Package 'atsd'

December 5, 2016

Title Support Querying Axibase Time-Series Database

Version 1.0.8441

Date 2015-03-16

Author Axibase Corporation

Maintainer Mikhail Zvagelsky <mikhail.zvagelsky@axibase.com>

URL https://github.com/axibase/atsd-api-r/

Description Provides functions for retrieving time-series and related meta-data such as entities, metrics, and tags from the Axibase Time-Series Database (ATSD). ATSD is a non-relational clustered database used for storing performance measurements from IT infrastructure resources: servers, network devices, storage systems, and applications.

Depends R (>= 3.1.2)

License Apache License 2.0

Imports RCurl (>= 1.95.4.5), httr (>= 0.6.1)

Suggests zoo, knitr, pander

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2016-12-05 18:28:47

R topics documented:

atsd *Support querying Axibase Time-Series Database.*

Description

The package lets you query Axibase Time-Series Database (ATSD) for time-series data and forecasts. List of package functions:

`set_connection`, `save_connection`, `show_connection` - are used to manage connection with ATSD: set up and store the url, user name, and password, configure cryptographic protocol and enforce SSL certificate validation in the case of https connection.

`query` – get time-series data and forecasts from ATSD.

`get_metrics` – get information about the metrics collected by ATSD.

`get_entities` – get information about the entities collected by ATSD.

`to_zoo` - converts a time-series data frame to 'zoo' object for manipulating irregular time-series with built-in functions in zoo package.

Type `browseVignettes(package = "atsd")` to view the complete package documentation and usage examples.

Author(s)

Axibase, api@axibase.com

---

get_entities *Get information about entities from Axibase Time-Series Database.*

Description

This function fetches a list of entities from ATSD, and convert it to a data frame.

Usage

```
get_entities(expression = "", active = "", tags = "*", limit = "",
  verbose = TRUE)
```

Arguments

`expression`	Optional string argument. Select entities matching particular name pattern and/or user-defined entity tags. The syntax of the `expression` argument is explained in the package vignette. Type `browseVignettes(package = "atsd")` to see the vignette.
`active`	Optional string argument: "true" or "false". Filter entities by `lastInsertTime`. If `active = "true"`, only entities with positive lastInsertTime are included in the response.

tags	Optional string argument. User-defined entitiy tags to be included in the response. By default, all the tags will be included.
limit	Optional integer argument. If limit > 0, the response shows the top-N entities ordered by name.
verbose	Optional boolean argument. If `verbose = FALSE` then all console output will be suppressed. By default, `verbose = TRUE`.

Value

A data frame. Each row of the data frame corresponds to an entity and its tags: `name`, `enabled`, `lastInsertTime` and user-defined entity tags as requested by the 'tags' argument. For more information look at the package vignette: `browseVignettes(package = "atsd")`.

See Also

Visit http://axibase.com/axibase-time-series-database/ for information about ATSD.

Examples

```
## Not run:
# get all entities and include all their tags in the data frame
get_entities()

# get all active entities and include all their tags in the data frame
get_entities(active = "true")

# Get the top 2 entities whose 'name'  and user-defined entity tag, 'app',
# match to the expression. Include the tag, 'app', into response
# and exclude oter user-defined entity tags.
get_entities(expression = "name like 'nur*' and lower(tags.app) like '*hbase*'",
             tags = "app", limit = 2)

## End(Not run)
```

---

get_metrics *Get information about metrics from Axibase Time-Series Database.*

Description

This function fetches a list of metrics and their tags from ATSD, and converts it to a data frame.

Usage

```
get_metrics(expression = "", active = "", tags = "*", limit = "",
  verbose = TRUE)
```

Arguments

expression	Optional string argument. Select metrics matching particular name pattern and/or user-defined metric tags. The syntax of the expression argument is explained in the package vignette. Type browseVignettes(package = "atsd") to see the vignette.
active	Optional string argument: "true" or "false". Filter metrics by lastInsertTime. If active = "true", only metrics with positive lastInsertTime are included in the response.
tags	Optional string argument. User-defined metric tags to be included in the response. By default, all the tags will be included.
limit	Optional integer argument. If limit > 0, the response shows the top-N metrics ordered by name.
verbose	Optional boolean argument. If verbose = FALSE then all console output will be suppressed. By default, verbose = TRUE.

Value

A data frame. Each row of the data frame corresponds to a metric and its tags: name, counter, lastInsertTime and user-defined metric tags as requested by the 'tags' argument. For more information view the package vignette: browseVignettes(package = "atsd").

See Also

Visit http://axibase.com/axibase-time-series-database/ for information about ATSD.

Examples

```
## Not run:
# get all metrics and include all their tags in the data frame
get_metrics()

# get the top 100 active metrics which have tag, 'table',
# include this tag into response and exclude oter user-defined metric tags
get_metrics(expression = "tags.table != ''", active = "true",
            tags = "table", limit = 100)

# get metrics which have user-defined metric tag, 'table',
# and whose name starts with 'cpu'
get_metrics(expression = "name like 'cpu*' and tags.table != ''")

# more complitcated expressions
get_metrics(expression = "likeAll(name, list('*disk*,*use*'))")
get_metrics(expression = "(name like 'cpu*' or tags.source = '') and tags.table like 'BC*'")

## End(Not run)
```

query *Fetch time-series historic data or forecasts from Axibase Time-Series Database.*

Description

This function fetches time-series from ATSD and creates a data frame from it.

Usage

```
query(metric, entity = NA, entity_group = NA, tags = character(),
  selection_interval, end_time = NA, aggregate_interval = NA,
  aggregate_statistics = "Avg", interpolation = "None",
  export_type = "History", verbose = TRUE)
```

Arguments

`metric`	Required string argument. The name of the metric you want to get data for. For example, `metric = "disk_used_percent"`. To obtain a list of metrics collected by ATSD use the `get_metrics` function.
`entity`	Optional string argument. The name of the entity you want to get data for. If not provided, then data for all entities will be fetched for the specified metric. Obtain the list of entities and their tags with the `get_entities` function.
`entity_group`	Optional string argument. You could specify a group of entities and extract data for entities from this group. For example, `entity_group = "HP Servers"`.
`tags`	Optional string vector argument. List of user-defined series tags to filter the fetched time-series data, for example, `c("disk_name=sda1", "mount_point=/")`.
`selection_interval`	Required string argument. This is the time interval for which the data will be selected. Specify it as "n-unit", where unit is a Second, Minute, Hour, Day, Week, Month, Quarter, or Year and n is the number of units, for example, "3-Week" or "12-Hour".
`end_time`	Optional string argument. The end time of the selection interval, for example, `end_time = "date(2014-12-27)"`. If not provided, the current time will be used. Specify the date and time, or use one of the supported expressions: http://axibase.com/axibase-time-series-database/screenshots-4/end-time/. For example, `'current_day'` would set the end of selection interval to 00:00:00 of the current day.
`aggregate_interval`	Optional string argument. The length of the aggregation interval. The period of produced time-series will be equal to the `aggregate_interval`. The value for each period is computed by the `aggregate_statistics` function applied to all samples of the original time-series within the period. The format of the `aggregate_interval` is the same as for the `selection_interval` argument, for example, "1-Minute".

`aggregate_statistics`
: Optional string vector argument. The statistic function used for aggregation. List of available functions: "Avg", "Min", "Max", "Sum", "Count", "StDev", "WAvg", "WTAvg", "Percentile 50", "Percentile 75", "Percentile 90", "Percentile 95", "Percentile 99", "Percentile 99.5", "Percentile 99.9". Multiple values are supported, for example, c("Min", "Avg", "StDev"). The default value is "Avg".

`interpolation`
: Optional string argument If aggregation is enabled, then the values for the periods without data will be computed by one of the following interpolation functions: "None", "Linear", "Step". The default value is "None".

`export_type`
: Optional string argument. It can take one of two values: "History" or "Forecast". The default value is "History". For example, `export_type = "Forecast"`.

`verbose`
: Optional boolean argument. If `verbose = FALSE` then all console output will be suppressed. By default, `verbose = TRUE`.

Details

The function has only two required arguments: `metric` and `selection_interval`.
Type `browseVignettes(package = "atsd")` to view the complete package documentation and usage examples.

Value

The function returns a data frame. It could be empty if no data match your query or if your request could not be processed by ATSD server. In any case you will get a console diagnostic message with a short description of the server responce.

See Also

Visit http://axibase.com/axibase-time-series-database/ for information about ATSD.

Examples

```
## Not run:
# Create data frame which contains time series for the given metric
# and all entities for the last 1 hour.
dfr <- query(metric = "disk_used_percent", selection_interval = "1-Hour")

dfr <- query( export_type = "Forecast",
              metric = "disk_used_percent",
              entity_group = "Linux",
              tags = c("mount_point=/boot", "file_system=/dev/sda1"),
              selection_interval = "1-Week",
              aggregate_statistics = c("Avg", "Min", "Max"),
              aggregate_interval = "1-Minute",
              interpolation = "Linear")

# Example of the end_time argument usage.
dfr <- query( metric = "cpu_usage",
              entity = "host-383",
              selection_interval = "1-Day",
```

```
                end_time = "date('2015-02-10 10:15:03')")

## End(Not run)
```

`save_connection` *Write connection parameters to configuration file.*

Description

The function writes the connection parameters into configuration file.

Usage

```
save_connection(url = NA, user = NA, password = NA, verify = NA,
  encryption = NA)
```

Arguments

`url`	Optional string argument. The url of ATSD with the port number.
`user`	Optional string argument. The user name.
`password`	Optional string argument. The user's password.
`verify`	Optional string argument – "yes" or "no". `verify = "yes"` ensures validation of ATSD SSL certificate and `verify = "no"` suppresses the validation (applicable in the case of 'https' protocol).
`encryption`	Optional string argument. Cryptographic protocol used by ATSD https server. Possible values are: "default", "ssl2", "ssl3", and "tls1" (In most cases, use "ssl3" or "tls1".)

Details

If you call `save_connection()` without arguments, then the current values of the connection parameters (including NAs) will be written to the configuration file. If you provide some arguments, they will be written into the configuration file. If configuration file is absent it will be created in the atsd package folder.

See Also

For more information about the configuration file view the package vignette: `browseVignettes(package = "atsd")`.

Examples

```
# Write the current values of the connection parameters to the configuration file
save_connection()

# Write the user name and the password to the configuration file
save_connection(user = "user001", password = "123456")
```

```
# Write all parameters nedeed for https connection to the configuration file
save_connection(url = "https://my.company.com:8443", user = "user001", password = "123456",
                verify = "no", encryption = "ssl3")
```

`set_connection` *Set up parameters of a connection with ATSD.*

Description

The function overrides the connection parameters for the duration of the current R session without changing the configuration file.

Usage

```
set_connection(url = NA, user = NA, password = NA, verify = NA,
  encryption = NA, file = NA)
```

Arguments

`url`	Optional string argument. The url of ATSD with the port number.
`user`	Optional string argument. The user name.
`password`	Optional string argument. The user's password.
`verify`	Optional string argument – "yes" or "no". `verify = "yes"` ensures validation of ATSD SSL certificate and `verify = "no"` suppresses the validation (applicable in the case of 'https' protocol).
`encryption`	Optional string argument. Cryptographic protocol used by ATSD https server. Possible values are: "default", "ssl2", "ssl3", and "tls1" (In most cases, use "ssl3" or "tls1".)
`file`	Optional string argument. The absolute path to the file from which the connection parameters could be read. The file should be formatted as the package configuration file, see the Details section below.

Details

The function overrides the connection parameters for the duration of the current R session without changing the configuration file. If called without arguments the function sets the connection parameters from the configuration file. If the file argument is provided the function use it. In both cases the current values of the parameters became the same as in the file. The file should be a plain text file formatted as the following:

```
# the url of ATSD including port number
url=http://host_name:port_number
# the user name
user=atsd_user_name
# the user's password
password=atsd_user_password
# validate ATSD SSL certificate: yes, no
```

```
verify=no
# cryptographic protocol used by ATSD https server:
# default, ssl2, ssl3, tls1
encryption=ssl3
```

In case the `file` argument is not provided, but some of other arguments are specified, the only specified parameters will be changed.

See Also

To see the current values of the connection parameters use the `show_connection` function. To change the configuration file use the `save_connection` function.

Examples

```
# Modify the user
set_connection(user = "user001")

# Modify the cryptographic protocol
set_connection(encryption = "tls1")

# Set up url, user and password
set_connection(url = "http://my.company.com:8088", user = "user001", password = "123456")

# Set up parameters of https connection
set_connection(url = "https://my.company.com:8443", user = "user001", password = "123456",
               verify = "no", encryption = "ssl3")

## Not run:
# Set up parameters from a file
set_connection(file = "/home/user001/atsd_https_connection.txt")

# Set up parameters from the configuration file
set_connection()

## End(Not run)
```

---

show_connection *Show connection parameters.*

Description

The function shows the current values of the connection parameters `url, user, password, verify` and `encryption`. They are used to arrange a connection with ATSD.

Usage

```
show_connection()
```

See Also

You could change the connection parameters with the `set_connection` function and save that changes to the configuration file with the `save_connection` function.

`to_zoo` *Build zoo object from data frame.*

Description

The function builds a zoo object based on the 'Timestamp' and 'Value' columns of the given data frame. Information from other columns will be lost. To use this function the 'zoo' package should be installed. To install the 'zoo' package type: `install.packages("zoo")`.

Usage

```
to_zoo(dfr)
```

Arguments

`dfr`	The data frame with 'Timestamp' and 'Value' columns.

Index