

# Package ‘DatastreamR’

May 13, 2024

**Title** Datastream API

**Version** 2.0.4

**Description** Access Datastream content through <https://product.datastream.com/dswsclient/Docs/Default.aspx>., our historical financial database with over 35 million individual instruments or indicators across all major asset classes, including over 19 million active economic indicators. It features 120 years of data, across 175 countries – the information you need to interpret market trends, economic cycles, and the impact of world events. Data spans bond indices, bonds, commodities, convertibles, credit default swaps, derivatives, economics, energy, equities, equity indices, ESG, estimates, exchange rates, fixed income, funds, fundamentals, interest rates, and investment trusts. Unique content includes I/B/E/S Estimates, Worldscope Fundamentals, point-in-time data, and Reuters Polls. Alongside the content, sit a set of powerful analytical tools for exploring relationships between different asset types, with a library of customizable analytical functions. In-house timeseries can also be uploaded using the package to combine with Datastream maintained datasets, use with these analytical tools and displayed in Datastream’s flexible charting facilities in Microsoft Office.

**License** GPL (>= 3)

**Encoding** UTF-8

**Imports** httr, jsonlite, logger, ini, stringr, methods

**RoxygenNote** 7.3.1

**Collate** 'Common.R' 'DSConnect.R' 'DSRequests.R' 'DSResponse.R'  
'DSUserDataObjectBase.R' 'Datastream.R' 'DatastreamR.R'  
'DatastreamUserCreated\_TimeSeries.R'

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** LSEG (Datastream) [aut, cre]

**Maintainer** LSEG (Datastream) <datastreamapi@lseg.com>

**Repository** CRAN

**Date/Publication** 2024-05-13 11:03:16 UTC

**R topics documented:**

DataClient . . . . .	2
DSCConnect . . . . .	7
DSTimeSeriesCarryIndicator . . . . .	9
DSTimeSeriesDataInput . . . . .	10
DSTimeSeriesDateAlignment . . . . .	11
DSTimeSeriesDateInfo . . . . .	12
DSTimeSeriesDateRange . . . . .	13
DSTimeSeriesDateRangeResponse . . . . .	14
DSTimeSeriesFrequencyConversion . . . . .	15
DSTimeSeriesRequestObject . . . . .	16
DSTimeSeriesResponseObject . . . . .	17
DSTimeSeriesUserObjectBase . . . . .	18
DSUserObjectAccessRights . . . . .	20
DSUserObjectBase . . . . .	20
DSUserObjectFrequency . . . . .	22
DSUserObjectGetAllResponse . . . . .	22
DSUserObjectResponse . . . . .	23
DSUserObjectResponseStatus . . . . .	24
DSUserObjectShareTypes . . . . .	25
DSUserObjectTypes . . . . .	26
TimeSeriesClient . . . . .	27
<b>Index</b>	<b>34</b>

---

DataClient

*DataClient*


---

**Description**

An RC class for accessing Datastream content through Datastream Web services API

**Value**

DataClient object

**Super class**

[DatastreamR::DSCConnect](#) -> DataClient

**Public fields**

useNaNforNotANumber : TRUE by default. NULLs are replaced to NaN (Not a Number)

**Methods****Public methods:**

- `DataClient$new()`
- `DataClient$getData()`
- `DataClient$getDataframe()`
- `DataClient$formatDataRequest()`
- `DataClient$getDataBundle()`
- `DataClient$toDataframe()`
- `DataClient$clone()`

**Method new():** Initialize method to create Datastream Object

*Usage:*

```
DataClient$new(
  config = NULL,
  username = "",
  password = "",
  proxys = NULL,
  sslCer = NULL
)
```

*Arguments:*

```
config : Config path
username : Your Id
password : Your Password
proxys : Proxy details (if any)
sslCer : ssl Certificates path
```

*Details:* Creates Datastream RC object for accessing Datastream content through Datastream Web Services API

*Returns:* DataClient object

*Examples:*

```
\dontrun{
# ds = DataClient$new(NULL, "YourID", "YourPswd")

# OR

# Login using config file
# Config file details provided in DSConnect.R
ds = DataClient("Config.ini")

# Timeseries requests

df = tryCatch (
  {ds$getData(tickers="VOD", fields=list("PH","PL"),
    start='2022-01-12', end='2022-07-13', kind=1)},
  error = function(e) { stop (message(e))})
```

```

    print(df)

# Snapshot requests

df = tryCatch (
{ ds$getData(tickers="VOD", fields=list("PH", "PL"),
             start='2022-01-12', kind=0)},
error = function(e) { stop (message(e))})
print(df)
}

```

**Method** `getData()`: `getData` posts the JSON formatted request and the JSON response is then converted to Dataframe, if `dataToDF` is TRUE.

*Usage:*

```

DataClient$getData(
  tickers,
  fields = NULL,
  start = "",
  end = "",
  freq = "D",
  kind = 1,
  properties = NULL
)

```

*Arguments:*

```

tickers : Intruments Eg: "VOD,BARC"
fields  : Datatypes Eg: list('PH', 'PL', 'PI')
start   : start date in "YYYY-mm-dd" format Eg: "2019-01-20"
end     : end date in
freq    : Refer DSDateFrequencyNames in DSRequests.R
kind    : 1 - TimeSeries Request (default), 0 - Snapshot Request
properties : properties

```

*Returns:* Response Class

**Method** `getDataframe()`: `getDataframe` posts the JSON formatted request and the JSON response is then converted to Dataframe

*Usage:*

```

DataClient$getDataframe(
  tickers,
  fields = NULL,
  start = "",
  end = "",
  freq = "D",
  kind = 1,
  properties = NULL
)

```

*Arguments:*

tickers : Instruments Eg: "VOD,BARC"  
 fields : Datatypes Eg: list('PH, 'PL', 'PI')  
 start : start date in "YYYY-mm-dd" format Eg: "2019-01-20"  
 end : end date in  
 freq : Refer DSDateFrequencyNames in DSRequests.R  
 kind : 1 - TimeSeries Request (default), 0 - Snapshot Request  
 properties : properties

*Returns:* Dataframe

**Method** `formatDataRequest()`: This method formats the request provided by client (in form of tickers and fields) to the JSON formatted request.

*Usage:*

```

DataClient$formatDataRequest(
  tickers,
  fields = NULL,
  start = "",
  end = "",
  freq = "D",
  kind = 1
)

```

*Arguments:*

tickers : Instruments  
 fields : Datatypes  
 start : start date  
 end : end date  
 freq : frequency  
 kind : kind = 0 for Snapshot request and Kind = 1 for Timeseries request

*Returns:* JSON formatted data request

**Method** `getDataBundle()`: `getData` posts the JSON formatted request and the JSON response is then converted to Dataframe, if `dataToDF` is TRUE.

*Usage:*

```
DataClient$getDataBundle(dataRequests, properties = NULL)
```

*Arguments:*

dataRequests : dataRequests should be formed using `formatDataRequest` method  
 properties : properties

*Returns:* Response Class

**Method** `toDataframe()`: Converts the Class response into a Dataframe which can be further used to plot

*Usage:*

```
DataClient$toDataframe(dataResponse)
```

*Arguments:*

dataResponse : The raw data response that is ingested into Dataframe

*Returns:* Dataframe

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
DataClient$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

**Examples**

```
## -----
## Method `DataClient$new`
## -----

## Not run:
# ds = DataClient$new(NULL, "YourID", "YourPswd")

# OR

# Login using config file
# Config file details provided in DSConnect.R
ds = DataClient("Config.ini")

# Timeseries requests

df = tryCatch (
  {ds$getData(tickers="VOD", fields=list("PH","PL"),
             start='2022-01-12', end='2022-07-13', kind=1)},
  error = function(e) { stop (message(e))})
print(df)

# Snapshot requests

df = tryCatch (
  { ds$getData(tickers="VOD", fields=list("PH", "PL"),
             start='2022-01-12', kind=0)},
  error = function(e) { stop (message(e))})
print(df)

## End(Not run)
```

---

DSConnect	<i>initializer</i>
-----------	--------------------

---

## Description

initializer

initializer

## Public fields

Config : config file path

Username : Your id

Password : Your Password

Proxies : Proxy Server details (if any)

SslCert : ssl Certificates file path

Url : internal parameter set by application

RequestUrl : internal parameter set up by application

Token : Token received on successful logon

TokenExpiry : Expiry time of the token

Timeout : Max Timeout set in Configuration for the process to complete

Version : Application Version

AppID : Application Id

## Methods

### Public methods:

- [DSConnect\\$new\(\)](#)
- [DSConnect\\$getJsonResponse\(\)](#)
- [DSConnect\\$getToken\(\)](#)
- [DSConnect\\$isValid\(\)](#)
- [DSConnect\\$checkToken\(\)](#)
- [DSConnect\\$clone\(\)](#)

**Method new():** DSConnect connects to the Datastream web service on behalf of derived classes and helps to send and retrieve data

DSConnect connects to the Datastream web service on behalf of derived classes

*Usage:*

```
DSConnect$new(  
    config = NULL,  
    username = NA,  
    password = NA,  
    proxies = NULL,
```

```

    sslCert = NULL,
    service = NULL
)

```

*Arguments:*

config : config path  
 username : your username  
 password : your password  
 proxies : proxy url, if any  
 sslCert : path to ssl cert file  
 service : internally set by the application

*Details:* user details can be supplied from a config file or passed directly as parameters in the constructor of the derived user object type class.

- 1) Using ini file (e.g. config.ini) with format

```

[credentials]
username=YourID
password=YourPwd

```

```

[proxies]
url=Your Proxy Url
port=Proxy port
username=Proxy Username
password=Proxy Password

```

```

[cert]
sslCertFile=YourCertFile
#only read if no sslCertFile entry
sslCertPath=YourCertPath

```

**Method** `getJsonResponse()`: This method makes the query against the API service and does some basic error handling

*Usage:*

```
DSConnect$getJsonResponse(requestUrl, request)
```

*Arguments:*

requestUrl : Url used to send the Post request  
 request : Raw Datastream request

**Method** `getToken()`: `getToken` uses you credentials to try and obtain a token to be used in subsequent request for data. The returned token is valid for 24 hours

*Usage:*

```
DSConnect$getToken()
```

**Method** `IsValid()`: `IsValid` checks whether the token is valid against the token expiry time

*Usage:*

```
DSConnect$IsValid()
```



**Method** CheckToken(): CheckToken checks whether the token is valid against the token expiry time and generates new token if the token has expired.

*Usage:*

DSCconnect\$CheckToken()

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

DSCconnect\$clone(deep = FALSE)

*Arguments:*

deep Whether to make a deep clone.

DSTimeSeriesCarryIndicator

*DSTimeSeriesCarryIndicator*

## Description

This list is a supporting attribute for the CarryIndicator properties of the DSTimeSeriesRequestObject and DSTimeSeriesResponseObjects. When you supply data which contains 'Not A Number' values (NaN) to denote non trading days, this list instructs the mainframe in how to store the values.

## Usage

DSTimeSeriesCarryIndicator

## Format

An object of class list of length 3.

## Value

numeric

## Fields

**Yes** Any incoming NaN values are replaced with the last non-NaN value (e.g. 1,2,3,NaN,5,NaN,7,8 will be converted and stored as 1,2,3,3,5,5,7,8).

**No** Any incoming NaN values are stored as is and returned as NaN values.

**Pad** This is similar to YES, but also pads the final value for any dates your users may request beyond the last date in your timeseries.

For example, if your timeseries supplies just 3 values 1, NaN and 3, and your user requests a range of dates two days before and two days after your range, your user will receive the following values:

No: NaN, NaN, 1, NaN, 3, NaN, NaN

Yes: NaN, NaN, 1, 1, 3, NaN, NaN

Pad: NaN, NaN, 1, 1, 3, 3, 3

---

DSTimeSeriesDataInput *DSTimeSeriesDataInput*


---

### Description

This class is a supporting attribute for the DateInput property of the DSTimeSeriesRequestObject. It is used to supply the raw data for the timeseries.

### Value

DSTimeSeriesDataInput object

### Public fields

StartDate A datetime value defining the start date for the timeseries.

EndDate A datetime value defining the end date for the timeseries.

Frequency The frequency of the timeseries. One of the DSUserObjectFrequency values defined in DSUserDataObjectBase.R

Values An array of float values. Use NULL to represent NotANumber for non-trading days. Alternatively, if you set the DatastreamUserCreated\_TimeSeries property useNaNforNotANumber as True, you can use float NaN values.

### Methods

#### Public methods:

- [DSTimeSeriesDataInput\\$new\(\)](#)
- [DSTimeSeriesDataInput\\$clone\(\)](#)

#### Method new():

*Usage:*

```
DSTimeSeriesDataInput$new(
  startDate = NULL,
  endDate = NULL,
  frequency = DSUserObjectFrequency$Daily,
  values = NULL
)
```

*Arguments:*

startDate A datetime value defining the start date for the timeseries.

endDate A datetime value defining the end date for the timeseries.

frequency The frequency of the timeseries.

values An array of float values

*Returns:* DSTimeSeriesDataInput object

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

```
DSTimeSeriesDataInput$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

**Note**

Datastream takes the StartDate, Frequency and Values properties defined here and creates the time-series based only on these parameters. The EndDate is not actually used internally other than for logging purposes. The true end date is calculated based on the start date, frequency and the supplied list of values. Supply too few or too many values and the mainframe will accept them and set the end date accordingly based on the given frequency for the item.

---

DSTimeSeriesDateAlignment

*DSTimeSeriesDateAlignment*

---

**Description**

This list is a supporting attribute for the DateAlignment properties of the DSTimeSeriesRequestObject and DSTimeSeriesResponseObjects. When you supply monthly, quarterly or annual data, the dates are stored internally as the first day of the given period and always returned to you through this interface as the first date of the given period. However, when your users request data from Datastream, you can specify whether the dates returned to users are returned with dates set as the start, mid or end of the requested period

**Usage**

```
DSTimeSeriesDateAlignment
```

**Format**

An object of class list of length 3.

**Value**

numeric

**Fields**

EndPeriod This will return dates to your users that represent the last day of the month, quarter or year.

StartPeriod This will return dates to your users that represent the first day of the month, quarter or year.

MidPeriod This will return dates to your users that represent the middle of the month (15th day), quarter (15th of the middle month) or year (30th June).

---

DSTimeSeriesDateInfo *DSTimeSeriesDateInfo*


---

### Description

This class is a supporting attribute for the DateInfo property of the DSTimeSeriesResponseObject. It describes the basic range of data for the timeseries.

The DateRange property (DSTimeSeriesDateRange described above) of the DSTimeSeriesResponseObject always returns the dates for a given frequency as the first date in each period (e.g. 2022-01-01, 2020-04-01, etc. for quarterly frequencies). However, The StartDate and EndDate values returned in this class for the DSTimeSeriesResponseObject reflect the start and end dates of the range of dates that would be returned to users requesting the data via Datastream For Office, charting, etc. This depends on the DateAlignment property (DSTimeSeriesDateAlignment) of the timeseries. The start and end dates returned here will be either the start, mid or end dates for the set frequency based on the DateAlignment property (see DSTimeSeriesDateAlignment).

### Value

DSTimeSeriesDateInfo object

### Public fields

StartDate A datetime value defining the start date of the timeseries data

EndDate A datetime value defining the end date for the timeseries.

Frequency The frequency of the timeseries. One of the DSUserObjectFrequency values defined in DSUserDataObjectBase.R

### Methods

#### Public methods:

- [DSTimeSeriesDateInfo\\$new\(\)](#)
- [DSTimeSeriesDateInfo\\$clone\(\)](#)

#### Method new():

*Usage:*

DSTimeSeriesDateInfo\$new(jsonDict)

*Arguments:*

jsonDict JSON dictionary (from JSON Response)

*Returns:* DSTimeSeriesDateInfo object

#### Method clone(): The objects of this class are cloneable with this method.

*Usage:*

DSTimeSeriesDateInfo\$clone(deep = FALSE)

*Arguments:*

deep Whether to make a deep clone.

---

DSTimeSeriesDateRange *DSTimeSeriesDateRange*

---

## Description

This class is a supporting attribute for the DateRange property of the DSTimeSeriesResponseObject. It returns the raw data for the timeseries. The DateRange property of the DSTimeSeriesResponseObject always returns the dates for a given frequency as the first date in each period. (e.g. 2022-01-01, 2020-04-01, etc. for quarterly frequencies). You specify whether you want your users to receive either the first, mid or end dates in the given period by setting the DateAlignment property (DSTimeSeriesDateAlignment) of the DSTimeSeriesRequestObject.

When you retrieve a list of all your available timeseries using the GetAllItems method, since this list could contain many thousand timeseries objects, the Dates and Values lists will always be NULL. Only the ValuesCount field will be set to reflect the number of datapoints available for each item. You need to request an individual timeseries (GetItem method) in order to receive a response containing actual data in the Dates and Values properties.

## Value

DSTimeSeriesDateRange object

## Public fields

- Dates • A list of datetime values specifying the dates for each datapoint.
- Values • A list of float values specifying the values for each datapoint.
- ValuesCount • A count of the number of datapoints in the timeseries.

## Methods

### Public methods:

- [DSTimeSeriesDateRange\\$new\(\)](#)
- [DSTimeSeriesDateRange\\$clone\(\)](#)

### Method new():

*Usage:*

```
DSTimeSeriesDateRange$new(jsonDict, convertNullToNans = FALSE)
```

*Arguments:*

jsonDict JSON dictionary (from JSON Response)

convertNullToNans FALSE by default, TRUE converts the NULLs in the NaNs (Not a Number)

*Returns:* DSTimeSeriesDateRange object

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

```
DSTimeSeriesDateRangeResponse$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

DSTimeSeriesDateRangeResponse

*DSTimeSeriesDateRangeResponse*

**Description**

DSTimeSeriesDateRangeResponse is the object returned from the timeseries GetTimeseriesDateRange method. This method allows you to determine the supported dates between given start and end dates at a specified frequency.

**Value**

DSTimeSeriesDateRangeResponse object

**Public fields**

**Dates** A list of datetime values representing the supported dates between requested start and end dates at a specified frequency.

**ResponseStatus** This property will contain a DSUserObjectResponseStatus value. DSUserObjectResponseStatus\$UserObjectSuccess represents a successful response.

**ErrorMessage** If ResponseStatus is not DSUserObjectResponseStatus\$UserObjectSuccess this status string will provide a description of the error condition.

**Properties** Not currently used and will currently always return NULL.

**Methods****Public methods:**

- [DSTimeSeriesDateRangeResponse\\$new\(\)](#)
- [DSTimeSeriesDateRangeResponse\\$clone\(\)](#)

**Method new():***Usage:*

```
DSTimeSeriesDateRangeResponse$new(jsonDict = NULL)
```

*Arguments:*

jsonDict : JSON dictionary (from JSON Response)

*Returns:* DSTimeSeriesDateRangeResponse object

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

```
DSTimeSeriesDateRangeResponse$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

---

DSTimeSeriesFrequencyConversion

*DSTimeSeriesFrequencyConversion*

---

### Description

This list is a supporting attribute for the FrequencyConversion properties of the DSTimeSeriesRequestObject and DSTimeSeriesResponseObjects. This enumeration specifies how to return values if your end users requests the timeseries at a lower frequency. For example: if your timeseries has daily frequency data, and your user requests monthly or quarterly data, the FrequencyConversion property instructs the mainframe how to return monthly or quarterly data.

### Usage

```
DSTimeSeriesFrequencyConversion
```

### Format

An object of class list of length 4.

### Value

numeric

### Fields

EndValue The daily value for the end of the requested period will be returned.

AverageValue The average of all the values for the requested period will be returned.

SumValues The sum of all the values for the requested period will be returned.

ActualValue The actual value for the requested start date will be returned for the same given date in the requested period.

---

```
DSTimeSeriesRequestObject
    DSTimeSeriesRequestObject
```

---

### Description

This is a subclass of DSTimeSeriesUserObjectBase and is used to create or modify a timeseries. (See DSTimeSeriesUserObjectBase for details of all the superclass properties.)

### Value

DSTimeSeriesRequestObject object

### Super classes

[DatastreamR::DSUserObjectBase](#) -> [DatastreamR::DSTimeSeriesUserObjectBase](#) -> DSTimeSeriesRequestObject

### Public fields

Id A valid TimeSeries Id

DataInput A DSTimeSeriesDataInput object used to supply the start date, end date, frequency and list of data values. (See DSTimeSeriesDataInput for details.)

### Methods

#### Public methods:

- [DSTimeSeriesRequestObject\\$new\(\)](#)
- [DSTimeSeriesRequestObject\\$clone\(\)](#)

#### Method new():

*Usage:*

```
DSTimeSeriesRequestObject$new(
  id = "",
  startDate = NULL,
  endDate = NULL,
  frequency = NULL,
  values = NULL
)
```

*Arguments:*

id A valid TimeSeries Id

startDate A datetime value defining the start date for the timeseries

endDate A datetime value defining the end date for the timeseries

frequency The frequency of the timeseries. DSUserObjectFrequency is defined in DSUser-DataObjectBase.R

values list of float values



*Returns:* DSTimeSeriesRequestObject object

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
DSTimeSeriesRequestObject$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

DSTimeSeriesResponseObject

*DSTimeSeriesResponseObject*

## Description

This is a subclass of DSTimeSeriesUserObjectBase and is used to return the details for a timeseries. (See DSTimeSeriesUserObjectBase for details of all the superclass properties.)

## Value

DSTimeSeriesResponseObject object

## Super classes

[DatastreamR::DSUserObjectBase](#) -> [DatastreamR::DSTimeSeriesUserObjectBase](#) -> DSTimeSeriesResponseObject

## Public fields

**DateInfo** A DSTimeSeriesDateInfo object defining the start date, end date and frequency of the timeseries.

**DateRange** A DSTimeSeriesDateRange object used to return the dates and values stored in the timeseries. See DSTimeSeriesDateRange for details.

## Methods

### Public methods:

- [DSTimeSeriesResponseObject\\$new\(\)](#)
- [DSTimeSeriesResponseObject\\$clone\(\)](#)

### Method new():

*Usage:*

```
DSTimeSeriesResponseObject$new(jsonDict = NULL, convertNullToNans = FALSE)
```

*Arguments:*

jsonDict : JSON dictionary (from JSON Response)

convertNullToNans : FALSE by default, TRUE converts the NULLs in the NaNs (Not a Number)

*Returns:* DSTimeSeriesResponseObject object

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
DSTimeSeriesResponseObject$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

DSTimeSeriesUserObjectBase

*DSTimeSeriesUserObjectBase*

## Description

This is the base object for creating or requesting timeseries data. It has two subclasses DSTimeSeriesRequestObject and DSTimeSeriesResponseObject. It defines the basic attributes for a timeseries. It subclasses DSUserObjectBase which defines the basic attributes common to all five user created item types supported by the API.

Specifics of some of the properties of the DSUserObjectBase superclass

### ID

The ID property is defined in DSUserObjectBase but has a specific format for timeseries. Timeseries IDs must be 8 alphanumeric characters long, start with TS followed by 6 uppercase alphanumeric characters. For example: TSTEST01, TS123456, TSMYTEST, etc.

### Mnemonic

The Mnemonic property is defined in DSUserObjectBase but should always be left empty or set the same as the ID property for timeseries requests. As a safety measure, this class always ensures it's the same as the ID. In a response from the API server, the value will always be the same as the ID. (see DSUserObjectBase for a description of the other properties)

DSTimeSeriesUserObjectBase specific properties

## Value

DSTimeSeriesUserObjectBase object

## Super class

[DatastreamR::DSUserObjectBase](#) -> DSTimeSeriesUserObjectBase

## Public fields

**ManagementGroup** This is an optional group name that allows you to organise timeseries into distinct 'folders' displayed in the search category of Navigator. This can be up to 10 uppercase alphanumeric characters. Leave blank for the item to be assigned under the 'GENERAL' group.

- Units** This is an optional qualifying unit for your data. For example: tons, US\$ millions, index, etc. Maximum 12 characters.
- DecimalPlaces** A numeric value between 0 and 8 decimal places specifying how many decimal places to use when storing data. The maximum length including decimals for a value is 10 characters including the decimal point. Boundary case examples are 0.12345678, 1234567890, 123456789.0, etc.
- FrequencyConversion** A DSTimeSeriesFrequencyConversion enum value specifying how to return values if a user requests data at a lower frequency than the timeseries data is supplied. See DSTimeSeriesFrequencyConversion for details.
- DateAlignment** A DSTimeSeriesDateAlignment enum value specifying whether dates for certain frequencies should be returned as the start, middle or end date of the period. See DSTimeSeriesDateAlignment for details.
- CarryIndicator** A DSTimeSeriesCarryIndicator enum value specifying how to treat 'Not A Number' values for non-trading days and how to represent values if users request data after the end of the timeseries range. See DSTimeSeriesCarryIndicator for details.
- PrimeCurrencyCode** An optional 2 character currency code for your timeseries.
- HasPadding** This property has been replaced with the CarryIndicator property and will always be False
- UnderCurrencyCode** This property has been deprecated and will always return NULL
- AsPercentage** This property has been deprecated and will always return False

## Methods

### Public methods:

- [DSTimeSeriesUserObjectBase\\$new\(\)](#)
- [DSTimeSeriesUserObjectBase\\$clone\(\)](#)

### Method new():

*Usage:*

```
DSTimeSeriesUserObjectBase$new(jsonDict)
```

*Arguments:*

jsonDict JSON dictionary (from JSON Response)

*Returns:* DSTimeSeriesUserObjectBase object

### Method clone(): The objects of this class are cloneable with this method.

*Usage:*

```
DSTimeSeriesUserObjectBase$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

---

 DSUserObjectAccessRights

*Datastream User Object access rights*


---

### Description

All user created objects have a flag specifying if they can be modified by the user. All items that have their ShareType property set to DSUserObjectShareTypes. PrivateUserGroup will also have their AccessRight property set to ReadWrite. Global expression objects, not being editable by users, will have the AccessRight property set to Read.

### Usage

DSUserObjectAccessRights

### Format

An object of class list of length 2.

### Value

numeric

---

 DSUserObjectBase

*DSUserObjectBase*


---

### Description

DSUserObjectBase is the base object for all five user object types. It defines the properties common to all the types.

### Value

DSUserObjectBase object

### Public fields

**Id** The object identifier. The format is specific to each object type. See the individual object file for the particular specification.

**Mnemonic** For all object types bar indices, this is the same as the Id property. For indices, the ID (of the form X#:Xnnnnn where n is a digit) is returned when you create an index and is used to manage the index via the API interface. The Mnemonic property is specified when creating an index and is used to reference the index when using Datastream tools such as Charting, Datastream For Office, etc. A mnemonic has format X#aaaaa where aaaaa is 1 to 6 alphanumeric characters.

**DisplayName** A string describing the object. The maximum length varies from object type to object type.

Expression: Max 30 alphanumeric characters.

Index: Max 60 alphanumeric characters.

List: Max 60 alphanumeric characters.

Regression: Max 50 alphanumeric characters.

Timeseries: Max 64 alphanumeric characters.

**Description** Currently this isn't supported. When the API returns an object, the Description property will be the same as the DisplayName property.

**Created** a datetime value representing the date when the object was first created.

**LastModified** a datetime value representing the date when the object was last updated.

**Owner** The parent Datastream ID that owns the object. This will be the parent of your Datastream ID. For global expressions this will always be 'Admin'

**ShareType** For all objects except global expressions, this will be DSUserObjectShareTypes.PrivateUserGroup. For global expressions it will be DSUserObjectShareTypes.Global.

**AccessRight** For all objects except global expressions, this will be DSUserObjectAccessRights.ReadWrite. For global expressions it will be DSUserObjectAccessRights.Read.

## Methods

### Public methods:

- [DSUserObjectBase\\$new\(\)](#)
- [DSUserObjectBase\\$SetSafeUpdateParams\(\)](#)
- [DSUserObjectBase\\$clone\(\)](#)

**Method** `new()`: Initialize

*Usage:*

```
DSUserObjectBase$new(jsonResp = NULL)
```

*Arguments:*

jsonResp : json Response

*Returns:* DSUserObjectBase object

**Method** `SetSafeUpdateParams()`: The following parameters are set only in response when we query for user created items. This method is called before Create or Update to ensure safe values set prior to JSON encoding

*Usage:*

```
DSUserObjectBase$SetSafeUpdateParams()
```

*Returns:* No return value

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
DSUserObjectBase$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

---

DSUserObjectFrequency *DSUserObjectFrequency*

---

### Description

Regressions and Timeseries objects specify a frequency for the underlying data. DSUserObjectFrequency defines the supported frequencies.

### Usage

DSUserObjectFrequency

### Format

An object of class list of length 5.

### Value

numeric

---

DSUserObjectGetAllResponse  
*DSUserObjectGetAllResponse*

---

### Description

This is the object returned for the client class' GetAllItems query only.

For GetAllItems queries only, the returned objects will not have all their property fields set. Specifically for below:

Expression: All property fields are fully populated.

Index: The ConstituentsCount property will correctly specify the number of constituents but the Constituents property will be NULL.

List: The ConstituentsCount property will correctly specify the number of constituents but the Constituents property will be NULL.

Regression: All property fields are fully populated.

Timeseries: The ValuesCount field of the DateRange property will specify the number of date value pairs, but the Dates and Values fields will be NULL.

You need to query for the individual object using the GetItem request to retrieve the full content for the object.

### Value

DSUserObjectGetAllResponse object

**Public fields**

UserObjectType specifies the returned object types. e.g. DSUserObjectTypes.List, DSUserObjectTypes.TimeSeries, etc.

UserObjects An array of the specified object types such as DSListUserObject, DSRegressionUserObject, etc.

UserObjectsCount The number of objects returned in the UserObjects property.

ResponseStatus This property will contain a DSUserObjectResponseStatus value. DSUserObjectResponseStatus.UserObjectSuccess represents a successful response.

ErrorMessage If ResponseStatus is not DSUserObjectResponseStatus.UserObjectSuccess this status string will provide a description of the error condition.

Properties Not currently used and will currently always return NULL.

**Methods****Public methods:**

- [DSUserObjectGetAllResponse\\$new\(\)](#)
- [DSUserObjectGetAllResponse\\$clone\(\)](#)

**Method** new(): Initialize

*Usage:*

```
DSUserObjectGetAllResponse$new(jsonResp = NULL)
```

*Arguments:*

jsonResp JSON Response

*Returns:* DSUserObjectGetAllResponse object

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
DSUserObjectGetAllResponse$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

---

DSUserObjectResponse *DSUserObjectResponse*

---

**Description**

This is the object returned from the client class' GetItem, CreateItem, UpdateItem and DeleteItem requests.

**Value**

DSUserObjectResponse object

**Public fields**

**UserObjectId** The ID of the object requested. If the item is deleted, the **UserObject** property will be NULL but the **UserObjectId** field will be populated

**UserObjectType** specifies the returned object type. e.g. **DSUserObjectTypes.List**, **DSUserObjectTypes.TimeSeries**, etc.

**UserObject** For all queries bar **DeletItem**, if the query is successful, this property will contain the user created item requested.

**ResponseStatus** This property will contain a **DSUserObjectResponseStatus** value. **DSUserObjectResponseStatus.UserObjectSuccess** represents a successful response.

**ErrorMessage** If **ResponseStatus** is not **DSUserObjectResponseStatus.UserObjectSuccess** this status string will provide a description of the error condition.

**Properties** Not currently used and will currently always return NULL.

**Methods****Public methods:**

- [DSUserObjectResponse\\$new\(\)](#)
- [DSUserObjectResponse\\$clone\(\)](#)

**Method new():** Initialize

*Usage:*

`DSUserObjectResponse$new(jsonResp = NULL)`

*Arguments:*

`jsonResp` JSON Response

*Returns:* **DSUserObjectResponse** object

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

`DSUserObjectResponse$clone(deep = FALSE)`

*Arguments:*

`deep` Whether to make a deep clone.

---

DSUserObjectResponseStatus

*DSUserObjectResponseStatus*

---



**Description**

All client methods to retrieve or modify user created items return a response object which includes a ResponseStatus property. The ResponseStatus property specifies success or failure for the request using a DSUserObjectResponseStatus value

Response Values:

UserObjectSuccess : The request succeeded and the response object's UserObject(s) property should contain the (updated) object (except for DeleteItem method).

UserObjectPermissions : Users need to be specifically permissioned to create custom objects. This flag is set if you are not currently permissioned.

UserObjectNotPresent : Returned if the requested ID does not exist.

UserObjectFormatError : Returned if your request object is not in the correct format.

UserObjectTypeError : Returned if your supplied object is not the same as the type specified.

UserObjectError : The generic error flag. This will be set for any error not specified above.

Examples are:

1. Requested object ID is not present
2. You have exceeded the number of custom objects permitted on your account.

**Usage**

DSUserObjectResponseStatus

**Format**

An object of class list of length 6.

**Value**

numeric

---

DSUserObjectShareTypes

*DSUserObjectShareTypes*

---

**Description**

All user created objects have a flag specifying how they are shared with other users. Currently only PrivateUserGroup and Global are supported.

PrivateUserGroup are items created by any Datastream ID that shares a parent Datastream ID with your ID. Only children of the parent ID can access the user o Expressions can be either PrivateUserGroup or Global. Like the other object types, PrivateUserGroup items are items created by users and visible to just children of their Datastream parent ID. PrivateUserGroup expressions have the ID signature Eaaa, where 'a' is any alphabetical character.

Global expressions are Datastream owned expressions that are available for use by any client. These have the signature nnnE, where n is a digit. Global expressions can be retrieved using the API service, but you cannot modify them. Only your PrivateUserGroup items can be modified.

**Usage**

DSUserObjectShareTypes

**Format**

An object of class list of length 5.

**Value**

numeric

---

DSUserObjectTypes      *DSUserObjectTypes*

---

**Description**

Five user created types are supported. When the client classes communicate with the API server, a DSUserObjectTypes property is set to specify the object type. Responses from the API server also specify the type of the object being returned.

**Usage**

DSUserObjectTypes

**Format**

An object of class list of length 6.

**Value**

numeric

---

TimeSeriesClient	<i>TimeSeriesClient</i>
------------------	-------------------------

---

## Description

This is the client class that manages the connection to the API server on your behalf. It allows you to query for all your timeseries and to create/modify new timeseries.

## Details

### Methods Supported

`GetAllItems` : Allows you to query for all the current timeseries available for your use.

`GetItem` : Allows you to download the details of a specific timeseries item.

`GetTimeseriesDateRange` : Allows you to determine the supported timeseries dates between supplied start and end dates at a specified frequency.

`CreateItem` : Allows you to create a new timeseries item with up to 130 years of daily data.

`UpdateItem` : Allows you to update an existing timeseries.

`DeleteItem` : Allows you to delete an existing timeseries.

## Value

TimeSeriesClient object

## Super class

`DatastreamR::DSConnect` -> TimeSeriesClient

## Public fields

`useNaNforNotANumber` If Enabled, NaN is appears in output response instead of NULL

`TimeseriesResponseType` Response type

## Methods

### Public methods:

- `TimeSeriesClient$new()`
- `TimeSeriesClient$.checkValidTimeseriesId()`
- `TimeSeriesClient$.checkTimeSeriesReqValidity()`
- `TimeSeriesClient$.checkKeyTimeseriesProperties()`
- `TimeSeriesClient$.asGetAllResponse()`
- `TimeSeriesClient$.asGetResponse()`
- `TimeSeriesClient$.jsonRequestEncoder()`
- `TimeSeriesClient$.jsonResponseDecoder()`
- `TimeSeriesClient$GetAllItems()`
- `TimeSeriesClient$GetItem()`

- [TimeSeriesClient\\$CreateItem\(\)](#)
- [TimeSeriesClient\\$updateItem\(\)](#)
- [TimeSeriesClient\\$DeleteItem\(\)](#)
- [TimeSeriesClient\\$getTimeseriesDateRange\(\)](#)
- [TimeSeriesClient\\$clone\(\)](#)

**Method new():** User details can be supplied from a config file or passed directly as parameters in the constructor of the derived user object type class. (See the DSConnect superclass for a description of the connection parameters required)

*Usage:*

```
TimeSeriesClient$new(  
  config = NULL,  
  username = NULL,  
  password = NULL,  
  proxies = NULL,  
  sslCer = NULL  
)
```

*Arguments:*

config Configuration File path  
username Your Datastream Id  
password Your Password  
proxies Proxies if any  
sslCer Path to CA bundle certificates file

*Details:* Timeseries Properties:

useNaNforNotANumber : Non-trading days are stored as double NaNs on Datastream, JSON protocol permits NaNs as valid numbers. Thus, all the NULLs in the converted to NaNs in the JSON requests. Responses contain the NULLs, But this should be converted to Nans for Plotting purposes. If you want to receive NaN float values, set useNaNforNotANumber to TRUE, any NULLs in the returned array of float values will be converted to NaNs.

*Returns:* TimeSeriesClient object

**Method .checkValidTimeseriesId():** A helper method to check the timeseries Id

*Usage:*

```
TimeSeriesClient$.checkValidTimeseriesId(inputId)
```

*Arguments:*

inputId : Timeseries Id

*Returns:* NULL if Timeseries id is valid else error string

**Method .checkTimeSeriesReqValidity():** A helper method to check some of the mandatory fields of timeseries for its validity

*Usage:*

```
TimeSeriesClient$.checkTimeSeriesReqValidity(tsItem)
```

*Arguments:*

tsItem Timeseries Item

*Returns:* NULL if Timeseries id is valid else error string

**Method** .checkKeyTimeseriesProperties(): A helper method to check the Timeseries properties

*Usage:*

TimeSeriesClient\$.checkKeyTimeseriesProperties(tsItem)

*Arguments:*

tsItem Timeseries Item

*Returns:* NULL if Timeseries id is valid else error string

**Method** .asGetAllResponse(): A helper method which converts the JSON response to GetAllResponse Object

*Usage:*

TimeSeriesClient\$.asGetAllResponse(jsonDict)

*Arguments:*

jsonDict JSON Response

*Returns:* DSUserObjectGetAllResponse object

**Method** .asGetResponse(): A helper method which converts the JSON response to GetResponse Object

*Usage:*

TimeSeriesClient\$.asGetResponse(jsonDict)

*Arguments:*

jsonDict JSON Response

*Returns:* DSUserObjectResponse object

**Method** .jsonRequestEncoder(): A helper method that reformats the raw request to JSON format

*Usage:*

TimeSeriesClient\$.jsonRequestEncoder(request)

*Arguments:*

request Raw request

*Returns:* return JSON formatted list

**Method** .jsonResponseDecoder(): A helper method that converts JSON Response to a given class response type

*Usage:*

TimeSeriesClient\$.jsonResponseDecoder(jsonResp, responseType)

*Arguments:*

jsonResp JSON Response

responseType GetResponse or GetAllResponse type

*Returns:* return DSUserObjectGetAllResponse or DSUserObjectGetResponse object

**Method GetAllItems():** This method returns all the current timeseries you can use in Datas-  
stream queries.

*Usage:*

TimeSeriesClient\$GetAllItems()

*Returns:* DSUserObjectGetAllResponse object

**Method GetItem():** GetItem returns the details for an individual timeseries.

*Usage:*

TimeSeriesClient\$GetItem(itemId)

*Arguments:*

itemId : a valid timeseries Id.

*Returns:* DSUserObjectResponse object

**Method CreateItem():** This method attempts to create the given DSTimeSeriesRequestObject  
via the API service

*Usage:*

TimeSeriesClient\$CreateItem(newItem, overWrite = FALSE, skipItemReturn = FALSE)

*Arguments:*

newItem A DSTimeSeriesRequestObject containing the data used for creating the Timeseries.

overWrite If the given Timeseries Id already exists on the system, the create call will be re-  
jected. Set overWrite = True to overwrite the existing item with new Timeseries.

skipItemReturn : Upon successful creation of an item, the server requests the new item from  
the mainframe and returns it in the response object. For faster processing, set skipItemRe-  
turn = True to skip returning the object in the response

*Returns:* DSUserObjectResponse object

**Method UpdateItem():** This method attempts to modify a timeseries item using the given  
DSTimeSeriesRequestObject via the API service

*Usage:*

TimeSeriesClient\$updateItem(item, skipItemReturn = FALSE)

*Arguments:*

item A DSTimeSeriesRequestObject containing the data used for creating the Timeseries.

skipItemReturn Upon successful creation of an item, the server requests the new item from the  
mainframe and returns it in the response object. For faster processing, set skipItemReturn  
= True to skip returning the object in the response.

*Returns:* DSUserObjectResponse object

**Method DeleteItem():** DeleteItem allows you to delete an existing timeseries

*Usage:*

TimeSeriesClient\$DeleteItem(itemId)

*Arguments:*

itemId a valid timeseries Id.

*Returns:* No return value

**Method** GetTimeseriesDateRange(): This method allows you to determine the supported dates between supplied start and end dates at a specified frequency.

*Usage:*

```
TimeSeriesClient$GetTimeseriesDateRange(
    startDate,
    endDate,
    frequency = DSUserObjectFrequency$Daily
)
```

*Arguments:*

startDate A date specifying the beginning of the date range

endDate A date specifying the end of the date range

frequency A DSUserObjectFrequency enumeration defining if the frequency should be daily, weekly, monthly, quarterly or yearly.

*Returns:* DSTimeSeriesDateRangeResponse object

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
TimeSeriesClient$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Note

: You need a Datastream ID which is permissioned to access the Datastream APIs. In addition, this ID also needs to be permissioned to access the custom user object service. Attempting to access this service without these permissions will result in a permission denied error response.

: For Daily and Weekly frequencies, if the supplied startDate falls on a weekend or a trading holiday, the returned starting date will be the first trading day before the given start date. If the supplied endDate falls on a weekend or a trading holiday, the returned final date will be the last trading day before the given end date. For Weekly frequencies, this will be the last date which matches the day of the week for the first returned start date.

For Monthly, Quarterly and Yearly frequencies, the returned dates are always the 1st day of each month, quarter or year. The returned start and end dates are always the 1st days of the requested month, quarter or year that the given start and end dates fall within.

: For Daily and Weekly frequencies, if the supplied startDate falls on a weekend or a trading holiday, the returned starting date will be the first trading day before the given start date. If the supplied endDate falls on a weekend or a trading holiday, the returned final date will be the last trading day before the given end date. For Weekly frequencies, this will be the last date which matches the day of the week for the first returned start date.

For Monthly, Quarterly and Yearly frequencies, the returned dates are always the 1st day of each month, quarter or year. The returned start and end dates are always the 1st days of the requested month, quarter or year that the given start and end dates fall within.

**Examples**

```

{
  # first logon with your credentials.
  # Creating a TimeSeriesClient instance with your credentials
  # automatically logs on for you.

  timeseriesClient = TimeSeriesClient$new(NULL, 'YourID', 'YourPwd')

  # query for all your current timeseries items

  itemsResp = timeseriesClient$GetAllItems()
  if (!is.null(itemsResp))
  {
    if (itemsResp$ResponseStatus != DSUserObjectResponseStatus$UserObjectSuccess)
    {
      # Your Datastream Id might not be permissioned for managing
      # user created items on this API

      print(paste('GetAllItems failed with error ',
        names(DSUserObjectResponseStatus)[[itemsResp$ResponseStatus + 1]],
        ': ', itemsResp$ErrorMessage))
    }
    else if (!is.null(itemsResp$UserObjects) & itemsResp$UserObjectsCount > 0)
    {
      # You do have access to some timeseries
      # Here we just put the timeseries details into a dataframe and list them
      print(paste('GetAllItems returned', itemsResp$UserObjectsCount, 'timeseries items.'))
      df = data.frame()

      for (tsItem in itemsResp$UserObjects)
      {
        if (!is.null(tsItem))
        {
          rowdata = list(Id = tsItem$Id,
            LastModified = tsItem$LastModified,
            StartDate = ifelse(!is.null(tsItem$DateInfo), as.character(tsItem$DateInfo$StartDate), ""),
            EndDate = ifelse(!is.null(tsItem$DateInfo), as.character(tsItem$DateInfo$EndDate), ""),
            Frequency = ifelse(!is.null(tsItem$DateInfo), tsItem$DateInfo$Frequency, 0),
            NoOfValues = ifelse(!is.null(tsItem$DateRange), tsItem$DateRange$ValuesCount, 0),
            Desc = tsItem$Description)
          df = rbind(df, rowdata)
        }
      }
      print(df)
    }
  }
  #Example to show how to GetItem
  # query for a specific timeseries

  tsName = 'TSZZZ001'
  tsResponse = timeseriesClient$GetItem(tsName)

```



```

# You may want to put the timeseries request response handling into a common function.
if (!is.null(tsResponse))
{
  # Any request dealing with a single user created item returns a DSUserObjectResponse.
  # This has ResponseStatus property that indicates success or failure

  if (tsResponse$ResponseStatus != DSUserObjectResponseStatus$UserObjectSuccess)
  {
    print(paste('Request failed for timeseries', tsName, 'with error',
              names(DSUserObjectResponseStatus)[[tsResponse$ResponseStatus+1]],
              ':', tsResponse$ErrorMessage))
  }
  else if (!is.null(tsResponse$UserObject))
  {
    # The timeseries item won't be returned if you set SkipItem true
    # in CreateItem or UpdateItem

    # Here we simply display the timeseries data using a dataframe.

    tsItem = tsResponse$UserObject
    metadata = c (Id = tsItem$Id,
                 Desc = tsItem$Description,
                 LastModified = as.character(tsItem$LastModified),
                 StartDate = ifelse (!is.null(tsItem$DateInfo), as.character(tsItem$DateInfo$StartDate), NULL),
                 EndDate = ifelse(!is.null(tsItem$DateInfo), as.character(tsItem$DateInfo$EndDate), NULL),
                 Frequency = ifelse(!is.null(tsItem$DateInfo),
                 names(DSUserObjectFrequency)[[tsItem$DateInfo$Frequency + 1]], NULL),
                 NoOfValues = ifelse(!is.null(tsItem$DateRange), tsItem$DateRange$ValuesCount , 0))

    df = data.frame(metadata)
    print(df)
    if (!is.null(tsItem$DateRange))
    {
      df = data.frame(Dates = sapply(tsItem$DateRange$Dates,
                                    FUN = function(x){ return (as.character(x)) } ),
                    Values = sapply(tsItem$DateRange$Values,
                                    FUN = function(x){ ifelse (is.null(x),
                                                                return (NA_character_ ), return (x) )} ))

      # Values if NULL, is printed as <NA> because, while
      # convertind list to vector either by using as.vector or sapply,
      # the NULL values in the list are deleted. and thus there will
      # be mismatch in no of rows and cannot be put in a dataframe

      print(df)
    }
  }
}
}
}

```

# Index

## \* datasets

- DSTimeSeriesCarryIndicator, [9](#)
- DSTimeSeriesDateAlignment, [11](#)
- DSTimeSeriesFrequencyConversion, [15](#)
- DSUserObjectAccessRights, [20](#)
- DSUserObjectFrequency, [22](#)
- DSUserObjectResponseStatus, [24](#)
- DSUserObjectShareTypes, [25](#)
- DSUserObjectTypes, [26](#)

DataClient, [2](#)

DatastreamR::DSConnect, [2](#), [27](#)

DatastreamR::DSTimeSeriesUserObjectBase, [16](#), [17](#)

DatastreamR::DSUserObjectBase, [16–18](#)

DSConnect, [7](#)

DSTimeSeriesCarryIndicator, [9](#)

DSTimeSeriesDataInput, [10](#)

DSTimeSeriesDateAlignment, [11](#)

DSTimeSeriesDateInfo, [12](#)

DSTimeSeriesDateRange, [13](#)

DSTimeSeriesDateRangeResponse, [14](#)

DSTimeSeriesFrequencyConversion, [15](#)

DSTimeSeriesRequestObject, [16](#)

DSTimeSeriesResponseObject, [17](#)

DSTimeSeriesUserObjectBase, [18](#)

DSUserObjectAccessRights, [20](#)

DSUserObjectBase, [20](#)

DSUserObjectFrequency, [22](#)

DSUserObjectGetAllResponse, [22](#)

DSUserObjectResponse, [23](#)

DSUserObjectResponseStatus, [24](#)

DSUserObjectShareTypes, [25](#)

DSUserObjectTypes, [26](#)

TimeSeriesClient, [27](#)