

# Package: RJSDMX (via r-universe)

October 15, 2024

**Version** 3.4-0

**Title** R Interface to SDMX Web Services

**Maintainer** Attilio Mattiocco <attilio.mattiocco@bancaditalia.it>

**Description** Provides functions to retrieve data and metadata from providers that disseminate data by means of SDMX web services. SDMX (Statistical Data and Metadata eXchange) is a standard that has been developed with the aim of simplifying the exchange of statistical information. More about the SDMX standard and the SDMX Web Services can be found at: <<https://sdmx.org>>.

**Depends** R (>= 4.2.0), rJava (>= 0.8-8), zoo (>= 1.6-4)

**SystemRequirements** Java (>= 8)

**License** EUPL

**URL** <https://github.com/amattioc/SDMX/>

**BugReports** <https://github.com/amattioc/SDMX/issues>

**Copyright** Banca d'Italia

**NeedsCompilation** no

**Author** Attilio Mattiocco [aut, cre]

**Repository** CRAN

**Date/Publication** 2024-09-13 09:50:07 UTC

## Contents

RJSDMX-package	2
addProvider	3
getCodes	3
getDimensions	4
getDSDIdentifier	5
getFlows	5
getProviders	6
getTimeSeries	7

getTimeSeries2 . . . . .	7
getTimeSeriesRevisions . . . . .	9
getTimeSeriesTable . . . . .	10
getTimeSeriesTable2 . . . . .	10
sdmxdf . . . . .	12
sdmxHelp . . . . .	13
setProviderCredentials . . . . .	13

## Index 14

---

RJSDMX-package	<i>Gets timeseries from SDMX data Provider</i>
----------------	--

---

## Description

This package provides functions to extract timeseries data and structural metadata from an SDMX Provider (e.g. ECB,OECD, EUROSTAT) via SDMX Web Service

## Details

Package: RJSDMX  
Type: Package

The SDMX Connectors framework (of which RJSDMX is part) aims to offer data users the means for efficiently interacting with SDMX Web Service providers from within the most popular statistical tools. The source code of the SDMX Connectors project can be found at:

<https://github.com/amattioc/SDMX>

Information about the R Connector can be found in the dedicated wiki page:

<https://github.com/amattioc/SDMX/wiki/RJSDMX:-Connector-for-R>

In particular, all information related to configuration (network, tracing, security) can be found at:

<https://github.com/amattioc/SDMX/wiki/Configuration>

## Author(s)

Attilio Mattiocco, Bank of Italy <attilio.mattiocco@bancaditalia.it>

## See Also

**getProviders, getTimeSeries, sdmxHelp**

## Examples

```
## Not run:
my_ts = getTimeSeries('ECB', 'EXR.M.USD.EUR.SP00.A')

## End(Not run)
```

---

addProvider	<i>add new provider</i>
-------------	-------------------------

---

**Description**

Configure a new data provider (only SDMX 2.1 REST providers are supported). This function can be used to configure a new (SDMX 2.1 compliant, REST based) data provider.

**Usage**

```
addProvider(name, endpoint, needsCredentials = FALSE, needsURLEncoding = FALSE,
  supportsCompression = TRUE, description = "", sdmxVersion = "V2")
```

**Arguments**

name	the name of the provider
endpoint	the URL where the provider resides
needsCredentials	set this to TRUE if the user needs to authenticate to query the provider
needsURLEncoding	set this to TRUE if the provider does not handle character '+' in URLs
supportsCompression	set this to TRUE if the provider is able to handle compression
description	a brief text description of the provider
sdmxVersion	what sdmx version this provider supports

**Examples**

```
## Not run:
addProvider('test', 'http://sdw-wsrest.ecb.europa.eu/service', FALSE)
getProviders()

## End(Not run)
```

---

getCodes	<i>get dsd codes for dataflow</i>
----------	-----------------------------------

---

**Description**

Extract the codes of a dimension. This function is used to retrieve the list of codes available for the input dimension and flow.

**Usage**

```
getCodes(provider, flow, dimension)
```

**Arguments**

flow	the identifier of the dataflow
dimension	the identifier of the dimension
provider	the name of the provider

**Details**

```
getCodes(provider, dataflow, dimension)
```

**Examples**

```
## Not run:  
codes=getCodes('ECB', 'EXR', 'FREQ')  
  
## End(Not run)
```

---

getDimensions	<i>get dsd dimensions for dataflow</i>
---------------	--

---

**Description**

Extract the dimensions of a DataFlow. This function is used to retrieve the list of dimensions of the input dataflow

**Usage**

```
getDimensions(provider, dataflow)
```

**Arguments**

dataflow	the identifier of the dataflow
provider	the name of the provider

**Details**

```
getDimensions(provider, dataflow)
```

**Examples**

```
## Not run:  
dims = getDimensions('ECB','EXR')  
  
## End(Not run)
```

---

getDSDIdentifier	<i>get DSD Identifier for dataflow</i>
------------------	--

---

### Description

Extract the dsd identifier of a DataFlow. This function is used to retrieve the name of the keyfamily of the input dataflow.

### Usage

```
getDSDIdentifier(provider, dataflow)
```

### Arguments

provider	the name of the provider
dataflow	the identifier of the dataflow

### Details

```
getDSDIdentifier(provider, dataflow)
```

### Examples

```
## Not run:  
id = getDSDIdentifier('ECB', 'EXR')  
  
## End(Not run)
```

---

getFlows	<i>get provider flow list</i>
----------	-------------------------------

---

### Description

Extract the list of DataFlows of a provider. This function is used to query the list of dataflows of the provider. A matching pattern can be provided, if needed.

### Usage

```
getFlows(provider, pattern = "")
```

### Arguments

pattern	the pattern to match against the dataflow id or description. If a pattern is not provided, all dataflows are returned.
provider	the name of the provider

**Details**

```
getFlows(provider, pattern)
```

**Author(s)**

Attilio Mattiocco <Attilio.Mattiocco@bancaditalia.it>, Diana Nicoletti

**Examples**

```
## Not run:  
## get all flows from ECB  
flows = getFlows('ECB')  
## get all flows that contain the 'EXR'  
flows = getFlows('ECB', '*EXR*')  
  
## End(Not run)
```

---

getProviders

*get available providers*

---

**Description**

Extract the list of available Data Providers. This function is used to query the list of data providers.

**Usage**

```
getProviders()
```

**Details**

```
getProviders()
```

**Examples**

```
## Not run:  
getProviders()  
  
## End(Not run)
```

---

getTimeSeries	<i>get time series</i>
---------------	------------------------

---

### Description

Extract a list of time series. This function is used to extract a list of time series identified by the parameters provided in input.

```
getTimeSeries(provider, id, start, end)
```

### Usage

```
getTimeSeries(provider, id, start='', end='')
```

### Arguments

provider	the name of the provider
id	identifier of the time series
end	the end time - optional
start	the start time - optional

### Examples

```
## Not run:  
# SDMX V2  
## get single time series:  
my_ts=getTimeSeries('ECB',id='EXR.A.USD.EUR.SP00.A')  
## get monthly and annual frequency:  
my_ts=getTimeSeries('ECB',id='EXR.A+M.USD.EUR.SP00.A')  
## get all available frequencies:  
my_ts=getTimeSeries('ECB',id='EXR..USD.EUR.SP00.A')  
  
## End(Not run)
```

---

getTimeSeries2	<i>get time series (SDMX v3)</i>
----------------	----------------------------------

---

### Description

Extract a list of time series . This function is used to extract a list of time series identified by the parameters provided in input.

**Usage**

```
getTimeSeries2(
  provider,
  dataflow,
  key = "",
  filter = "",
  start = "",
  end = "",
  attributes = "all",
  measures = "all"
)
```

**Arguments**

provider	the name of the provider
dataflow	dataflow of the time series
key	timeseries key - optional
filter	optional filter to be applied - optional
start	the start time - optional
end	the end time - optional
attributes	the comma separated list of attributes to be returned - optional, default='all', 'none' for no attributes
measures	the comma separated list of measures to be returned - optional, default='all', 'none' for no measures

**Examples**

```
## Not run:
# SDMX V3

## get single time series:
my_ts=getTimeSeries2('ECB', dataflow='EXR', key='A.USD.EUR.SP00.A')
## get all available frequencies:
my_ts=getTimeSeries2('ECB', dataflow='EXR', key='.USD.EUR.SP00.A')

#or

#' ## get single time series: EXR.A.USD.EUR.SP00.A
my_ts=getTimeSeries2('ECB', dataflow='EXR',
  filter='c[FREQ]=A&c[CURRENCY]=USD&c[CURRENCY_DENOM]=EUR&c[EXR_TYPE]=SP00&c[EXR_SUFFIX]=A')
## get monthly and annual frequency:
my_ts=getTimeSeries2('ECB', dataflow='EXR',
  filter='c[FREQ]=A,M&c[CURRENCY]=USD&c[CURRENCY_DENOM]=EUR&c[EXR_TYPE]=SP00&c[EXR_SUFFIX]=A')
## get all available frequencies:
my_ts=getTimeSeries2('ECB', dataflow='EXR',
  filter='c[CURRENCY]=USD&c[CURRENCY_DENOM]=EUR&c[EXR_TYPE]=SP00&c[EXR_SUFFIX]=A')

## End(Not run)
```



---

getTimeSeriesRevisions  
*get data revisions*

---

## Description

Extract time series starting from a specific update time and with history of revisions. This function works as getTimeSeriesTable but the query can be narrowed to getting only observations that were updated after a specific point in time, and eventually it returns the revision history of the matching time series.

The result is packed into a data.frame

## Usage

```
getTimeSeriesRevisions(provider, id, start = "", end = "",  
  updatedAfter = "", includeHistory = TRUE)
```

## Arguments

id	identifier of the time series
provider	the name of the provider
end	the end time - optional
start	the start time - optional
updatedAfter	the updatedAfter time - optional. It has to be in the form: 'YYYY-MM-DD'
includeHistory	boolean parameter - optional. If TRUE the full list of revisions will be returned

## Details

```
getTimeSeriesRevisions(provider, id, start, end, updatedAfter, includeHistory)
```

## Examples

```
## Not run:  
# get single time series with history:  
my_ts=getTimeSeriesRevisions('ECB','EXR.A.USD.EUR.SP00.A', includeHistory=TRUE)  
  
# get single time series (only observations updated after january 1st 2015):  
my_ts=getTimeSeriesRevisions('ECB','EXR.A.USD.EUR.SP00.A',  
  updatedAfter='2015', includeHistory=FALSE)  
  
# get single time series (full revision history starting from january 1st 2015):  
my_ts=getTimeSeriesRevisions('ECB','EXR.A.USD.EUR.SP00.A',  
  updatedAfter='2015', includeHistory=TRUE)  
  
## End(Not run)
```

---

getTimeSeriesTable     *get time series and return a data.frame*

---

### Description

Extract a list of time series identified by the parameters provided in input, and return a data.frame as result.

```
getTimeSeriesTable(provider, id, start, end, gregorianTime)
```

### Usage

```
getTimeSeriesTable(provider, id, start='', end='', gregorianTime=F)
```

### Arguments

id	identifier of the time series
provider	the name of the provider
end	the end time - optional
start	the start time - optional
gregorianTime	set to true to have all daily dates - optional

### Examples

```
## Not run:
# SDMX V2
## get single time series:
my_ts=getTimeSeriesTable('ECB',id='EXR.A.USD.EUR.SP00.A')
## get monthly and annual frequency:
my_ts=getTimeSeriesTable('ECB',id='EXR.A+M.USD.EUR.SP00.A')
## get all available frequencies:
my_ts=getTimeSeriesTable('ECB',id='EXR..USD.EUR.SP00.A')

## End(Not run)
```

---

getTimeSeriesTable2     *get time series and return a data.frame (SDMX v3)*

---

### Description

Extract a list of time series identified by the parameters provided in input, and return a data.frame as result.

**Usage**

```
getTimeSeriesTable2(
  provider,
  dataflow,
  key = "",
  filter = "",
  start = "",
  end = "",
  attributes = "all",
  measures = "all",
  updatedAfter = .jnull(),
  includeHistory = FALSE,
  gregorianTime = F
)
```

**Arguments**

provider	the name of the provider
dataflow	dataflow of the time series
key	timeseries key
filter	optional filter to be applied
start	the start time - optional
end	the end time - optional
attributes	the comma separated list of attributes to be returned - optional, default='all', 'none' for no attributes
measures	the comma separated list of measures to be returned - optional, default='all', 'none' for no measures
updatedAfter	return only changes after this date - optional
includeHistory	include history of revisions - optional, default=false
gregorianTime	set to true to have all daily dates - optional

**Examples**

```
## Not run:
# SDMX V3

## get single time series:
my_ts=getTimeSeriesTable('ECB', dataflow='EXR', key='A.USD.EUR.SP00.A')
## get monthly and annual frequency:
my_ts=getTimeSeriesTable('ECB', dataflow='EXR', key='A+M.USD.EUR.SP00.A')
## get all available frequencies:
my_ts=getTimeSeriesTable('ECB', dataflow='EXR', key='.USD.EUR.SP00.A')

#or

## get single time series: EXR.A.USD.EUR.SP00.A
```

```

my_ts=getTimeSeriesTable('ECB', dataflow='EXR',
  filter='c[FREQ]=A&c[CURRENCY]=USD&c[CURRENCY_DENOM]=EUR&c[EXR_TYPE]=SP00&c[EXR_SUFFIX]=A')
## get monthly and annual frequency:
my_ts=getTimeSeriesTable('ECB', dataflow='EXR',
  filter='c[FREQ]=A,M&c[CURRENCY]=USD&c[CURRENCY_DENOM]=EUR&c[EXR_TYPE]=SP00&c[EXR_SUFFIX]=A')
## get all available frequencies:
my_ts=getTimeSeriesTable('ECB', dataflow='EXR',
  filter='c[CURRENCY]=USD&c[CURRENCY_DENOM]=EUR&c[EXR_TYPE]=SP00&c[EXR_SUFFIX]=A')

## End(Not run)

```

---

sdmxdf

*convert time series to data.frame*


---

## Description

This function is used to transform the output of the `getSDMX` (or `getTimeSeries`) functions from a list of time series to a `data.frame`. The metadata can be requested by explicitly passing the appropriate parameters.

## Usage

```
sdmxdf(tslist, meta = FALSE, id = TRUE)
```

## Arguments

<code>tslist</code>	the list of time series to be converted
<code>meta</code>	set this to <code>TRUE</code> if you want metadata to be included (default: <code>FALSE</code> , as this may increase the size of the result quite a bit)
<code>id</code>	set this to <code>FALSE</code> if you do not want the time series id to be included (default: <code>TRUE</code> )

## Details

```
sdmxdf()
```

## Examples

```

## Not run:
a=getSDMX('ECB', 'EXR.A|Q|M|D.USD.EUR.SP00.A')
ddf = sdmxdf(a)
ddf = sdmxdf(a, meta=TRUE)

## End(Not run)

```

---

sdmxHelp	<i>open helper</i>
----------	--------------------

---

**Description**

Open a helper graphical application. This function opens a small sdmx metadata browser that can be helpful when building queries.

**Usage**

```
sdmxHelp(internalJVM = T)
```

**Arguments**

internalJVM	TRUE (default) if the GUI has to live in the R JVM. Set this to FALSE in MAC, to avoid issue #41
-------------	--

**Details**

```
sdmxHelp()
```

**Examples**

```
## Not run:  
#opens the helper in the R JVM  
sdmxHelp()  
#opens the helper in an external JVM  
sdmxHelp(FALSE)  
  
## End(Not run)
```

---

setProviderCredentials	<i>set Credentials for authenticating providers</i>
------------------------	---

---

**Description**

Sets the credentials for providers that need authentication

**Usage**

```
setProviderCredentials(provider, user=.jnull(), pw=.jnull())
```

**Arguments**

provider	the name of the provider
user	the username
pw	the password

# Index

- \* **package**

  - RJSDMX-package, [2](#)

- \* **rJava**

  - getFlows, [5](#)

- addProvider, [3](#)

- getCodes, [3](#)

- getDimensions, [4](#)

- getDSDIdentifier, [5](#)

- getFlows, [5](#)

- getProviders, [6](#)

- getTimeSeries, [7](#)

- getTimeSeries2, [7](#)

- getTimeSeriesRevisions, [9](#)

- getTimeSeriesTable, [10](#)

- getTimeSeriesTable2, [10](#)

- RJSDMX (RJSDMX-package), [2](#)

- RJSDMX-package, [2](#)

- sdmxdf, [12](#)

- sdmxHelp, [13](#)

- setProviderCredentials, [13](#)