Package: fedstatAPIr (via r-universe)

October 29, 2026

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

Version 2.0.0

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

Roxygen list(markdown = TRUE)

Imports httr, jsonlite, xml2, readsdmx, magrittr, utils, stats, tools, data.table

Depends R (>= 3.5.0)

Config/roxygen2/version 8.1.0

RoxygenNote 7.2.3

Config/pak/sysreqs libxml2-dev libssl-dev

Repository https://denchpokepon.r-universe.dev

Date/Publication 2026-10-29 08:08:25 UTC

RemoteUrl https://github.com/denchpokepon/fedstatapir

RemoteRef HEAD

RemoteSha b6cf1cafb32beefc14449529f58aca731b08d1df

Contents

fedstat_cleanup 2
fedstat_collect 3
fedstat_data_ids_filter 4

fedstat_data_load_with_filters 5
fedstat_dictionary 8
fedstat_download 9
fedstat_get_data_ids 13
fedstat_indicator_info 15
fedstat_indicators_names_database 15
fedstat_parse_sdmx_to_table 16
fedstat_post_data_ids_filtered 17
fedstat_show_filters 19
parse_js1 20
parse_js2 21

Index **22**

`fedstat_cleanup` *Free the disk space used by a finished download*

Description

Deletes the saved parts of a complete `fedstat_download()`. Save the table from `fedstat_collect()` first: after cleanup the download cannot be collected or continued again.

Usage

```
fedstat_cleanup(x)
```

Arguments

x	A `fedstat_download()` result or its folder.

Details

Incomplete, changed or actively used downloads are not cleaned. Cleanup can be repeated if it was interrupted.

Value

Invisibly, the download folder. Only the part files listed in the manifest are removed; the manifest, diagnostic log and any other files stay.

See Also

`fedstat_download()`, `fedstat_collect()`

Examples

```
## Not run:
data <- fedstat_collect("cpi-download")
saveRDS(data, "cpi.rds")
fedstat_cleanup("cpi-download")

## End(Not run)
```

`fedstat_collect` *Get the table from a finished download*

Description

Combines the saved parts of a complete `fedstat_download()` into one table. It works offline and can be repeated until you remove the parts with `fedstat_cleanup()`.

Columns: the indicator value in `ObsValue`; one column per filter with its name, and a matching `_code` column with Fedstat's classification code; time columns such as `Time`. If Fedstat renamed a category during the download, every row uses the newest name. If Fedstat stopped offering some selected values, a message lists them.

Usage

```
fedstat_collect(x, remove_parts = FALSE, try_to_parse_ObsValue = TRUE)
```

Arguments

`x`	A `fedstat_download()` result or its folder.
`remove_parts`	If `TRUE`, delete the saved parts after the table has been built and checked. Default `FALSE`. The table then exists only in memory until you save it; to save first and remove parts later, use `fedstat_cleanup()`.
`try_to_parse_ObsValue`	If `TRUE` (default), return `ObsValue` as numbers: decimal commas are read as decimal points and empty values become `NA`. If any other value is not a number, collection stops with examples and nothing is removed; use `FALSE` to get every value as text.

Value

A data.frame. The whole table is loaded into memory; for very large downloads read the individual part files listed in `manifest.rds` instead. Saved parts keep the original text values.

See Also

`fedstat_download()`, `fedstat_cleanup()`

Examples

```
## Not run:
data <- fedstat_collect("cpi-download")
saveRDS(data, "cpi.rds")

## End(Not run)
```

`fedstat_data_ids_filter`

Filters `data_ids` *based on* `filters` *that are given in JSON format*

Description

For advanced use. Most downloads should use `fedstat_download()`, which splits large selections, pauses between requests and saves progress.

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"

All filter fields with more than one possible value must be explicitly specified in `filters`. Use `"*"` to select all values for a filter field. Filter fields with only one possible value are auto-selected.

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 warnings about non matched filter_value_title in filters and filter_value_title from data_ids.

Value

data.frame, filtered data_ids

See Also

fedstat_get_data_ids, fedstat_post_data_ids_filtered

Examples

```
## Not run:
# Get data filters identificators 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

(Deprecated) Download indicator data from fedstat.ru in one request

---

Description

Deprecated: use fedstat_download() and fedstat_collect(). This function sends the whole selection to Fedstat as one request, with no pauses or saved progress. Large selections are slow, often fail and put heavy load on Fedstat. It is kept so that existing scripts keep working. Replace data <- fedstat_data_load_with_filters("31074", filters = filters) with job <- fedstat_download("31074", filters = filters) and data <- fedstat_collect(job).

Size is the number of filter combinations: the product of the number of selected values of every filter. Up to 50,000 combinations, a deprecation message is shown once per session. Above 50,000,

every call warns. Above 100,000, the function stops unless `allow_large_requests = TRUE`, and above 250,000 it always stops: such requests load Fedstat for all its users and rarely succeed.

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"

All filter fields with more than one possible value must be explicitly specified in `filters`. Use `"*"` to select all values. Unspecified multi-value filters will produce an error.

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,
  httr_verbose = NULL,
  loading_steps_verbose = TRUE,
  return_type = c("data", "dictionary"),
  try_to_parse_ObsValue = TRUE,
  allow_large_requests = 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

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

`retry_max_times`
: numeric, maximum pipeline attempts, including the first. Only transient failures and one session repair are retried. Filter metadata are retained after a successful GET; retries do not nest.

`disable_warnings`
: bool, enables or disables warnings about non matched `filter_value_title` in `filters` and `filter_value_title` from `data_ids`.

`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). `fedstat_dictionary()` gets the dictionary without choosing filters.

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

`allow_large_requests`
: logical. `TRUE` allows selections of up to 250,000 filter combinations instead of 100,000. Larger selections are always refused; use `fedstat_download()` for them.

Value

data.frame with filtered indicator data from fedstat.ru

See Also

`fedstat_download`, `fedstat_get_data_ids`, `fedstat_data_ids_filter`, `fedstat_post_data_ids_filtered`, `fedstat_parse_sdmx_to_table`

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_dictionary` *Download the code lists (dictionary) of an indicator*

Description

Returns the classifications Fedstat uses in the data of an indicator: for every coded column, all its codes and their names. The data from `fedstat_collect()` has a column with names (such as `s_grtov`) and a column with codes (`s_grtov_code`) for each classification; this table lists every code, including ones absent from your download.

Fedstat sends these lists only together with data, so the function downloads a small piece of data and keeps only its code lists. Any piece will do: every export carries the complete lists. It first tries the selection the website shows when the indicator page opens, and then small random selections if that one is empty.

Usage

```
fedstat_dictionary(
  indicator_id,
  filters = NULL,
  ...,
  attempts = 5L,
  timeout_seconds = 180,
  retry_max_times = 3L,
  request_pause = c(10, 20),
  verbose = TRUE
)
```

Arguments

`indicator_id`	Indicator code from its page address, for example `"31074"`.
`filters`	Optional selection to request, as in `fedstat_download()`. Useful when the automatic selections return no data: choose values that show data on the website. Then only this selection is tried.
`...`	httr settings applied to every request, for example `httr::user_agent("my-project")`. Arguments after `...` must be named in full.

`attempts`	Most requests for data, including retries after network errors.
`timeout_seconds`	Longest wait for one request, in seconds.
`retry_max_times`	Attempts to open the indicator page, including the first.
`request_pause`	Seconds between attempts: one number, or two numbers for a random pause in that range.
`verbose`	`TRUE` shows short progress messages.

Details

The code lists come from the SDMX export, not from the indicator page. Names usually match the filter values on the page, but codes are a different system from the IDs in `fedstat_get_data_ids()`. Year, period and unit of measurement are not coded and are not listed.

Each try is one small request (at most 5,000 filter combinations), with a pause between tries.

Value

A data.frame with one row per code: `field_id` (the column name in the data), `field_title` (the classification name), `value_id` (the code) and `value_title` (its name).

See Also

`fedstat_show_filters()` for the filters of the indicator page.

Examples

```
## Not run:
dictionary <- fedstat_dictionary("31074")
# All codes of one classification
dictionary[dictionary$field_id == "s_grtov", c("value_id", "value_title")]

## End(Not run)
```

---

fedstat_download *Download Fedstat data in small parts that survive interruptions*

Description

Downloads one indicator for the filter values you choose and saves it piece by piece in `cache_dir`. Fedstat builds every combination of the selected values (years x months x territories x products ...), so a large selection is split into parts of at most `max_nominal_cells_request` combinations. Parts are requested one at a time, with pauses in between. This keeps the load on Fedstat light and makes long downloads finish.

If a run stops (time is up, Fedstat is unavailable, R was closed), run the same call again: saved parts are skipped. When the download is complete, `fedstat_collect()` returns the table.

Usage

```
fedstat_download(
  indicator_id,
  filters = list(),
  cache_dir = NULL,
  ...,
  max_nominal_cells_request = 50000,
  allow_large_requests = FALSE,
  preview_first = FALSE,
  total_time_limit_seconds = 3600,
  timeout_seconds = 180,
  retry_max_times = 2L,
  request_pause = c(30, 60),
  verbose = TRUE,
  log_file = file.path(cache_dir, "download.log"),
  token_policy = c("reuse", "refresh"),
  max_depth = 20L,
  max_failures = 6L,
  recovery_max_times = 3L,
  recovery_pause = 60,
  access_pause = 300
)
```

Arguments

indicator_id	Indicator code from its page address, for example "31074" for https://www.fedstat.ru/indicator/3
filters	A named list: filter names and the values to download, both in Russian, as shown on Fedstat. "*" selects every value of a filter. Every filter with more than one value must be listed. Case and extra spaces are ignored. The single string "*" instead of a list selects everything.
cache_dir	Folder for the saved parts, created if needed. Use one folder per download and keep it until you have collected and saved the table. If omitted, a folder such as fedstat-31074-1a2b3c4d is created in the current folder: its name depends on the indicator and the filters, so running the same call again continues in the same folder. The full path is shown.
...	For advanced users: httr settings applied to every request, for example httr::user_agent("my-project" or httr::config(connecttimeout = 20). Any other unnamed or unknown argument is an error. Arguments after ... must be named in full.
max_nominal_cells_request	Largest number of filter combinations in one request. Default 50,000. Above 50,000 you get a warning, above 100,000 allow_large_requests = TRUE is required, and 250,000 is the maximum.
allow_large_requests	TRUE turns off the warning for max_nominal_cells_request above 50,000 and allows up to 250,000.
preview_first	If TRUE, ask Fedstat to show each part's table first, as its website does when you press "Apply", and then download it. Fedstat sometimes only returns data after

this step. Default FALSE: it adds a request per attempt and can roughly double the download time.

total_time_limit_seconds
: Time for this run in seconds, including pauses; default one hour. Run the same call again to continue.

timeout_seconds
: Longest wait for one request, in seconds.

retry_max_times
: Attempts per part before it is split or the download pauses, including the first.

request_pause
: Seconds between requests: one number, or two numbers for a random pause in that range.

verbose
: TRUE (default) shows short progress messages, FALSE hides them, "detailed" shows every event. Warnings and errors are always shown.

log_file
: Diagnostic log, by default download.log inside cache_dir; NULL turns it off. Attach it when reporting a problem.

token_policy
: "reuse" (default) keeps the download session's token until Fedstat rejects it; "refresh" gets a new one before every request, which costs an extra page request each time.

max_depth
: How many times a part may be split again.

max_failures
: Failed data requests in a row before the download pauses to let Fedstat recover.

recovery_max_times
: How many recovery pauses are allowed since the last saved part before the run stops. 0 stops after ordinary retries.

recovery_pause
: First recovery pause in seconds; it doubles each time.

access_pause
: Seconds to wait when Fedstat seems to check or refuse access; default five minutes, doubling if it happens again. No check is bypassed.

Value

A fedstat_download object: the folder (path), whether the download is complete, a table of parts, and the values Fedstat stopped offering during the download (unavailable). The data stay on disk; use fedstat_collect() to load them. If the run stops, an error is raised and saved parts are kept for the next run.

Resuming

Run the same call with the same filters and cache_dir. Filters may be reordered or differ in case and spacing. A different selection needs a different folder. "*" means the values offered when the download started: values Fedstat adds later are not included. Each run has its own time allowance; saving a part that has already arrived may run slightly over it. Only one R session can use a folder at a time. If R crashed, the folder can stay marked as in use; the error message says how to clear it.

Request size

Size is counted in filter combinations, the product of the number of selected values of every filter, not in rows: many combinations can be empty. A part over the limit is split along its largest filter into

the fewest groups that fit. Fedstat checks every combination, even empty ones, so large requests load it for all its users and often fail. Every function that requests data uses the same limits: a warning above 50,000 combinations, `allow_large_requests = TRUE` needed above 100,000, and no request above 250,000.

Reliability

Requests are sequential and paced. A failed request is retried after a pause. A part is halved only when its size is the likely cause: a request timeout, a gateway timeout (HTTP 504), an export marked as truncated, or a refusal that persists after renewing the session. Dropped or refused connections, server errors and rate limits lead to a recovery pause instead. Splits made after failures are undone on the next run if none of their parts was saved. Fedstat answers some empty selections with a redirect; a part counts as empty only when Fedstat's table preview confirms zero rows.

When Fedstat changes its filters

The selected filters are compared with the indicator page at the start of every run, when the session is renewed and before completion. Filter and value IDs identify what is requested. If Fedstat adds or removes a whole filter, the download stops: start again in a new folder. A selected value that Fedstat no longer offers is left out of unfinished parts, with a warning, and listed in `unavailable`. Renamed values continue with a warning, and `fedstat_collect()` uses the newest name for each code. A changed row/column layout on the website is ignored. If an export contains values that were not requested, the download stops. A change in meaning that keeps the same IDs and names cannot be detected.

Completion

A download is complete when every part has been received and checked, and the saved parts together contain no duplicate observations. This check needs some free disk space in `cache_dir` but never loads the whole table. A completed download opens again without network access. A long download is not a snapshot of one moment: Fedstat may publish revisions while it runs. Saved parts keep values exactly as Fedstat sent them.

Diagnostic log

`download.log` is a JSON Lines file with request stages, timings, counts and recovery decisions. It never contains HTTP headers, cookies, tokens or response bodies.

See Also

`fedstat_collect()`, `fedstat_cleanup()`

Examples

```
## Not run:
job <- fedstat_download("31074", filters = "*", cache_dir = "cpi-download")
data <- fedstat_collect(job)

# Filter names as on the indicator page (English placeholders: CRAN needs ASCII)
filters <- list("Year" = "2024", "Period" = "January", "Territory" = "*")
```

```
# Without cache_dir, a folder is chosen for you and its path is shown
job <- fedstat_download("31074", filters)

## End(Not run)
```

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

Description

For advanced use. Most downloads should use `fedstat_download()`, which splits large selections, pauses between requests and saves progress.

To query data from fedstat we need to POST some filters in form of filter numeric identificators. 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

`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
`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 `https://www.fedstat.ru/indicator/downloadData.do?format={format}` for all indicators and is often quite fast.

Token lifetime is controlled by Fedstat. Reuse may succeed, but tokens can be rejected. The wrapper `fedstat_data_load_with_filters` refreshes refused tokens while retaining the resolved filters.

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.

The returned data.frame also carries session and CSRF token metadata as attributes (`fedstat_handle`, `fedstat_base_url`, `fedstat_csrf_token`, `fedstat_csrf_token_name`, `fedstat_indicator_id`). The `fedstat_handle` attribute holds an `httr::handle` object that preserves session cookies for the downstream POST request. These are used internally by `fedstat_post_data_ids_filtered` and must not be removed. A shared in-memory session attribute keeps refreshed tokens consistent across filtered copies. Metadata can be reused; session handles are not a disk cache.

The `fedstat_default_selection` attribute lists the values the website selects when the indicator page opens, as `"<filter_field_id>_<filter_value_id>"`. `fedstat_show_filters()` prints the filters in a readable form.

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_show_filters`, `fedstat_data_ids_filter`, `fedstat_post_data_ids_filtered`

Examples

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

## End(Not run)
```

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
indicators <- fedstat_indicator_info()

## 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. The database can be refreshed by the user via `fedstat_indicator_info()`.

Usage

```
fedstat_indicators_names_database
```

Format

A data frame with the following variables:

department the name of the department from which the data comes

group thematic grouping of the indicator

id indicator id (used in URL and API calls)

title indicator name/title

hidden boolean, TRUE if indicator is hidden on the website

Details

The version shipped with the package was last refreshed on 2026-09-04.

Source

https://fedstat.ru/organizations/

`fedstat_parse_sdmx_to_table` *Parse sdmx raw bytes to data.frame*

Description

For advanced use. Most downloads should use `fedstat_download()`, which splits large selections, pauses between requests and saves progress.

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 or a path to an existing SDMX XML file
`return_type`	character, "data" or "dictionary", data for actual data, dictionary for sdmx lookup table (full data codes dictionary). `fedstat_dictionary()` gets the dictionary without choosing filters.
`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 identificators 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

For advanced use. Most downloads should use fedstat_download(), which splits large selections, pauses between requests and saves progress.

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

The request includes a CSRF token that is automatically extracted by fedstat_get_data_ids and passed through via data.frame attributes. Token reuse may succeed, but Fedstat can reject a token at any time. This primitive sends one POST; the convenience wrapper and batch downloader own retry and token-refresh decisions.

If no CSRF token is found in data_ids attributes (e.g. when using cached data_ids from a previous session), the function will attempt to fetch a fresh token automatically, using the same HTTP configuration.

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

Request size is limited to protect Fedstat. Size is the number of filter combinations: the product of the number of selected values of every filter. Above 50,000 combinations this function warns, above 100,000 it stops unless `allow_large_requests = TRUE`, and above 250,000 it always stops. `fedstat_download()` splits larger selections into parts.

Usage

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

Arguments

`data_ids`	data.frame, a result of `fedstat_data_ids_filter` to download a subset of the available data (all data of an indicator usually exceed the size limit)
`...`	other arguments passed to httr::POST
`data_format`	string, one of sdmx, excel
`timeout_seconds`	numeric, maximum time before POST request times out
`retry_max_times`	numeric, not used for POST (exports are not retried here), but used for fallback GET if CSRF token needs to be fetched
`httr_verbose`	`httr::verbose()` or NULL, outputs messages to the console about the processing of the request
`allow_large_requests`	logical. `TRUE` turns off the warning above 50,000 filter combinations and allows up to 250,000. Larger requests are always refused.

Value

raw bytes (sdmx or excel)

See Also

`fedstat_parse_sdmx_to_table`

Examples

```
## Not run:
# Get data filters identificators 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)
```

`fedstat_show_filters` *Show the filters of an indicator, ready to copy into your code*

Description

Prints every filter of an indicator with the number of its values and a few examples, followed by a `filters <- list(...)` you can copy into `fedstat_download()`. That list repeats the selection the website shows when the indicator page opens; edit it to choose what you need.

Usage

```
fedstat_show_filters(
  indicator_id,
  ...,
  timeout_seconds = 180,
  retry_max_times = 3L
)
```

Arguments

`indicator_id`	Indicator code from its page address, for example `"31074"`, or the result of `fedstat_get_data_ids()`.
`...`	httr settings applied to the page request, for example `httr::user_agent("my-project")`. Arguments after `...` must be named in full.
`timeout_seconds`	Longest wait for the page, in seconds.
`retry_max_times`	Attempts to open the page, including the first.

Value

Invisibly, the printed `filters` list. Filters with only one value are left out: they are selected automatically.

See Also

`fedstat_dictionary()` for the codes used in the data.

Examples

```
## Not run:
filters <- fedstat_show_filters("31074")
job <- fedstat_download("31074", filters)

# All values of one filter
ids <- fedstat_get_data_ids("31074")
unique(ids[c("filter_field_title", "filter_value_title")])

## 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, 15

fedstat_cleanup, 2
fedstat_cleanup(), *3*, *12*
fedstat_collect, 3
fedstat_collect(), *2*, *5*, *8*, *9*, *11*, *12*
fedstat_data_ids_filter, 4, *7*, *14*
fedstat_data_load_with_filters, 5
fedstat_dictionary, 8
fedstat_dictionary(), *7*, *16*, *20*
fedstat_download, *7*, 9
fedstat_download(), *2–5*, *7*, *8*, *13*, *16–19*
fedstat_get_data_ids, *5–7*, 13, *15*
fedstat_get_data_ids(), *9*, *19*
fedstat_indicator_info, 15
fedstat_indicators_names_database, 15
fedstat_parse_sdmx_to_table, *7*, 16, *17*, *18*
fedstat_post_data_ids_filtered, *5*, *7*, *14*, 17
fedstat_show_filters, *14*, 19
fedstat_show_filters(), *9*, *14*

parse_js1, 20
parse_js2, 21