

Package ‘DatastreamDSWS2R’

July 9, 2026

Type Package

Title Provides a Link Between the 'LSEG Datastream' System and R

Version 1.11.0

Date 2026-07-09

Maintainer Charles Cara <charles.cara@absolute-strategy.com>

Description Provides a set of functions and a class to connect, extract and upload information from the 'LSEG Datastream' database. This package uses the 'DSWS' API. Details of this API are available at <<https://www.lseg.com/en/data-analytics>>. Please report issues at <<https://github.com/CharlesCara/DatastreamDSWS2R/issues>>.

License GPL-3

LazyData TRUE

Depends R (>= 4.1.0)

Imports http2, jsonlite, stringi, stringr, xts, zoo, methods, foreach, dplyr

Suggests testthat

Collate 'DatastreamDSWS2R.R' 'common.R' 'classConstructor.R' 'wrapper.R' 'UCTSUpload.R' 'cbindRobust.R' 'data.R'

Encoding UTF-8

Language en-GB

URL <https://github.com/CharlesCara/DatastreamDSWS2R>

BugReports <https://github.com/CharlesCara/DatastreamDSWS2R/issues>

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Charles Cara [aut, cre, cph] (affiliation: Absolute Strategy Research Ltd)

Repository CRAN

Date/Publication 2026-07-09 16:30:09 UTC

Contents

cbindRobust	2
classconstructor	3
currencyDS2ISO	9
DatastreamDSWS2R	10
getDataStream	10
listRequest	11
myStaticRequestSet	12
staticListRequestSet	12
staticRequest	13
staticRequestSet	14
timeSeriesListRequest	15
timeSeriesRequest	16
UCTSAppend	17
UCTSUpload	18
Index	20

cbindRobust	<i>Function to combine time series that fixes the NA problem</i>
-------------	--

Description

When combining two xts time series in which one series is an empty NA series and the other is a character series, then the normal cbind function will return a time series with the correct number of rows and columns but with every cell occupied with NA. This function overcomes this problem by allowing us to combine an empty series and a character series.

Usage

```
cbindRobust(xts1, xts2)
```

Arguments

- | | |
|------|-------------------------------|
| xts1 | First time series to combine |
| xts2 | Second time series to combine |

classconstructor	dsws
------------------	------

Description

An R5/RC object for accessing the LSEG Datastream DSWS service.

Details

Creates an R5/RC4 object for accessing the LSEG Datastream DSWS service

Fields

tokenList fieldDescription
tokenSource fieldDescription
serverURL fieldDescription
username fieldDescription
password fieldDescription
initialised fieldDescription
errorlist fieldDescription
requestList fieldDescription
jsonResponseSaveFile fieldDescription
jsonResponseLoadFile fieldDescription
dataResponse fieldDescription
symbolList fieldDescription
myValues fieldDescription
myTypes fieldDescription
logging fieldDescription
numDatatype fieldDescription
numInstrument fieldDescription
numRequests fieldDescription
numChunks fieldDescription
chunkLimit fieldDescription
requestStringLimit fieldDescription
logFileFolder fieldDescription

Methods

UCTSUpload(tsData, TSCode = "", MGMTGroup = "MGMTG", freq = c("D", "W", "M", "Q", "Y"), seriesName, Units = "")

Uploads an xts into a UCTS in the Datastream Database

Note this function does not check to see if there is a pre-existing timeseries already in Datastream. It will just overwrite any existing UCTS.

Parameters are:

tsData an xts (or timeseries object that can be converted to one) to be uploaded.

TSCode The mnemonic of the target UCTS

MGMTGroup Must have management group. Only the first characters will be used.

freq The frequency of the data to be uploaded

seriesName the name of the series - can be no more than XX characters - excess will be trimmed to that length

Units Units of the data - can be no more than 12 characters - excess will be trimmed to that length

Decimals Number of Decimals in the data - a number between 0 and 9 - if outside that range then trimmed

ActPer Whether the values are percentages ("N") or actual numbers ("Y")

freqConversion How to do any FX conversions

Alignment Alignment of the data within periods

Carry whether to carry data over missing dates

PrimeCurr the currency of the timeseries

Returns TRUE if the upload has been a success, or FALSE with field 'errorlist' containing the error message

Examples:

```
mysdws <- dsdws$new()
```

```
sPost <- mysdws$UCTSUpload(tsData = testData,
                           TSCode = "TSTEST01",
                           MGMTGroup = "TEST",
                           freq = "D",
                           seriesName = "UCTS Upload Test",
                           Units = "",
                           Decimals = 2,
                           ActPer = "Y",
                           freqConversion = "END",
                           Alignment = "MID",
                           Carry = "NO",
                           PrimeCurr = "")
```

```
expect_equal(sPost, TRUE)
```

```
expect_equal(mysdws$errorlist, NULL)
```

initialize(dsdws.serverURL = "", getTokenFunction = NULL, token = NULL, username = "", password = "", connect = TRUE)
initialises the class. Unless noConnect is TRUE also connects to the Datastream dsdws server.

Authentication can be set in three ways: 1) If getTokenFunction is not null then that function is called. It is expected to return a list with items 'TokenValue' and 'TokenExpiry'.

2) An access token can also be passed into the class on initialisation, so that it can be shared between sessions. 'token' is expected to be a list with items 'TokenValue' and 'TokenExpiry'.

3) A username and password that are used to fetch a token from the DSWS server. If the username and password are not provided, then they are sourced from system environment variables (ie Sys.getenv) 'DatastreamUsername' and 'DatastreamPassword' or alternatively (not preferred) then from options()\$Datastream.Username and options()\$Datastream.Password

This allows the password to be stored in .Renvirom or .RProfile rather than in the source code.

There different accounts have different limits according to their licence. Most users are limited to 50 items while enterprise users have a limit of 2000L. The chunk limit can be controlled by setting the chunkLimit parameter of the dsws object. If options()\$Datastream.ChunkLimit is set then the value is taken from there.

`listRequest(instrument, datatype = "", expression = "", requestDate)` Make a listRequest from Datastream DSWS. This is the equivalent to the Excel static request for a list.

Parameters are:

instrument should contain a list mnemonic, such as 'LFTSE100' Can be a user created list or index. The UCL can contain expressions

datatype array of datatypes eg NAME, MNEM, P, PE etc

expression if datatype is null or "" then an expression eg PCH#(XXXX,3M)

requestDate either a Date or a string with a datastream relative date eg '-3M'

Returns a data.frame with the requested data.

Examples:

```
mysdws$listRequest(instrument = "LFTSE100",
  datatype = c("NAME", "P"),
  requestDate = "-00")
```

```
mysdws$listRequest(instrument = "LFTSE100",
  expression = "PCH#(XXXX,3M)", requestDate = Sys.Date())
```

`snapshotRequest(instrument, datatype = "", expression = "", requestDate)` Make a snapshotRequest from Datastream DSWS. This is the equivalent to the Excel static request for an array of instruments.

Parameters are:

instrument should one or more instruments eg "MKS" or c("MKS", "@AAPL"). The array can contain Economics codes and Expressions.

datatype array of datatypes eg NAME, MNEM, P, PE etc

expression if datatype is null or "" then an expression eg PCH#(XXXX,3M)

requestDate either a Date or a string with a datastream relative date eg '-3M'

Returns a data.frame with the requested data.

Examples:

```

mysdws$snapshotRequest(instrument = c("MKS","AAPL"),
  datatype = c("NAME","P"), requestDate = "-0D")

mysdws$snapshotRequest(instrument = c("MKS","AAPL"),
  expression = "PCH#(XXXX,3M)", requestDate = "-0D")

```

```
timeSeriesListRequest( instrument, datatype = "", expression = "", startDate, endDate, frequency = "D", format = "ByInstrument")
```

Make a timeSeriesListRequest from Datastream DSWs. This is the equivalent to the Excel timeseries request for an array of instruments. Should request either a datatype or an expression not both. If a datatype is provided then anything in Expression will be ignored.

Parameters are:

instrument should contain a list mnemonic, such as "LFTSE100" . Can be a user created list or index. The UCL can contain expressions.

datatype array of datatypes eg P, PE etc

expression if datatype is null or "" then an expression
eg PCH#(XXXX,3M)

startDate either a Date or a string with a datastream relative date
eg '-3M'

endDate either a Date or a string with a datastream relative date
eg '-0D'

frequency one of the standard Datastream frequencies - D, W, M, Q, or Y

format can be either "ByInstrument" or "ByDatatype".

Returns either a single xts or a list of xts a data.frame with the requested data. If "ByInstrument" then the data is returned as one or more (ie a list) wide xts with one column per instrument. If "ByDatatype" then the data is returned as one or more (ie a list) of wide xts with one column per Datatype. This format is more compatible with the quantmod package.

Examples:

```

mysdws$timeSeriesListRequest(instrument = "LFTSE100",
  datatype = "P", startDate = "-30D",
  endDate = "-0D", frequency = "D")

mysdws$timeSeriesListRequest(instrument = "LFTSE100",
  expression = "PCH#(XXXX,3M)",
  startDate = "-30D",
  endDate = "-0D",

```

```
frequency = "D")
```

```
mysdws$timeSeriesListRequest(instrument = "LFTSE100",
```

```
datatype = ("P","UP"), startDate = "-30D",
```

```
endDate = "-0D",
```

```
frequency = "D", format = "ByDatatype")
```

```
timeSeriesRequest( instrument, datatype = "", expression = "", startDate, endDate, frequency = "D", format
```

Return a timeSeriesRequest from Datastream dsws. Should request either a datatype or an expression not both. If a datatype is provided then anything in Expression will be ignored.

Make a timeSeriesRequest from Datastream DSWS. This is the equivalent to the Excel time-series request for an array of instruments.

Parameters are:

instrument should one or more instruments eg "MKS" or c("MKS","@AAPL"). The array can contain Economics codes and Expressions.

datatype array of datatypes eg P, PE etc

expression if datatype is null or "" then an expression eg PCH#(XXXX,3M)

startDate either a Date or a string with a datastream relative date eg '-3M'

endDate either a Date or a string with a datastream relative date eg '-0D'

frequency one of the standard Datastream frequencies - D, W, M, Q, or Y

format can be either "ByInstrument" or "ByDatatype".

Returns either a single xts or a list of xts a data.frame with the requested data. If "ByInstrument" then the data is returned as one or more (ie a list) wide xts with one column per instrument. If "ByDatatype" then the data is returned as one or more (ie a list) of wide xts with one column per Datatype. This format is more compatible with the quantmod package.

Examples:

```
mysdws$timeSeriesRequest(instrument = c("MKS", "@AAPL"),
```

```
datatype = "P", startDate = "-30D",
```

```
endDate = "-0D", frequency = "D")
```

```
mysdws$timeSeriesRequest(instrument = c("MKS"),
```

```
expression = "PCH#(XXXX,3M)", startDate = "-30D",
```

```
endDate = "-0D", frequency = "D")
```

```
mysdws$timeSeriesRequest(instrument = c("MKS", "@AAPL"),
```

```

datatype = ("P","UP"), startDate = "-30D",
endDate = "-0D", frequency = "D", format = "ByDatatype")

```

Examples

Not run:

```

mysdws <- dsws$new()
# Snapshot requests

myData <- mysdws$snapshotRequest(instrument = c("ABF","RIO","WPP"),
                                datatype = "P",
                                requestDate = "0D")

myData <- mysdws$snapshotRequest(instrument = c("ABF","RIO","WPP"),
                                expression = "PCH#(XXXX,3M)",
                                requestDate = "0D")
myData <- mysdws$listRequest(instrument = "LFTSE100", datatype = "P", requestDate = "0D")

mysdws$snapshotRequest(instrument = c("SWCNB10","UKEUSCCIR"),
                       datatype = c("MNEM","UPDATE"),
                       requestDate = "0D")
mysdws$snapshotRequest(instrument = c("VOD", "HSBA"),
                       datatype="QTEALL",
                       requestDate = Sys.Date())
mysdws$snapshotRequest(instrument = "STATS",
                       datatype = "DS.USERSTATS",
                       requestDate = Sys.Date())

# Timeseries requests

xtsData <- mysdws$timeSeriesRequest(instrument = "MKS",
                                   datatype = "MV",
                                   startDate = "-30D",
                                   endDate = "-0D",
                                   frequency = "D")

xtsData <- mysdws$timeSeriesListRequest(instrument = "LFTSE100",
                                       datatype = "MV",
                                       startDate = "-30D",
                                       endDate = "-0D",
                                       frequency = "D")

# UCTS upload

testData <- xts::xts(x = c(1, 2.2, 3.12345, 4.5),

```



```

order.by = as.Date(c("2014-04-22", "2014-04-23", "2014-04-24", "2014-04-25"))

mysws <- dsws$new()

sPost <- mydsws$UCTSUpload(TSCode = "TSTEST01",
                           MGMTGroup = "TEST",
                           freq = "D",
                           seriesName = "Upload Demonstration",
                           Units = "",
                           Decimals = 2,
                           ActPer = "Y",
                           freqConversion = "END",
                           Alignment = "MID",
                           Carry = "NO",
                           PrimeCurr = "",
                           tsData = testData)

## End(Not run)

```

currencyDS2ISO

Conversion table of Datastream to ISO currency codes

Description

Conversion table of Datastream to ISO currency codes

Usage

```
data(currencyDS2ISO)
```

Format

A data frame with 161 rows and 3 variables:

dsCode the datastream code

isoCode the ISO code for the currency

primeCode primaryCode for currency or alternative

Multiplier the units of the currency

DataStreamDSWS2R

DatastreamDSWS2R

Description

A package to manage access to the LSEG Datastream DSWS webservice

Author(s)

Maintainer: Charles Cara <charles.cara@absolute-strategy.com> (affiliation: Absolute Strategy Research Ltd) [copyright holder]

Authors:

- Charles Cara <charles.cara@absolute-strategy.com> (affiliation: Absolute Strategy Research Ltd) [copyright holder]

See Also

Useful links:

- <https://github.com/CharlesCara/DatastreamDSWS2R>
- Report bugs at <https://github.com/CharlesCara/DatastreamDSWS2R/issues>

getDataStream

Initialise connection with Datastream DSWS server (Deprecated)

Description

getDataStream initialises an R5 object that contains a connection with the Datastream DWE server. This function has been provided for backward compatibility

Usage

```
getDataStream(
  dweURLwsdl = "",
  User = as.character("USERNAME"),
  Pass = as.character("PASSWORD")
)
```

Arguments

dweURLwsdl	Ignored
User	Ignored - now sourced from options()\$DataStream.Username
Pass	Ignored - now sourced from options()\$DataStream.Password

Details

Initialise connection with Datastream DSWs server. Provided for backwards compatibility

Value

a dsws object

listRequest	<i>Make a list request for static data (Deprecated)</i>
-------------	---

Description

listRequest Function that returns a the value of Expression for the instrument list in DSCode from Datastream

Usage

```
listRequest(
    dwei = getDataStream(),
    DSCode,
    Expression = "",
    startDate = Sys.Date(),
    endDate = Sys.Date(),
    frequency = "D",
    verbose = FALSE
)
```

Arguments

dwei	- A Datastream Client Interface object created with getDataStream
DSCode	- the constituent list for the request eg LDJSTOXX
Expression	- the data to return eg MNEM or NAME. If NULL or "" then we will return the code that has been loaded into the User Created List.
startDate	- the date of the request, or the string "TODAY"
endDate	- Ignored
frequency	- the frequency of the request
verbose	- whether to give messages during the request

Details

Make a list request for static data

Value

returns an array of the requested information

myStaticRequestSet	<i>myStaticRequestSet (Deprecated)</i>
--------------------	--

Description

internal function for requesting an expression for an array of instruments. The function will initially try a snapshot request, and if this fails try a timeseries request.

Usage

```
myStaticRequestSet(  
  mydsws = dsws$new(),  
  instrument,  
  iExpression,  
  endDate = Sys.Date(),  
  frequency = "D"  
)
```

Arguments

mydsws	a dsws object, if not provided a new one will be created
instrument	array of instruments
iExpression	an expression such as PCH#(XXXX,1M)
endDate	the date of the request
frequency	optional frequency defaults to "D"

Details

Internal function

Value

a dataframe of the

staticListRequestSet	<i>staticListRequestSet</i>
----------------------	-----------------------------

Description

This function creates a dataframe set of static list requests for a constituent list

Usage

```
staticListRequestSet(  
  mydsws = dsws$new(),  
  instrument,  
  expression = "",  
  endDate = Sys.Date(),  
  frequency = "D"  
)
```

Arguments

- mydsws a dsws object, if not provided a new one will be created
- instrument array of instruments
- expression an array of expressions such as PCH#(XXXX,1M)
- endDate the date of the request
- frequency optional frequency defaults to "D"

Details

This function creates a dataframe set of static list requests for a constituent list

Value

a dataframe of the data

staticRequest	<i>make a static request (Deprecated)</i>
---------------	---

Description

makes a static (or snapshot request) from the Datastream DSWS server

Usage

```
staticRequest(  
  dwei = getDataStream(),  
  DSCode,  
  Expression = "",  
  endDate = Sys.Date(),  
  frequency = "D",  
  verbose = FALSE,  
  noCache = FALSE  
)
```

Arguments

dwei	- A Datastream Client Interface object created with getDataStream
DSCode	- an array of instruments eg c("RIO","MKS")
Expression	- the data to return eg MNEM or NAME
endDate	- the date of the request, or the string "TODAY"
frequency	- the frequency of the request
verbose	- whether to give messages during the request
noCache	- no longer used

Details

staticRequest Function that returns a the value of Expression for the array of instruments in DSCode from Datastream parameters are

Value

returns an array of the requested information

staticRequestSet	<i>staticRequestSet</i>
------------------	-------------------------

Description

This function creates a dataframe set of static requests for a set of stocks/indices

Usage

```
staticRequestSet(
  mydsws = dsws$new(),
  instrument,
  expression = "",
  endDate = Sys.Date(),
  frequency = "D",
  verbose = FALSE
)
```

Arguments

mydsws	a dsws object, if not provided a new one will be created
instrument	array of instruments
expression	an array of expressions such as PCH#(XXXX,1M) or Dataitems
endDate	the date of the request
frequency	optional frequency defaults to "D"
verbose	whether to display messages as making the request

Details

return a dataframe of static data

Value

a dataframe of the data

timeSeriesListRequest *make a timeSeries request for a list (Deprecated)*

Description

make a timeseries request for a constituent list from Datastream DSWS timeSeriesListRequest
Function that returns a timeseries from Datastream constituent list parameters are

Usage

```
timeSeriesListRequest(
  dwei = getDataStream(),
  DSCode,
  Instrument,
  startDate,
  endDate = Sys.Date(),
  frequency = "D",
  sStockList,
  aTimeSeries,
  verbose = FALSE
)
```

Arguments

dwei	- A Datastream Client Interface object created with getDataStream
DSCode	- the constituent list requested eg 'LFTSE100'
Instrument	- the expression to return for each member of constituent list
startDate	- the start date of the timeseries
endDate	- the end date of the timeseries
frequency	- the frequency of the request
sStockList	- variable that is returned with list of of the stocks
aTimeSeries	- variable that is returned with the set of timeseries
verbose	- whether to give messages during the request

Details

List request

Value

whether the request has been successful , but also in sStockList: a list a two element vector of the displayname and symbol for each timeseries in aTimeSeries: a list of class xts with the requested timeseries information

timeSeriesRequest	<i>make a timeseries request (Deprecated)</i>
-------------------	---

Description

make a timeseries request from the Datastream DSWS server

Usage

```
timeSeriesRequest(
  dwei = getDataStream(),
  DSCodes = "",
  Instrument = "",
  startDate = Sys.Date(),
  endDate = Sys.Date(),
  frequency = "D",
  sStockList,
  aTimeSeries,
  myType = "numeric",
  verbose = FALSE
)
```

Arguments

dwei	- A Datastream Client Interface object created with getDataStream
DSCodes	- one or more codes to return, eg "MKS" or c("MKS", "SAB")
Instrument	- the instrument or expression to return eg PCH#(XXXX,1M)
startDate	- the start date of the timeseries
endDate	- the end date of the timeseries
frequency	- the frequency of the request
sStockList	- variable that is returned with list of the stocks
aTimeSeries	- variable that is returned with the set of timeseries. This is a list that is not guaranteed to be in the same order as sStockList
myType	- the type of the return values eg numeric (default), Date or Character
verbose	- whether to give messages during the request

Details

function timeSeriesRequest obtains a timeseries from Datastream

Value

whether the request has been successful in sStockList: a list a two element vector of the displayname and symbol for each timeseries in aTimeseries: a list of class xts with the requested timeseries information

UCTSAppend	<i>Append a xts to an existing UCTS timeseries in Datastream</i>
------------	--

Description

Uploads and appends an xts into a UCTS in the Datastream Database

Usage

```
UCTSAppend(
  tsData,
  TSCode = "",
  MGMTGroup = "ABC",
  freq = c("D", "W", "M", "Q", "Y"),
  seriesName,
  Units = "",
  Decimals = 2,
  ActPer = c("N", "Y"),
  freqConversion = c("ACT", "SUM", "AVG", "END"),
  Alignment = c("1ST", "MID", "END"),
  Carry = c("YES", "NO", "PAD"),
  PrimeCurr = "",
  overwrite = TRUE,
  mydsws = dsws$new(),
  strUsername = NULL,
  strPassword = NULL,
  strServerName = "https://product.datastream.com",
  strServerPage = "/dswsclient/V1/DSUserDataService.svc/rest/UpdateItem"
)
```

Arguments

tsData	- an xts (or timeseries object that can be converted to one) to be uploaded.
TSCode	The mnemonic of the target UCTS
MGMTGroup	Must have managment group. Only the first characters will be used.
freq	The frequency of the data to be uploaded
seriesName	the name of the series
Units	Units of the data - can be no more than 12 characters - excess will be trimmed to that length

Decimals	Number of Decimals in the data - a number between 0 and 9 - if outside that range then trimmed
ActPer	Whether the values are percentages ("N") or actual numbers ("Y")
freqConversion	How to do any FX conversions
Alignment	Alignment of the data within periods
Carry	whether to carry data over missing dates
PrimeCurr	the currency of the timeseries
overwrite	if TRUE then existing data in the UCTS will be overwritten
mysws	a dsws object that can be passed in. Use this to avoid creating another dsws object in the same session.
strUsername	your Datastream username
strPassword	your Datastream Password
strServerName	URL of the Datastream server
strServerPage	page on the datastream server

Details

This function checks if there is a pre-existing timeseries already in Datastream. If there is then it will append the xts onto the existing series. If there are any overlapping dates then depending on the setting of overwrite then the new data will overwrite the existing data in the UCTS

Value

TRUE if the upload has been a success, otherwise an error message

UCTSUpload	<i>Upload a UCTS timeseries into Datastream</i>
------------	---

Description

Uploads an xts into a UCTS in the Datastream Database

Usage

```
UCTSUpload(
  tsData,
  TSCode = "",
  MGMTGroup = "ABC",
  freq = c("D", "W", "M", "Q", "Y"),
  seriesName,
  Units = "",
  Decimals = 2,
  ActPer = c("N", "Y"),
  freqConversion = c("ACT", "SUM", "AVG", "END"),
```

```

Alignment = c("1ST", "MID", "END"),
Carry = c("YES", "NO", "PAD"),
PrimeCurr = "",
strUsername = NULL,
strPassword = NULL,
mysdws = dsds$new(),
strServerName = "https://product.datastream.com",
strServerPage = "/dsdwsclient/V1/DSUserDataService.svc/rest/UpdateItem"
)

```

Arguments

tsData	- an xts (or timeseries object that can be converted to one) to be uploaded.
TSCode	The mnemonic of the target UCTS
MGMTGroup	Must have management group. Only the first characters will be used.
freq	The frequency of the data to be uploaded
seriesName	the name of the series - can be no more than XX characters - excess will be trimmed to that length
Units	Units of the data - can be no more than 12 characters - excess will be trimmed to that length
Decimals	Number of Decimals in the data - a number between 0 and 9 - if outside that range then trimmed
ActPer	Whether the values are percentages ("N") or actual numbers ("Y")
freqConversion	How to do any FX conversions
Alignment	Alignment of the data within periods
Carry	whether to carry data over missing dates
PrimeCurr	the currency of the timeseries
strUsername	Deprecated will be removed in a future release - ignored
strPassword	Deprecated will be removed in a future release - ignored
mysdws	a dsds connection object
strServerName	URL of the Datastream server
strServerPage	page on the datastream server

Details

Note this function does not check to see if there is a pre-existing timeseries already in Datastream. It will just overwrite any existing UCTS.

Value

TRUE if the upload has been a success, FALSE with attribute error containing the error message

Index

* **datasets**

currencyDS2ISO, [9](#)

cbindRobust, [2](#)

classconstructor, [3](#)

currencyDS2ISO, [9](#)

DatastreamDSWS2R, [10](#)

DatastreamDSWS2R-package
(DatastreamDSWS2R), [10](#)

dsws (classconstructor), [3](#)

getDataStream, [10](#)

listRequest, [11](#)

myStaticRequestSet, [12](#)

staticListRequestSet, [12](#)

staticRequest, [13](#)

staticRequestSet, [14](#)

timeSeriesListRequest, [15](#)

timeSeriesRequest, [16](#)

UCTSAppend, [17](#)

UCTSUpload, [18](#)