

# Package ‘fedstatAPIr’

January 14, 2022

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

**Version** 0.2.0

**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** 7.1.2

**Imports** htr, rvest, jsonlite, stringr, dplyr, xml2, readsdmx, magrittr, utils, rlang, methods

**Depends** R (>= 2.10)

**NeedsCompilation** no

**Author** Denis Krylov [aut, cre]

**Maintainer** Denis Krylov <[deniskrylovvit@gmail.com](mailto:deniskrylovvit@gmail.com)>

**Repository** CRAN

**Date/Publication** 2022-01-14 16:52:46 UTC

## R topics documented:

fedstatAPIr-package . . . . .	2
fedstat_check_data_update . . . . .	2
fedstat_check_data_update_ . . . . .	4
fedstat_data_ids_filter . . . . .	6
fedstat_data_load_with_filters . . . . .	7
fedstat_get_data_ids . . . . .	10
fedstat_get_data_ids_special_cases_handle . . . . .	11
fedstat_indicators_names_database . . . . .	13

fedstat_java_script_data_ids_parse_to_json . . . . .	14
fedstat_java_script_default_data_ids_object_ids_parse_to_json . . . . .	14
fedstat_parse_all_indicators_names . . . . .	15
fedstat_parse_sdmx_to_table . . . . .	15
fedstat_post_data_ids_filtered . . . . .	16
fedstat_prepare_reference_data_for_check_data_update . . . . .	18

<b>Index</b>	<b>21</b>
--------------	-----------

---

fedstatAPIr-package	<i>Unofficial API for fedstat (EMISS) for automatic and efficient data queries</i>
---------------------	--

---

## Description

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

## Author(s)

**Maintainer:** Denis Krylov <[deniskrylovvit@gmail.com](mailto:deniskrylovvit@gmail.com)>

## See Also

Useful links:

- <https://github.com/DenchPokepon/fedstatAPIr>
- Report bugs at <https://github.com/DenchPokepon/fedstatAPIr/issues>

---

fedstat_check_data_update	<i>Very basic task scheduler for checking the indicator data updates (EXPERIMENTAL)</i>
---------------------------	---

---

## Description

In practice fedstat often does not give true information about when the data was updated. Because of this, the only universal and accurate way to check for updated data is to try to download it and compare it to past data. If a query with the same filters as before returned different `data_ids` or `data_df` (result of `fedstat_parse_sdmx_to_table`), then the data has been updated.

This function is not tested on real data

Logging to a file is supported

**Usage**

```

fedstat_check_data_update(
  prepared_reference_data_for_check_data_update,
  ...,
  pause_minutes_between_requests = 30,
  max_checks = 50,
  verbose_tries = TRUE,
  timeout_seconds = 180,
  retry_max_times = 10,
  disable_warnings = TRUE,
  htrr_verbose = NULL,
  logfile = NULL,
  overwrite_logfile = FALSE
)

```

**Arguments**

prepared_reference_data_for_check_data_update	result of <a href="#">fedstat_prepare_reference_data_for_check_data_update</a>
...	other arguments passed to <code>htrr::GET</code> and <code>htrr::POST</code>
pause_minutes_between_requests	numeric, the number of minutes to wait before the next request, cannot be less than 10 minutes to avoid a heavy load on the fedstat.ru The time between requests will not be exactly equal to the specified value, because the countdown starts only after the request is finished
max_checks	numeric, maximum number of checks before returning FALSE
verbose_tries	bool, enables or disables verbose messages, sends messages to the console about the progress of attempts to find out whether the data have been updated
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 in <code>fedstat_data_ids_filter</code> : <ol style="list-style-type: none"> <li>1. About non matched <code>filter_value_title</code> in <code>filters</code> and <code>filter_value_title</code> from <code>data_ids</code>;</li> <li>2. About unspecified <code>filter_filed_title</code> in <code>filters</code>.</li> </ol>
htrr_verbose	<code>htrr::verbose()</code> or NULL, outputs messages to the console about the processing of the request
logfile	the path to the log file, if NULL, the messages are sent to the console
overwrite_logfile	boolean, if true overwrites content of logfile

**Value**

bool, TRUE if indicator data has been updated, FALSE if `max_checks` is exceeded

**Examples**

```
## Not run:
is_updated <- fedstat_prepare_reference_data_for_check_data_update(
  indicator_id = "37426",
  filters = list(
    "Territory" = "Russian Federation",
    "Types of goods and services" = "Sahar-pesok, kg"
  )
) %>%
  fedstat_check_data_update(logfile = "logfile.txt")

if (is_updated) print("Data for the indicator 37426 has been updated!")

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

## End(Not run)
```

---

fedstat\_check\_data\_update\_

*Compares reference\_data and new\_data for any changes*

---

**Description**

Compares reference\_data and new\_data for any changes

**Usage**

```
fedstat_check_data_update_(
  indicator_id,
  reference_data_ids_unfiltered_special_cases_handled,
  reference_data_df,
  ...,
  filters = list(),

  time_filter_fields_titles = c("\u041f\u0435\u0440\u0438\u043e\u0434",
    "\u0413\u043e\u0434"),
  time_fields_titles_in_df = c("PERIOD", "Time"),
  filter_value_title_alias_lookup_table = data.frame(filter_value_title = character(),
    filter_value_title_alias = character(), stringsAsFactors = FALSE),
  timeout_seconds = 180,
  retry_max_times = 3,
  disable_warnings = FALSE,
  httr_verbose = httr::verbose(data_out = FALSE)
)
```

**Arguments**

<code>indicator_id</code>	character, indicator id/code from indicator URL. For example for indicator with URL <a href="https://www.fedstat.ru/indicator/37426">https://www.fedstat.ru/indicator/37426</a> indicator id will be 37426
<code>reference_data_ids_unfiltered_special_cases_handled</code>	data_ids of reference_data without filtering and with special cases handled
<code>reference_data_df</code>	data_df of reference_data
<code>...</code>	other arguments passed to <code>httr::GET</code> and <code>httr::POST</code>
<code>filters</code>	JSON in R list form. The structure should be like this: <pre> {   "filter_field_title1": ["filter_value_title1", "filter_value_title2"],   "filter_field_title2": ["filter_value_title1", "filter_value_title2"],   ... } </pre> <p>Where for example <code>filter_field_title1</code> could be a string "Year" with <code>filter_value_title1</code> equal to 2020 and <code>filter_field_title2</code> could be a string "OKATO" with <code>filter_value_title1</code> equal to "Russian Federation" Not actual filter field titles and filter values titles because of ASCII requirement for CRAN</p>
<code>time_filter_fields_titles</code>	character, title names of time filter fields in filters list, currently only default values are allowed
<code>time_fields_titles_in_df</code>	character, title names of time related columns in data_df (result of <code>fedstat_parse_sdmx_to_table</code> ), currently only default values are allowed
<code>filter_value_title_alias_lookup_table</code>	data.frame with columns <code>filter_value_title</code> and <code>filter_value_title_alias</code> . Used to replace <code>filter_value_title</code> with standard forms of filter value titles. It is mainly used to set consistent names. For example, the Dalnevostochnyj federalnyj okrug ( s 03.11.2018) (transliteration for CRAN) in <code>filter_value_title</code> can be simply replaced with <code>filter_value_title_alias</code> as Dalnevostochnyj federalnyj okrug. In this example in fact, these are two different entities (after the inclusion of Buryatia and Transbaikalia and before), but in most cases such a replacement is more convenient for loading data, since you need the whole time series. By default it's empty data.frame, e.g. no replacement for anything.
<code>timeout_seconds</code>	numeric, maximum time before a new GET and POST request is tried
<code>retry_max_times</code>	numeric, maximum number of tries to GET and POST data_ids
<code>disable_warnings</code>	bool, enables or disables following warnings: <ol style="list-style-type: none"> <li>About non matched <code>filter_value_title</code> in filters and <code>filter_value_title</code> from data_ids;</li> <li>About unspecified <code>filter_field_title</code> in filters.</li> </ol>
<code>httr_verbose</code>	<code>httr::verbose()</code> or NULL, outputs messages to the console about the processing of the request

**Value**

bool, TRUE if indicator data has been updated, FALSE if not

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 week prices
# standardize names for DVFO and extract week numbers
# filter the data_ids to get data for week 21 and 22 of 2021
# for all goods and services for Russian Federation
data_ids_filtered <- fedstat_get_data_ids("37426") %>%
  fedstat_get_data_ids_special_cases_handle(
    filter_value_title_alias_lookup_table = data.frame(
      "filter_value_title" = "Dalnevostochnyj federalnyj okrug ( s 03.11.2018)",
      "filter_value_title_alias" = "Dalnevostochnyj federalnyj okrug"
    )
  ) %>%
  fedstat_data_ids_filter(
    filters = list(
      "Territory" = "Russian Federation",
      "Year" = "2021",
      "Period" = c(21, 22),
      "Types of goods and services" = "*"
    )
  )

# In this example names for Far Eastern Federal District are latinized for CRAN
# 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 field reflects the individual property specific value of the data point. (e.g. 2021 for Year, "Russian Federation" for 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. asterisk (\*), alias for "select all possible filter values for this filter field"

Unspecified filters use asterisk 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(),
  filter_value_title_alias_lookup_table = data.frame(filter_value_title = character(),
    filter_value_title_alias = character(), stringsAsFactors = FALSE),
  timeout_seconds = 180,
  retry_max_times = 3,
  disable_warnings = FALSE,
  httr_verbose = httr::verbose(data_out = FALSE)
)
```

## 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 `httr::GET` and `httr::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



**filter\_value\_title\_alias\_lookup\_table**  
 data.frame with columns `filter_value_title` and `filter_value_title_alias`.  
 Used to replace `filter_value_title` with standard forms of filter value titles.  
 It is mainly used to set consistent names. For example, the Dalnevostochnyj federalnyj okrug ( s 03.11.2018) (transliteration for CRAN) in `filter_value_title` can be simply replaced with `filter_value_title_alias` as Dalnevostochnyj federalnyj okrug. In this example in fact, these are two different entities (after the inclusion of Buryatia and Transbaikalia and before), but in most cases such a replacement is more convenient for loading data, since you need the whole time series. By default it's empty data.frame, e.g. no replacement for anything.

**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 `filter_field_title` in filters.

**httr\_verbose** `httr::verbose()` or NULL, outputs messages to the console about the processing of the request

**Value**

data.frame with filtered indicator data from fedstat.ru

**See Also**

[fedstat\\_get\\_data\\_ids](#), [fedstat\\_get\\_data\\_ids\\_special\\_cases\\_handle](#), [fedstat\\_data\\_ids\\_filter](#), [fedstat\\_post](#)

**Examples**

```
## Not run:
# Download weekly goods and services prices data for week 21 and 22 of 2021
# for all goods and services for Russian Federation
data <- fedstat_data_load_with_filters(
  indicator_id = "37426",
  filters = list(
    "Territory" = "Russian Federation",
    "Year" = "2021",
    "Period" = c(21, 22),
    "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 = httr::verbose(data_out = FALSE)
)
```

### Arguments

indicator_id	character, indicator id/code from indicator URL. For example for indicator with URL <a href="https://www.fedstat.ru/indicator/37426">https://www.fedstat.ru/indicator/37426</a> indicator id will be 37426
...	other arguments passed to httr::GET
timeout_seconds	numeric, maximum time before a new GET request is tried
retry_max_times	numeric, maximum number of tries to GET data_ids
httr_verbose	httr::verbose() or NULL, 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);
6. `filter_field_object_ids_order` - sorting for `filter_field_object_ids`, determines the order of the filters fields.

## See Also

[fedstat\\_get\\_data\\_ids\\_special\\_cases\\_handle](#), [fedstat\\_data\\_ids\\_filter](#), [fedstat\\_post\\_data\\_ids\\_filtered](#)

## Examples

```
## Not run:
# Get data filters identifiers for week prices
data_ids <- fedstat_get_indicator_data_ids("37426")

## End(Not run)
```

---

```
fedstat_get_data_ids_special_cases_handle
```

```
    Handle special cases strings for filter_value_title col in
    data_ids
```

---

## Description

Handles special cases strings in `filter_value_title` to standardize them. Currently does only 2 things:

1. Replaces `filter_value_title` with given aliases in `filter_value_title_alias_lookup_table`;

2. Replaces week period titles with week numbers to filter by it instead of a complex unstandardized "period" string and adds a new column `filter_value_title_week` for weekly data for using original week title if needed;

Function will be supplemented with new methods for processing special cases as they are found

### Usage

```
fedstat_get_data_ids_special_cases_handle(
  data_ids,
  filter_value_title_alias_lookup_table = data.frame(filter_value_title = character(),
    filter_value_title_alias = character(), stringsAsFactors = FALSE)
)
```

### Arguments

`data_ids` data.frame, result of `fedstat_get_data_ids`

`filter_value_title_alias_lookup_table` data.frame with columns `filter_value_title` and `filter_value_title_alias`. Used to replace `filter_value_title` with standard forms of filter value titles. It is mainly used to set consistent names. For example, the Dalnevostochnyj federalnyj okrug ( s 03.11.2018) (transliteration for CRAN) in `filter_value_title` can be simply replaced with `filter_value_title_alias` as Dalnevostochnyj federalnyj okrug. In this example in fact, these are two different entities (after the inclusion of Buryatia and Transbaikalia and before), but in most cases such a replacement is more convenient for loading data, since you need the whole time series. By default it's empty data.frame, e.g. no replacement for anything.

### Value

data.frame, `data_ids` with replaced by aliases `filter_value_title` column and a new column `filter_value_title_week`, which consists only of NA in case of non-weekly data

### See Also

[fedstat\\_get\\_data\\_ids](#)

### Examples

```
## Not run:
# Get data filters identifiers for week prices
# and standardize names for DVFO and extract week numbers
data_ids_special_cases_handled <- fedstat_get_data_ids("37426") %>%
  fedstat_get_indicator_data_ids_special_cases_handle(
    filter_value_title_alias_lookup_table = data.frame(
      "filter_value_title" = "Dalnevostochnyj federalnyj okrug ( s 03.11.2018)",
      "filter_value_title_alias" = "Dalnevostochnyj federalnyj okrug"
    )
  )
# In this example names for Far Eastern Federal District are latinized for CRAN
```

```
## 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\_java\_script\_data\_ids\_parse\_to\_json  
*Get data ids from java script source*

---

**Description**

Get data ids from java script source

**Usage**

```
fedstat_java_script_data_ids_parse_to_json(java_script_source_code)
```

**Arguments**

java\_script\_source\_code  
character, java script source code with data ids

**Value**

json in form of list with data ids

---

fedstat\_java\_script\_default\_data\_ids\_object\_ids\_parse\_to\_json  
*Get default data ids object ids from java script source*

---

**Description**

Get default data ids object ids from java script source

**Usage**

```
fedstat_java_script_default_data_ids_object_ids_parse_to_json(  
  java_script_source_code  
)
```

**Arguments**

java\_script\_source\_code  
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

---

fedstat\_parse\_all\_indicators\_names

*Parses https://fedstat.ru/organizations/ indicators hierarchical list into data.frame for reference*

---

**Description**

Parses https://fedstat.ru/organizations/ indicators hierarchical list into data.frame for reference

**Usage**

```
fedstat_parse_all_indicators_names(  
  url = "https://fedstat.ru/organizations/",  
  ...  
)
```

**Arguments**

url	https://fedstat.ru/organizations/
...	passed to httr::GET

**Value**

data.frame, database of all indicators on fedstat.ru with hierarchical structure

---

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

**Usage**

```
fedstat_parse_sdmx_to_table(data_raw)
```

**Arguments**

data_raw	sdmx raw bytes
----------	----------------

**Value**

data.frame

**See Also**[fedstat\\_parse\\_sdmx\\_to\\_table](#)**Examples**

```
## Not run:
# Get data filters identifiers for week prices
# standardize names for DVFO and extract week numbers
# filter the data_ids to get data for week 21 and 22 of 2021
# 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("37426") %>%
  fedstat_get_data_ids_special_cases_handle(
    filter_value_title_alias_lookup_table = data.frame(
      "filter_value_title" = "Dalnevostochnyj federalnyj okrug ( s 03.11.2018)",
      "filter_value_title_alias" = "Dalnevostochnyj federalnyj okrug"
    )
  ) %>%
  fedstat_data_ids_filter(
    filters = list(
      "Territory" = "Russian Federation",
      "Year" = "2021",
      "Period" = c(21, 22),
      "Types of goods and services" = "*"
    )
  ) %>%
  fedstat_post_data_ids_filtered() %>%
  fedstat_parse_sdmx_to_table()

# In this example names for DVFO are latinized for CRAN
# 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](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 writeLines 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 = httr::verbose(data_out = FALSE)
)

```

**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_handle</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 <code>sdmx</code> , <code>excel</code>
<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 week prices
# standardize names for DVFO and extract week numbers
# filter the data_ids to get data for week 21 and 22 of 2021
# for all goods and services for Russian Federation
# POST filters and download data in sdmx format
data <- fedstat_get_data_ids("37426") %>%
  fedstat_get_data_ids_special_cases_handle(
    filter_value_title_alias_lookup_table = data.frame(
      "filter_value_title" = "Dalnevostochnyj federalnyj okrug ( s 03.11.2018)",
      "filter_value_title_alias" = "Dalnevostochnyj federalnyj okrug"
    )
  ) %>%
  fedstat_data_ids_filter(
    filters = list(

```

```

    "Territory" = "Russian Federation",
    "Year" = "2021",
    "Period" = c(21, 22),
    "Types of goods and services" = "*"
  )
) %>%
fedstat_post_data_ids_filtered()

# In this example names for DVFO are latinized for CRAN
# Not actual filter field titles and filter values titles because of ASCII requirement for CRAN

## End(Not run)

```

---

```
fedstat_prepare_reference_data_for_check_data_update
```

*Loads the data that will be used to compare with the new data to detect data updates*

---

## Description

Used only as a first argument in [fedstat\\_check\\_data\\_update](#). Stores the original filters, `data_ids`, `data_df` and other arguments. Combines it all into one object, which ensures that filters and other arguments are identical for old and new data queries.

You *shouldn't* specify time filters in `filters` list. This function does this automatically, it takes all PERIODs and the last year.

Please try to specify filters as much as possible (for example, using filters only for Russia and a certain type of indicator/good/service, etc.), it will reduce the query execution time and server load on `fedstat.ru`. You don't need all the data to check the data updates anyway, one observation with the new date is enough. However, keep in mind that you need to choose observations that are sure to be in the new data release.

## Usage

```

fedstat_prepare_reference_data_for_check_data_update(
  indicator_id,
  ...,
  filters = list(),

  time_filter_fields_titles = c("\u041f\u0435\u0440\u0438\u043e\u0434",
    "\u0413\u043e\u0434\u0434"),
  time_fields_titles_in_df = c("PERIOD", "Time"),
  filter_value_title_alias_lookup_table = data.frame(filter_value_title = character(),
    filter_value_title_alias = character(), stringsAsFactors = FALSE),
  timeout_seconds = 180,
  retry_max_times = 3,
  disable_warnings = FALSE,
  httr_verbose = httr::verbose(data_out = FALSE)
)

```

**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 `httr::GET` and `httr::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

`time_filter_fields_titles` character, title names of time filter fields in `filters` list, currently only default values are allowed

`time_fields_titles_in_df` character, title names of time related columns in `data_df` (result of `fedstat_parse_sdmx_to_table`), currently only default values are allowed

`filter_value_title_alias_lookup_table` `data.frame` with columns `filter_value_title` and `filter_value_title_alias`. Used to replace `filter_value_title` with standard forms of filter value titles. It is mainly used to set consistent names. For example, the Dalnevostochnyj federalnyj okrug ( s 03.11.2018) (transliteration for CRAN) in `filter_value_title` can be simply replaced with `filter_value_title_alias` as Dalnevostochnyj federalnyj okrug. In this example in fact, these are two different entities (after the inclusion of Buryatia and Transbaikalia and before), but in most cases such a replacement is more convenient for loading data, since you need the whole time series. By default it's empty `data.frame`, e.g. no replacement for anything.

`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 `filter_field_title` in `filters`.

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

**Value**

list with the following fields: `indicator_id`, `filters`, `time_filter_fields_titles`, `time_fields_titles_in_df`, `filter_value_title_alias_lookup_table`, `reference_data_ids_unfiltered_special_cases_handled`, `reference_data_df`

**Examples**

```
## Not run:
reference_data <- fedstat_prepare_reference_data_for_check_data_update(
  indicator_id = "37426",
  filters = list(
    "Territory" = "Russian Federation",
    "Types of goods and services" = "Sahar-pesok, kg"
  )
)
# Not actual filter field titles and filter values titles because of ASCII requirement for CRAN

## End(Not run)
```

# Index

## \* datasets

- fedstat\_indicators\_names\_database,  
[13](#)
  
- fedstat\_check\_data\_update, [2](#), [18](#)
- fedstat\_check\_data\_update\_, [4](#)
- fedstat\_data\_ids\_filter, [6](#), [9](#), [11](#)
- fedstat\_data\_load\_with\_filters, [7](#)
- fedstat\_get\_data\_ids, [7-9](#), [10](#), [12](#)
- fedstat\_get\_data\_ids\_special\_cases\_handle,  
[9](#), [11](#), [11](#)
- fedstat\_indicators\_names\_database, [13](#)
- fedstat\_java\_script\_data\_ids\_parse\_to\_json,  
[14](#)
- fedstat\_java\_script\_default\_data\_ids\_object\_ids\_parse\_to\_json,  
[14](#)
- fedstat\_parse\_all\_indicators\_names, [15](#)
- fedstat\_parse\_sdmx\_to\_table, [9](#), [15](#), [16](#),  
[17](#)
- fedstat\_post\_data\_ids\_filtered, [7](#), [9](#), [11](#),  
[16](#)
- fedstat\_prepare\_reference\_data\_for\_check\_data\_update,  
[3](#), [18](#)
- fedstatAPIr (fedstatAPIr-package), [2](#)
- fedstatAPIr-package, [2](#)