)
```

Arguments

data_ids data.frame, result of fedstat_get_data_ids with or without conjunction with fedstat_get_data_ids_special_cases_handle

filters JSON in R list form. The structure should be like this:

```
{
  "filter_field_title1": ["filter_value_title1", "filter_value_title2"],
  "filter_field_title2": ["filter_value_title1", "filter_value_title2"],
  ...
}
```

Where for example filter_field_title1 could be a string "Year" with filter_value_title1 equal to 2020 and filter_field_title2 could be a string "OKATO" with filter_value_title1 equal to "Russian Federation" Not actual filter field titles and filter values titles because of ASCII requirement for CRAN

disable_warnings

bool, enables or disables following warnings:

1. About non matched filter_value_title in filters and filter_value_title from data_ids;
2. About unspecified filter_field_title in filters.

Value

data.frame, filtered data_ids

See Also

[fedstat_get_data_ids](#), [fedstat_post_data_ids_filtered](#)

Examples

```
## Not run:
# Get data filters identifiers for CPI
# filter the data_ids to get data for january of 2023
# for all goods and services for Russian Federation
data_ids_filtered <- fedstat_get_data_ids("31074") %>%
  fedstat_data_ids_filter(
    filters = list(
      "Territory" = "Russian Federation",
      "Year" = "2023",
      "Period" = "January",
      "Types of goods and services" = "*"
    )
  )

# Not actual filter field titles and filter values titles because of ASCII requirement for CRAN

## End(Not run)
```

fedstat_data_load_with_filters

Download subset of indicator data from fedstat.ru by specifying filters in JSON

Description

This function is a wrapper for the other functions of the package to provide a simple one function API for fedstat.ru

There are two basic terms in this API: `filter_field` and `filter_value`

The `filter_field` reflects the individual property of the data point. For example, Year, Region, Unit of measurement, etc. Each filter field has its own title (`filter_field_title`), it is simply a human-readable word or phrase (e.g. "Year", "Region") that reflects the essence of the property by which filtering takes place

The `filter_value` reflects the individual property specific value of the data point. (e.g. 2021 for the Year, "Russian Federation" for the region, etc.) It also has a title (`filter_value_title`) with the same purpose as `filter_field_title`

`filters` should use `filter_field_title` in names and `filter_value_title` in values as they are presented on fedstat.ru. If for some reason the specified filters do not return the expected result, it

is worth using `fedstat_get_data_ids` separately and inspecting possible filter values in `data_ids` to see if the strings are defined correctly (e.g. encoding issues, mixing latin and cyrillic symbols) `filter_value_title` currently supports the following special values:

1. asterix (*), alias for "select all possible filter values for this filter field"

Unspecified filters use asterix as a default (i.e. all possible filter values are selected and a warning is given)

Internally normalized `filter_field_title` and `filter_value_title` are used (all lowercase, removed extra whitespaces) to compare the equality of `data_ids` and `filters`

Usage

```
fedstat_data_load_with_filters(
  indicator_id,
  ...,
  filters = list(),
  timeout_seconds = 180,
  retry_max_times = 3,
  disable_warnings = FALSE,
  htrr_verbose = NULL,
  loading_steps_verbose = TRUE,
  return_type = c("data", "dictionary"),
  try_to_parse_obsvalue = TRUE
)
```

Arguments

`indicator_id` character, indicator id/code from indicator URL. For example for indicator with URL <https://www.fedstat.ru/indicator/37426> indicator id will be 37426

... other arguments passed to `htrr::GET` and `htrr::POST`

`filters` JSON in R list form. The structure should be like this:

```
{
  "filter_field_title1": ["filter_value_title1", "filter_value_title2"],
  "filter_field_title2": ["filter_value_title1", "filter_value_title2"],
  ...
}
```

Where for example `filter_field_title1` could be a string "Year" with `filter_value_title1` equal to 2020 and `filter_field_title2` could be a string "OKATO" with `filter_value_title1` equal to "Russian Federation" Not actual filter field titles and filter values titles because of ASCII requirement for CRAN

`timeout_seconds` numeric, maximum time before a new GET and POST request is tried

`retry_max_times` numeric, maximum number of tries to GET and POST `data_ids`

`disable_warnings` bool, enables or disables following warnings:

1. About non matched filter_value_title in filters and filter_value_title from data_ids;
2. About unspecified filterFiled_title in filters.

httr_verbose httr::verbose() or NULL, outputs messages to the console about the processing of the request

loading_steps_verbose logical, print data loading steps to console

return_type character, "data" or "dictionary", data for actual data, dictionary for sdmx lookup table (full data codes dictionary)

try_to_parse_ObsValue logical, try to parse ObsValue column from character to R numeric type

Value

data.frame with filtered indicator data from fedstat.ru

See Also

[fedstat_get_data_ids](#), [fedstat_data_ids_filter](#), [fedstat_post_data_ids_filtered](#), [fedstat_parse_sdmx_to](#)

Examples

```
## Not run:
# Download CPI data
# for all goods and services for Russian Federation
data <- fedstat_data_load_with_filters(
  indicator_id = "31074",
  filters = list(
    "Territory" = "Russian Federation",
    "Year" = "2023",
    "Period" = "January",
    "Types of goods and services" = "*"
  )
)
# Not actual filter field titles and filter values titles because of ASCII requirement for CRAN

## End(Not run)
```

fedstat_get_data_ids *Get data filters ids from fedstat.ru indicator web page*

Description

To query data from fedstat we need to POST some filters in form of filter numeric identifiers. Most filters don't have some rule from which their ids can be generated based on filters titles and values. It seems like these ids are just indexes in the fedstat inner database. So in order to get the data, we first need to get the ids of the filter values by parsing specific part of java script source code on indicator web page.

Usage

```

fedstat_get_data_ids(
  indicator_id,
  ...,
  timeout_seconds = 180,
  retry_max_times = 3,
  httr_verbose = NULL
)

```

Arguments

<code>indicator_id</code>	character, indicator id/code from indicator URL. For example for indicator with URL https://www.fedstat.ru/indicator/37426 indicator id will be 37426
<code>...</code>	other arguments passed to <code>httr::GET</code>
<code>timeout_seconds</code>	numeric, maximum time before a new GET request is tried
<code>retry_max_times</code>	numeric, maximum number of tries to GET <code>data_ids</code>
<code>httr_verbose</code>	<code>httr::verbose()</code> or <code>NULL</code> , outputs messages to the console about the processing of the request

Details

It is known that the fedstat lags quite often. Sometimes site never responds at all. This is especially true for the most popular indicators web pages. In this regard, by default, a GET request is sent 3 times with a timeout of 180 seconds and with initially small, but growing exponentially, pauses between requests.

As a rule, requests to the indicator web page take much longer than requests to get the data itself. A POST request for data is sent to a single URL [https://www.fedstat.ru/indicator/data.do?format=\(excel or sdmx\)](https://www.fedstat.ru/indicator/data.do?format=(excel or sdmx)) for all indicators and is often quite fast. In this regard, for many indicators, it makes sense to cache `data_ids` to increase the speed of data download. This is not possible for all data, for example, for weekly prices, each new week adds a new filter (new week), the id of which can only be found on the indicator web page. But for most data (e.g. monthly frequency), time filters are trivial. There are 12 months in total with unique ids that do not change and year ids that match their values (that is, `filter_value_id = filter_value`, in other words 2020 = 2020)

Correct `filter_field_object_ids` are needed to get data. For the `sdmx` format, these ids do not change anything, except for the standard data sorting, but their incorrect specification will lead either to incomplete data loading or to no data at all. For the `excel` format, these ids determine the form of data presentation, as in the data preview on the fedstat site. For now only default `filter_field_object_ids` are used, which are parsed from java script source code on indicator web page. Users can specify `filter_field_object_ids` for each `filter_field` in resulting `data_ids` table.

Value

`data.frame` with all character type columns:

1. `filter_field_id` - id for filter field;

2. filter_field_title - filter field title string representation;
3. filter_value_id - id for filter field value;
4. filter_value_title - filter field value title string representation;
5. filter_field_object_ids - special strings that define the location of the filters fields. It can take the following values: lineObjectIds (filters in lines), columnObjectIds (filters in columns), filterObjectIds (hidden filters for all data);

See Also

[fedstat_data_ids_filter](#), [fedstat_post_data_ids_filtered](#)

Examples

```
## Not run:  
# Get data filters identifiers for CPI  
data_ids <- fedstat_get_data_ids("31074")  
  
## End(Not run)
```

fedstat_indicators_names_database

Database of all indicator names presented on the fedstat.ru with hierarchical grouping

Description

Allows researchers to search for interesting indicators more easily

Usage

```
fedstat_indicators_names_database
```

Format

A data frame with 9335 rows and 10 variables:

name indicator name

url indicator url

excluded boolean, TRUE if indicator is not used and updated anymore

department the name of the department from which the data is coming from

group_level_2 grouping of indicator

group_level_3 grouping of indicator

group_level_4 grouping of indicator

group_level_5 grouping of indicator

group_level_6 grouping of indicator

date_of_update date of the last update of the current database

Source

<https://fedstat.ru/organizations/>

fedstat_indicator_info

Download indicator information

Description

Download indicator information from <https://www.fedstat.ru/organizations/>
Result table contains fedstat indicator id which is needed to request fedstat data
Indicator with condition hidden == TRUE shows disabled records in fedstat hence ones might not be requested

Usage

```
fedstat_indicator_info()
```

Value

data.frame

See Also

[fedstat_get_data_ids](#)

Examples

```
## Not run:  
Get all indicator info  
get_indicators()  
  
## End(Not run)
```

fedstat_parse_sdmx_to_table

Parse sdmx raw bytes to data.frame

Description

Parses sdmx raw bytes received in response to POST request. This function is a wrapper around `readsdmx::read_sdmx`, in addition to reading data, automatically adds columns with values from lookup tables. Can also return full data codes dictionary for the indicator

Usage

```
fedstat_parse_sdmx_to_table(  
  data_raw,  
  return_type = c("data", "dictionary"),  
  try_to_parse_ObsValue = TRUE  
)
```

Arguments

data_raw	sdmx raw bytes
return_type	character, "data" or "dictionary", data for actual data, dictionary for sdmx lookup table (full data codes dictionary)
try_to_parse_ObsValue	logical, try to parse ObsValue column from character to R numeric type

Value

data.frame

See Also

[fedstat_parse_sdmx_to_table](#)

Examples

```
## Not run:  
# Get data filters identifiers for CPI  
# filter the data_ids to get data for january of 2023  
# for all goods and services for Russian Federation  
# POST filters and download data in sdmx format  
# Parse raw sdmx to data.frame  
data <- fedstat_get_data_ids("31074") %>%  
  fedstat_data_ids_filter(  
    filters = list(  
      "Territory" = "Russian Federation",  
      "Year" = "2023",  
      "Period" = "January",  
      "Types of goods and services" = "*" )  
  ) %>%  
  fedstat_post_data_ids_filtered() %>%  
  fedstat_parse_sdmx_to_table()  
  
# Not actual filter field titles and filter values titles because of ASCII requirement for CRAN  
  
## End(Not run)
```

 fedstat_post_data_ids_filtered

Post data filters ids to fedstat.ru and download specified subset of data

Description

Creates a request body from `data_ids` and sends it to `https://www.fedstat.ru/indicator/data.do?format=data_format`. Gets an sdmx or excel with data in binary format.

sdmx raw bytes can be passed to `fedstat_parse_sdmx_to_table` to create a `data.frame` or to `rawToChar` and `writeln` to create an xml file

excel raw bytes can be passed to `writeBin` to create an xls file

Usage

```
fedstat_post_data_ids_filtered(
  data_ids,
  ...,
  data_format = c("sdmx", "excel"),
  timeout_seconds = 180,
  retry_max_times = 3,
  httr_verbose = NULL
)
```

Arguments

<code>data_ids</code>	data.frame, can be a result of <code>fedstat_get_data_ids</code> or <code>fedstat_get_data_ids_special_cases_han</code> to download all available data, or a result of <code>fedstat_data_ids_filter</code> to download subset of available data
<code>...</code>	other arguments passed to <code>httr::POST</code>
<code>data_format</code>	string, one of sdmx, excel
<code>timeout_seconds</code>	numeric, maximum time before a new POST request is tried
<code>retry_max_times</code>	numeric, maximum number of tries to POST <code>data_ids</code>
<code>httr_verbose</code>	<code>httr::verbose()</code> or <code>NULL</code> , outputs messages to the console about the processing of the request

Value

raw bytes (sdmx or excel)

See Also

[fedstat_parse_sdmx_to_table](#)

Examples

```
## Not run:
# Get data filters identifiers for CPI
# filter the data_ids to get data for january of 2023
# for all goods and services for Russian Federation
# POST filters and download data in sdmx format
data <- fedstat_get_data_ids("31074") %>%
  fedstat_data_ids_filter(
    filters = list(
      "Territory" = "Russian Federation",
      "Year" = "2023",
      "Period" = "January",
      "Types of goods and services" = "*"
    )
  ) %>%
  fedstat_post_data_ids_filtered()

# Not actual filter field titles and filter values titles because of ASCII requirement for CRAN

## End(Not run)
```

parse_js1

Get data ids from java script source

Description

Get data ids from java script source

Usage

```
parse_js1(script)
```

Arguments

script character, java script source code with data ids

Value

json in form of list with data ids

`parse_js2`*Get default data ids object ids from java script source*

Description

Get default data ids object ids from java script source

Usage

```
parse_js2(script)
```

Arguments

`script` character, java script source code with data ids and default object ids in it

Value

json in form of list with 3 character vectors for `lineObjectIds`, `columnObjectIds`, `filterObjectIds`, which consist of `filters_id`

Index

* datasets

fedstat_indicators_names_database,
7

fedstat_data_ids_filter, 2, 5, 7
fedstat_data_load_with_filters, 3
fedstat_get_data_ids, 3-5, 5, 8
fedstat_indicator_info, 8
fedstat_indicators_names_database, 7
fedstat_parse_sdmx_to_table, 5, 8, 9, 10
fedstat_post_data_ids_filtered, 3, 5, 7,
10

parse_js1, 11
parse_js2, 12