Package ‘DSI’

May 18, 2020

**Type** Package

**Title** 'DataSHIELD' Interface

**Version** 1.1.0

**Description** 'DataSHIELD' is an infrastructure and series of R packages that enables the remote and 'non-disclosive' analysis of sensitive research data. This package defines the API that is to be implemented by 'DataSHIELD' compliant data repositories.

**Depends** R (≥ 3.1), methods, progress, R6

**Suggests** testthat (≥ 2.1.0)

**License** LGPL (≥ 2.1)

**URL** http://datashield.ac.uk

**BugReports** https://github.com/datashield/DSI

**RoxygenNote** 7.1.0

**Encoding** UTF-8

**LazyData** true

**Collate** 'DSObject.R'
  'hidden.R'
  'DSConnection.R'
  'DSDriver.R'
  'DSLoginBuilder.R'
  'DSResult.R'
  'datashield.aggregate.R'
  'datashield.assign.R'
  'datashield.connections.R'
  'datashield.errors.R'
  'datashield.list.R'
  'datashield.login.R'
  'datashield.logout.R'
  'datashield.status.R'
R topics documented:

'datashield.symbol.R'
'datashield.workspace.R'
'rd.R'
'utils.R'

R topics documented:

datashield.aggregate .............................. 3
datashield.assign ................................. 4
datashield.assign.expr ............................ 5
datashield.assign.resource ......................... 6
datashield.assign.table .......................... 7
datashield.connections ............................ 8
datashield.connections_default .................... 9
datashield.connections_find ....................... 10
datashield.errors ................................ 11
datashield.login ................................... 11
datashield.logout .................................. 13
datashield.methods ................................. 14
datashield.method_status ........................... 14
datashield.pkg_check .............................. 15
datashield.pkg_status .............................. 15
datashield.resources ............................... 16
datashield.resource_status ......................... 16
datashield.rm .................................... 17
datashield.symbols ................................. 17
datashield.tables .................................. 17
datashield.table_status ............................. 18
datashield.workspaces .............................. 18
datashield.workspace_rm ............................ 19
datashield.workspace_save ......................... 19
dsAggregate ........................................ 20
dsAssignExpr ....................................... 20
dsAssignResource .................................. 21
dsAssignTable ...................................... 22
dsConnect ........................................... 23
DSConnection-class ................................ 24
dsDisconnect ........................................ 25
DSDriver-class ...................................... 26
dsFetch .............................................. 26
dsGetInfo ........................................... 27
dsHasResource ...................................... 28
dsHasTable .......................................... 28
dsIsAsync ........................................... 29
dsListMethods ...................................... 30
dsListPackages ..................................... 31
dsListResources .................................... 31
dsListSymbols ...................................... 32
datashield.aggregate

Description

Aggregates the expression result using the specified aggregation method in the current Datashield session.

Usage

datashield.aggregate(conns, expr, async = TRUE)

Arguments

- **conns**: `DSConnection-class` object or a list of `DSConnection-class`es.
- **expr**: Expression to evaluate.
- **async**: Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.

Value

The result of the aggregation

Examples

```r
## Not run:
# call aggregate function on server side
datashield.aggregate(conns, expr = quote(someFunction(D, 123)))

## End(Not run)
```
datashield.assign  Data assignment

Description

Assign a table or an expression result to a R symbol in the Datashield R session.

Usage

```r
datashield.assign(
  conns,
  symbol,
  value,
  variables = NULL,
  missings = FALSE,
  identifiers = NULL,
  id.name = NULL,
  async = TRUE
)
```

Arguments

- `conns`  
  DSConnection-class object or a list of DSConnection-classes.
- `symbol`  
  Name of the R symbol.
- `value`  
  Fully qualified name of a table reference in data repositories (see datashield.assign.table for more details) or a R expression with allowed assign functions calls.
- `variables`  
- `missings`  
  If TRUE, missing values will be pushed from data repository to R, default is FALSE. Ignored if value is an R expression.
- `identifiers`  
  Name of the identifiers mapping to use when assigning entities to R (if supported by data repository).
- `id.name`  
  Name of the column that will contain the entity identifiers. If not specified, the identifiers will be the data frame row names. When specified this column can be used to perform joins between data frames.
- `async`  
  Whether the result of the call should be retrieved asynchronously (TRUE means that calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests).

Examples

```r
## Not run:
# assign a list of variables from table HOP
```

---

The `datashield.assign` function in Datashield allows you to assign a table or an expression result to a R symbol in the Datashield R session. The function takes several arguments:

- `conns`: A DSConnection-class object or a list of DSConnection-classes.
- `symbol`: The name of the R symbol.
- `value`: A fully qualified name of a table reference in data repositories or a R expression with allowed assign functions calls.
- `variables`: A list of variable names or a Javascript expression that selects the variables of a table.
- `missings`: A logical value indicating whether missing values should be pushed from the data repository to R. Defaults to FALSE.
- `identifiers`: The name of the identifiers mapping to use when assigning entities to R.
- `id.name`: The name of the column that will contain the entity identifiers.
- `async`: A logical value indicating whether the result of the call should be retrieved asynchronously. Defaults to TRUE.

The function is used to assign data or expressions to R symbols, facilitating data manipulation and analysis within the Datashield framework.
datashield.assign(expr)

**Expression result assignment**

Description

Assign the result of the execution of an expression to a R symbol in the Datashield R session.

Usage

datashield.assign(expr(conns, symbol, expr, async = TRUE)

Arguments

- **conns**: DSConnection-class object or a list of DSConnection-classes.
- **symbol**: Name of the R symbol.
- **expr**: R expression with allowed assign functions calls.
- **async**: Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.

Examples

## Not run:
# assign a G symbol
datashield.assign(expr(conns, symbol = "G", expr = quote(as.numeric(D$GENDER)))

## End(Not run)
Description

Assign a resource object of class 'ResourceClient' to a R symbol in the Datashield R session.

Usage

datashield.assign.resource(conns, symbol, resource, async = TRUE)

Arguments

- **conns**: DSConnection-class object or a list of DSConnection-classes.
- **symbol**: Name of the R symbol.
- **resource**: Fully qualified name of a resource reference in the data repository (can be a vector or must be the same in each data repository); or a named list of fully qualified resource names (one per server name); or a data frame with 'server' and 'resource' columns (such as the one that is used in datashield.login).
- **async**: Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.

Examples

```r
## Not run:
# assign a resource HOP
datashield.assign.resource(conn, symbol="rsrc", resource="demo.HOP")

# assign the tables that are defined in the logindata (server' and 'resource' columns are expected) data frame that is used in datashield.login() function. Connections names and server names must match.
datashield.assign.resource(conns, "rsrc", logindata)

# assign the resources that are defined in the provided named list. Connections names # and server names must match.
datashield.assign.resource(conns, "rsrc",
  list(server1="datashield.CNSIM1", server2="datashield.CNSIM2"))
## End(Not run)
```
datashield.assign.table

Table assignment

Description

Assign a table to a R symbol in the Datashield R session.

Usage

datashield.assign.table(
  conns,
  symbol,
  table,
  variables = NULL,
  missings = FALSE,
  identifiers = NULL,
  id.name = NULL,
  async = TRUE
)

Arguments

conns  

DSConnection-class object or a list of DSConnection-classes.

symbol  

Name of the R symbol.

table  

Fully qualified name of a table in the data repository (can be a vector or must be the same in each data repository); or a named list of fully qualified table names (one per server name); or a data frame with 'server' and 'table' columns (such as the one that is used in datashield.login)

variables  

List of variable names or Javascript expression that selects the variables of a table. See javascript documentation: http://opaldoc.obiba.org/en/latest/magma-user-guide/variable/

missings  

If TRUE, missing values will be pushed from data repository to R, default is FALSE. Ignored if value is an R expression.

identifiers  

Name of the identifiers mapping to use when assigning entities to R (if supported by the data repository).

id.name  

Name of the column that will contain the entity identifiers. If not specified, the identifiers will be the data frame row names. When specified this column can be used to perform joins between data frames.

async  

Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.
datashield.connections

List the DSConnection objects in the analytic environment

Description

This function identifies and prints all DSConnection-class objects in the analytic environment. If there are no DSConnection servers in the analytic environment, datashield.connections reminds the user that they have to login to a valid set of DataSHIELD aware servers. If there is only one set of DSConnections, it copies that one set and names the copy 'default.connections'. This default set will then be used by default by all subsequent calls to client-side functions. If there is more than one set of DSConnections in the analytic environment, datashield.connections tells the user that they can either explicitly specify the DSConnections to be used by each client-side function by providing an explicit "datasources=" argument for each call, or can alternatively use the datashield.connections_default function to specify a default set of DSConnections to be used by all client-side calls unless over-ruled by the 'datasources=' argument.

Usage

datashield.connections(env = getOption("datashield.env", globalenv()))

Arguments

env The environment where to search for the connection symbols. Try to get it from the 'datashield.env' option, with default to the Global Environment.
**Value**

Returns a list of `DSConnection-class` objects and advises the user how best to respond depending whether there are zero, one or multiple connections detected.

**See Also**

Other Connections management: `datashield.connections_default()`, `datashield.connections_find()`

datashield.connections_default

*Set or get the default list of DSConnection objects in the analytic environment*

**Description**

By default if there is only one set of `DSConnection-class` objects that is available for analysis, all DataSHIELD client-side functions will use that full set of DSConnections unless the 'datasources=' argument has been set and specifies that a particular subset of those DSConnections should be used instead. The correct identification of the full single set of opals is based on the `datashield.connections_find` function. To illustrate, if the single set of Opals is called 'study.opals' and consists of six servers numbered studies[1] to studies[6] then all client-side functions will use data from all six of these 'studies' unless, say, datasources=studies[c(2,5)] is declared and only data from the second and fifth studies will then be used. On the other hand, if there is more than one set of DSConnections in the analytic environment client-side functions will be unable to determine which set to use. The function `datashield.connections_find` has therefore been written so that if one of the DSConnection sets is called 'default.connections' then that set - i.e. 'default.connections' - will be selected by default by all DataSHIELD client-side functions. If there is more than one set of DSConnections in the analytic environment and NONE of these is called 'default.connections', the function `datashield.connections_find` will fail. Therefore `datashield.connections_default` copies the provided set of DSConnections as 'default.connections'. This set will then be selected by default by all client-side functions, unless it is deleted and an alternative set of DSConnections is copied and named 'default.connections'. Regardless how many sets of DSConnections exist and regardless whether any of them may be called 'default.connections', the 'datasources=' argument overrides the defaults and allows the user to base his/her analysis on any set of DSConnections and any subset of those DSConnections.

**Usage**

datashield.connections_default(
  name = NULL,
  env =getOption("datashield.env", globalenv())
)
datashield.connections.find

Arguments

name  Symbol name that identifies the set of DSConnection-class objects to be used by default. If not provided, the 'default.connections' variable value is returned.

env  The environment where to search for the connection symbols. Try to get it from the 'datashield.env' option, with default to the Global Environment.

Value

The 'default.connections' value from the analytic environment or NULL if the 'default.connections' symbol is not defined.

See Also

Other Connections management: datashield.connections_find(), datashield.connections()

datashield.connections_find

Search for DSConnection objects in the analytic environment

Description

If the user does not set the argument 'datasources' in the client side analysis functions, this function is called to search for DSConnection-class objects in the environment (default environment is the Global one). If one set of DSConnection objects is found, it is assigned to 'default.connections' symbol in the analytic environment. If more than one set of DSConnection objects is found and none of them is called 'default.connections', the function stops and suggests user to use the datashield.connections_default function.

Usage

datashield.connections_find(env = getOption("datashield.env", globalenv()))

Arguments

env  The environment where to search for the connection symbols. Try to get it from the 'datashield.env' option, with default to the Global Environment.

Value

Returns a list of DSConnection-class objects or stops the process

See Also

Other Connections management: datashield.connections_default(), datashield.connections()
datashield.errors

**List R last errors**

**Description**

Get the R last errors available after the datashield.assign or datashield.aggregate calls in the Datashield R session.

**Usage**

```r
datashield.errors()
```

---

datashield.login

**Logs in a DataSHIELD R sessions and optionally assigns variables to R**

**Description**

This function allows for clients to login to data repository servers and (optionaly) assign all the data or specific variables from the data repositories tables to R data frames. The assigned dataframes (one for each data repository) are named 'D' (by default). Different login strategies are supported: using a certificate/private key pair (2-way SSL encryption mechanism), using user credentials (user name and password) or using a personal access token (could be combined with a user name, depending on the data repository system).

**Usage**

```r
datashield.login(
  logins = NULL,
  assign = FALSE,
  variables = NULL,
  missings = FALSE,
  symbol = "D",
  id.name = NULL,
  opts = getOption("datashield.opts", list()),
  restore = NULL
)
```

**Arguments**

- **logins** A dataframe table that holds login details. This table holds five elements required to login to the servers where the data to analyse is stored. The expected column names are 'driver' (the DSDriver-class name, default is "OpalDriver"), 'server' (the server name), url' (the server url), 'user' (the user name or the certificate PEM file path), 'password' (the user password or the private key PEM file path), 'token' (the personal access
datashield.login

token, ignored if 'user' is defined), 'table' (the fully qualified name of the table in the data repository), 'resource' (the fully qualified name of the resource reference in the data repository), 'options' (the SSL options).

An additional column 'identifiers' can be specified for identifiers mapping (if supported by data repository). See also the documentation of the examplar input table logindata for details of the login elements.

assign
A boolean which tells whether or not data should be assigned from the data repository table to R after login into the server(s).

variables
Specific variables to assign. If assign is set to FALSE this argument is ignored otherwise the specified variables are assigned to R. If no variables are specified (default) the whole data repository’s table is assigned.

missings
If TRUE, missing values will be pushed from data repository to R, default is FALSE.

symbol
A character, the name of the data frame to which the data repository’s table will be assigned after login into the server(s).

id.name
Name of the column that will contain the entity identifiers. If not specified, the identifiers will be the data frame row names. When specified this column can be used to perform joins between data frames.

opts
Default SSL options to be used in case it is not specified in the logins structure.

restore
The workspace name to restore (optional).

Value
object(s) of class DSConnection

Examples

## Not run:

#### The below examples illustrate an analyses that use test/simulated data ####

# build your data.frame
builder <- newDSLoginBuilder()
builder$append(server="server1", url="https://opal-demo.obiba.org", table="datashield.CNSIM1", resource="datashield.CNSIM1r", user="dsuser", password="password", options="list(ssl_verifyhost=0,ssl_verifypeer=0)"
builder$append(server="server2", url="dslite.server", table="CNSIM2", resource="CNSIM2r", driver="DSLiteDriver"
builder$append(server="server3", url="https://molgenis.example.org", table="CNSIM3", resource="CNSIM3r", token="123456789", driver="MolgenisDriver"
builder$append(server="server4", url="dslite.server", table="CNSIM4", resource="CNSIM4r", driver="DSLiteDriver"
logindata <- builder$build()

# or load the data.frame that contains the login details
data(logindata)
# Example 1: just login (default)
connections <- datashield.login(logins=logindata)

# Example 2: login and assign the whole dataset
connections <- datashield.login(logins=logindata, assign=TRUE)

# Example 3: login and assign specific variable(s)
myvar <- list("LAB_TSC")
connections <- datashield.login(logins=logindata, assign=TRUE, variables=myvar)

# note that the assignment information can also be provided afterwards
builder <- newDSLoginBuilder()
builder$append(server="server1", url="https://opal-demo.obiba.org",
                   user="dsuser", password="password")
builder$append(server="server2", url="https://opal-test.obiba.org",
                   token="123456789")
logindata <- builder$build()
connections <- datashield.login(logins=logindata)
datashield.assign.table(connections, symbol = "D",
                         table = list(server1 = "CNSIM.CNSIM1",
                                      server2 = "CNSIM.CNSIM2"))
datashield.assign.resource(connections, symbol = "rsrc",
                           table = list(server1 = "res.CNSIM1",
                                         server2 = "res.CNSIM2"))

## End(Not run)

datashield.logout

Logout from DataSHIELD R sessions

Description
Clear the Datashield R sessions and logout from DataSHIELD data repositories.

Usage
datashield.logout(conns, save = NULL)

Arguments

conns  DSConnection-class object or a list of DSConnection-classes.
save   Save datashield sessions on each DataSHIELD data repository (if feature is supported) with provided ID (must be a character string).
datashield.methods  List of DataSHIELD methods

Description
Get the list of all the DataSHIELD methods from the different data repositories.

Usage
datashield.methods(conns, type = "aggregate")

Arguments
- **conns**: DSConnection-class object or a list of DSConnection-classes.
- **type**: Type of the method: "aggregate" (default) or "assign".

Value
Methods details from all the servers.

datashield.method_status  Status of the DataSHIELD methods

Description
Get the status of the DataSHIELD methods in the different data repositories to check if any method is missing.

Usage
datashield.method_status(conns, type = "aggregate")

Arguments
- **conns**: DSConnection-class object or a list of DSConnection-classes.
- **type**: Type of the method: "aggregate" (default) or "assign".

Value
Methods availability on each server.
datashield.pkg_check  Check server-side package minimum version

Description
Check for each of the server, accessible through provided DSConnection-class objects, whether the installed

Usage
datashield.pkg_check(
  conns,
  name,
  version,
  env = getOption("datashield.env", globalenv())
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conns</td>
<td>DSConnection-class object or a list of DSConnection-classes.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the server-side package.</td>
</tr>
<tr>
<td>version</td>
<td>The minimum package version number to be matched.</td>
</tr>
<tr>
<td>env</td>
<td>Environment where the package status result should be cached. Try to get it from the 'datashield.env' option, with default to the Global Environment.</td>
</tr>
</tbody>
</table>

datashield.pkg_status  Status of the DataSHIELD packages

Description
Get the status of the DataSHIELD packages in the different data repositories to check if any package is missing.

Usage
datashield.pkg_status(conns)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conns</td>
<td>DSConnection-class object or a list of DSConnection-classes.</td>
</tr>
</tbody>
</table>

Value
Packages status for each server.
**datashield.resources**  
*List of the resources*

**Description**
Get the list of all the resources from the different data repositories.

**Usage**
datashield.resources(conns)

**Arguments**

- **conns**  
  *DSConnection-class* object or a list of *DSConnection-classes*.

**Value**
Resource unique names from all the servers.

**Examples**

```r
## Not run:
datashield.resources(conns)

## End(Not run)
```

---

**datashield.resource_status**  
*Status of some resources*

**Description**
Get whether some identified resources are accessible in each of the data repositories.

**Usage**
datashield.resource_status(conns, resource)

**Arguments**

- **conns**  
  *DSConnection-class* object or a list of *DSConnection-classes*.

- **resource**  
  Fully qualified name of a resource in the data repository (can be a vector or must be the same in each data repository); or a named list of fully qualified resource names (one per server name); or a data frame with 'server' and 'resource' columns (such as the one that is used in *datashield.login*).

**Value**
Resource status for each server.
**datashield.rm**  
*Remove a R symbol*

**Description**
Remove a symbol from the current Datashield session.

**Usage**
```r
datashield.rm(conns, symbol)
```

**Arguments**
- `conns` : `DSConnection-class` object or a list of `DSConnection-class`
s.
- `symbol` : Name of the R symbol.

---

**datashield.symbols**  
*List R symbols*

**Description**
Get the R symbols available after the `datashield.assign` calls in the Datashield R session.

**Usage**
```r
datashield.symbols(conns)
```

**Arguments**
- `conns` : `DSConnection-class` object or a list of `DSConnection-class`
s.

---

**datashield.tables**  
*List of the tables*

**Description**
Get the list of all the tables from the different data repositories.

**Usage**
```r
datashield.tables(conns)
```

**Arguments**
- `conns` : `DSConnection-class` object or a list of `DSConnection-class`
s.
**Value**
Table unique names from all the servers.

**Examples**
```r
## Not run:
datashield.tables(conns)
## End(Not run)
```

---

**datashield.table_status**

*Status of some tables*

**Description**
Get whether some identified tables are accessible in each of the data repositories.

**Usage**
```r
datashield.table_status(conns, table)
```

**Arguments**
- **conns**: DSConnection-class object or a list of DSConnection-classes.
- **table**: Fully qualified name of a table in the data repository (can be a vector or must be the same in each data repository); or a named list of fully qualified table names (one per server name); or a data frame with 'server' and 'table' columns (such as the one that is used in datashield.login)

**Value**
Table status for each server.

---

**datashield.workspaces**

*List saved DataSHIELD R workspaces*

**Description**
Get the list of R workspaces that were saved during a Datashield R session.

**Usage**
```r
datashield.workspaces(conns)
```

**Arguments**
- **conns**: DSConnection-class object or a list of DSConnection-classes.
datashield.workspace_rm

Remove a DataSHIELD workspace

Description

Remove in each data repository the workspace with the provided name.

Usage

datashield.workspace_rm(conns, ws)

Arguments

conns DSConnection-class object or a list of DSConnection-classes.
ws The workspace name

datashield.workspace_save

Save DataSHIELD R session to a workspace

Description

Save the current state of the DataSHIELD R session in a workspace with the provided name in each data repository. The workspace can be restored on the next datashield.login.

Usage

datashield.workspace_save(conns, ws)

Arguments

conns DSConnection-class object or a list of DSConnection-classes.
ws The workspace name
dsAggregate  Aggregate data

Description

Aggregate some data from the DataSHIELD R session using a valid R expression. The aggregation expression must satisfy the data repository’s DataSHIELD configuration.

Usage

dsAggregate(conn, expr, async = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conn</td>
<td>An object that inherits from DSConnection-class.</td>
</tr>
<tr>
<td>expr</td>
<td>Expression to evaluate.</td>
</tr>
<tr>
<td>async</td>
<td>Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.</td>
</tr>
</tbody>
</table>

See Also

Other DSConnection generics: DSConnection-class, dsAssignExpr(), dsAssignResource(), dsAssignTable(), dsDisconnect(), dsGetInfo(), dsHasResource(), dsHasTable(), dsIsAsync(), dsListMethods(), dsListPackages(), dsListResources(), dsListSymbols(), dsListTables(), dsListWorkspaces(), dsRmSymbol(), dsRmWorkspace(), dsSaveWorkspace()

Examples

```r
## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
               "username", "password", "https://opal.example.org")
dsAssignTable(con, "D", "test.CNSIM")
dsAggregate(con, as.symbol("meanDS(D$WEIGHT)"))
dsDisconnect(con)
## End(Not run)
```

---

dsAssignExpr  Assign an expression result

Description

Assign the result of the evaluation of an expression to a symbol the DataSHIELD R session. The assignment expression must satisfy the data repository’s DataSHIELD configuration.
Usage

dsAssignExpr(conn, symbol, expr, async = TRUE)

Arguments

conn        An object that inherits from \texttt{DSConnection-class}.
symbol      Name of the R symbol.
expr        A R expression with allowed assign functions calls.
async       Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.

See Also

Other DSConnection generics: \texttt{DSConnection-class}, \texttt{dsAggregate()}, \texttt{dsAssignResource()}, \texttt{dsAssignTable()}, \texttt{dsDisconnect()}, \texttt{dsGetInfo()}, \texttt{dsHasResource()}, \texttt{dsHasTable()}, \texttt{dsIsAsync()}, \texttt{dsListMethods()}, \texttt{dslistPackages()}, \texttt{dsListResources()}, \texttt{dsListSymbols()}, \texttt{dsListTables()}, \texttt{dsListWorkspaces()}, \texttt{dsRmSymbol()}, \texttt{dsRmWorkspace()}, \texttt{dsSaveWorkspace()}

Examples

```r
# Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
                 "username", "password", "https://opal.example.org")
dAssignExpr(con, "C", as.symbol("c(1, 2, 3)"))
dDisconnect(con)

# End(Not run)
```

Description

Assign a resource object of class 'ResourceClient' from the data repository to a symbol in the DataSHIELD R session. The resource reference to be assigned must exist (i.e. proper permissions apply) for the DataSHIELD user.

Usage

dsAssignResource(conn, symbol, resource, async = TRUE)
dsAssignTable

Assign a data table

Description
Assign a data table from the data repository to a symbol in the DataSHIELD R session. The table to be assigned must exist (i.e. proper permissions apply) for the DataSHIELD user.

Usage

```r
dsAssignTable(
    conn, symbol, table, variables = NULL, missings = FALSE, identifiers = NULL, id.name = NULL, async = TRUE
)
```

Arguments

- **conn**: An object that inherits from `DSConnection-class`
- **symbol**: Name of the R symbol.
- **resource**: Fully qualified name of a resource reference in the data repository.
- **async**: Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.

See Also
Other DSConnection generics: `DSConnection-class`, `dsAggregate()`, `dsAssignExpr()`, `dsAssignTable()`, `dsDisconnect()`, `dsGetInfo()`, `dsHasResource()`, `dsHasTable()`, `dsIsAsync()`, `dsListMethods()`, `dsListPackages()`, `dsListResources()`, `dsListSymbols()`, `dsListTables()`, `dsListWorkspaces()`, `dsRmSymbol()`, `dsRmWorkspace()`, `dsSaveWorkspace()`

Examples
```r
## Not run:
con <- dbConnect(DSOpal::Opal(), "server1", "username", "password", "https://opal.example.org")
dsAssignResource(con, "D", "test.CNSIM")
dsDisconnect(con)
## End(Not run)
```
Arguments

- **conn**: An object that inherits from `DSConnection-class`.
- **symbol**: Name of the R symbol.
- **table**: Fully qualified name of a table in the data repository.
- **missings**: If TRUE, missing values will be pushed from data repository to R, default is FALSE.
- **identifiers**: Name of the identifiers mapping to use when assigning entities to R (if supported by the data repository).
- **id.name**: Name of the column that will contain the entity identifiers. If not specified, the identifiers will be the data frame row names. When specified this column can be used to perform joins between data frames.
- **async**: Whether the result of the call should be retrieved asynchronously. When TRUE (default) the calls are parallelized over the connections, when the connection supports that feature, with an extra overhead of requests.

See Also

Other DSConnection generics: `DSConnection-class`, `dsAggregate()`, `dsAssignExpr()`, `dsAssignResource()`, `dsDisconnect()`, `dsGetInfo()`, `dsHasResource()`, `dsHasTable()`, `dsIsAsync()`, `dsListMethods()`, `dsListPackages()`, `dsListResources()`, `dsListSymbols()`, `dsListTables()`, `dsListWorkspaces()`, `dsRmSymbol()`, `dsRmWorkspace()`, `dsSaveWorkspace()`

Examples

```r
## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
                 "username", "password", "https://opal.example.org")
dsAssignTable(con, "D", "test.CNSIM")
dsDisconnect(con)
## End(Not run)
```

Description

Connect to a data repository going through the appropriate authentication procedure. Some implementations may allow you to have multiple connections open, so you may invoke this function repeatedly assigning its output to different objects. The authentication mechanism is left unspecified, so check the documentation of individual drivers for details.
Usage

dsConnect(drv, name, restore = NULL, ...)

Arguments

`drv`  an object that inherits from `DSDriver-class`.
`name`  Name of the connection, which must be unique among all the DataSHIELD connections.
`restore`  Workspace name to be restored in the newly created DataSHIELD R session.

...  authentication arguments needed by the data repository instance; these typically include ‘username’, ‘password’, ‘token’, ‘host’, ‘port’, ‘dbname’, etc. For details see the appropriate ‘DSDriver’.

See Also

dsDisconnect to disconnect from a data repository.

Other DSDriver generics: `DSDriver-class`, `dsGetInfo()`

Examples

```r
## Not run:
con <- dsConnect(DSOpal::Opal(), "server1", "username", "password", "https://opal.example.org")
con
dsListTables(con)
dsDisconnect(con)

## End(Not run)
```

DSConnection-class  

**DSConnection class**

Description

This virtual class encapsulates the connection to a DataSHIELD-aware data repository, and it provides access to data assignments and aggregations etc.

Implementation note

Individual drivers are free to implement single or multiple simultaneous connections.

See Also

Other DS classes: `DSDriver-class`, `DSObject-class`, `DSResult-class`

Other DSConnection generics: `dsAggregate()`, `dsAssignExpr()`, `dsAssignResource()`, `dsAssignTable()`, `dsDisconnect()`, `dsGetInfo()`, `dsHasResource()`, `dsHasTable()`, `dsIsAsync()`, `dsListMethods()`, `dsListPackages()`, `dsListResources()`, `dsListSymbols()`, `dsListTables()`, `dsListWorkspaces()`, `dsRmSymbol()`, `dsRmWorkspace()`, `dsSaveWorkspace()`
Examples

## Not run:
con <- dsConnect(DSOpal::Opal(), "server1", "username", "password", "https://opal.example.org")
con
dsDisconnect(con)

## End(Not run)

---

### dsDisconnect

**Disconnect (close) a connection**

#### Description

This closes the connection, discards all pending work, and frees resources (e.g., memory, sockets).

#### Usage

```r
dsDisconnect(conn, save = NULL)
```

#### Arguments

- `conn`: An object inheriting from `DSConnection-class`.
- `save`: Save DataSHIELD session in data repository with provided identifier string.

#### See Also

Other DSConnection generics: `DSConnection-class`, `dsAggregate()`, `dsAssignExpr()`, `dsAssignResource()`, `dsAssignTable()`, `dsGetInfo()`, `dsHasResource()`, `dsHasTable()`, `dsIsAsync()`, `dsListMethods()`, `dsListPackages()`, `dsListResources()`, `dsListSymbols()`, `dsListTables()`, `dsListWorkspaces()`, `dsRmSymbol()`, `dsRmWorkspace()`, `dsSaveWorkspace()`

#### Examples

## Not run:
con <- dbConnect(DSOpal::Opal(), "server1", 
                 "username", "password", "https://opal.example.org")
dsDisconnect(con)

## End(Not run)
**DSDriver-class**  
**DSDriver class**

**Description**
Base class for all DataSHIELD-aware data repositories drivers (e.g., Opal, ...). The virtual class ‘DSDriver’ defines the operations for creating connections.

**See Also**
Other DS classes: **DSConnection-class, DSObject-class, DSResult-class**  
Other DSDriver generics: **dsConnect(), dsGetInfo()**

---

**dsFetch**  
*Get the raw result*

**Description**
Wait for the result to be available and fetch the result from a previous assignment or aggregation operation that may have been run asynchronously, in which case it is a one-shot call. When the assignment or aggregation operation was not asynchronous, the result is wrapped in the object and can be fetched multiple times.

**Usage**

```r
dsFetch(res)
```

**Arguments**

- **res**  
  An object inheriting from **DSResult-class**.

**See Also**
Other DSResult generics: **DSResult-class, dsGetInfo()**

**Examples**
```r
## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
  "username", "password", "https://opal.example.org")
dsAssignExpr(con, "C", as.symbol("c(1, 2, 3)"))
res <- dsAggregate(con, as.symbol("length(C)"))
length <- dsFetch(res)
dsDisconnect(con)

## End(Not run)
```
dsGetInfo

Get DataSHIELD-aware data repository metadata

Description

Get DataSHIELD-aware data repository metadata

Usage

dsGetInfo(dsObj, ...)

Arguments

dsObj  An object inheriting from DSObject-class, i.e. DSDriver-class, DSConnection-class, or a DSResult-class.

... Other arguments to methods.

Value

a named list

Implementation notes

For ‘DSDriver’ subclasses, this should include the version of the package (‘driver.version’) and the version of the underlying client library (‘client.version’).

For ‘DSConnection’ objects this should report the version of the data repository application (‘repo.version’) and its name (‘repo.name’), the database name (‘dbname’), username, (‘username’), host (‘host’), port (‘port’), etc. It MAY also include any other arguments related to the connection (e.g., thread id, socket or TCP connection type). It MUST NOT include the password.

For ‘DSResult’ objects, this should include the R expression being executed (‘expression’) and if the query is complete (‘has.completed’).

See Also

Other DSDriver generics: DSDriver-class, dsConnect()

Other DSConnection generics: DSConnection-class, dsAggregate(), dsAssignExpr(), dsAssignResource(), dsAssignTable(), dsDisconnect(), dsHasResource(), dsHasTable(), dsIsAsync(), dsListMethods(), dsListPackages(), dsListResources(), dsListSymbols(), dsListTables(), dsListWorkspaces(), dsRmSymbol(), dsRmWorkspace(), dsSaveWorkspace()

Other DSResult generics: DSResult-class, dsFetch()
dsHasResource  

*Check remote resource exists*

**Description**

Check if a remote resource reference exists in the data repository. Returns a logical indicating the existence of a remote resource accessible through this connection.

**Usage**

```r
dsHasResource(conn, resource)
```

**Arguments**

- `conn`: An object that inherits from `DSConnection-class`.
- `resource`: the resource fully qualified name

**See Also**

Other DSConnection generics: `DSConnection-class, dsAggregate(), dsAssignExpr(), dsAssignResource(), dsAssignTable(), dsDisconnect(), dsGetInfo(), dsHasTable(), dsIsAsync(), dsListMethods(), dsListPackages(), dsListResources(), dsListSymbols(), dsListTables(), dsListWorkspaces(), dsRmSymbol(), dsRmWorkspace(), dsSaveWorkspace()`

**Examples**

```r
## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
                 "username", "password", "https://opal.example.org")
dsHasResource(con, "test.CNSIM")
dsDisconnect(con)
## End(Not run)
```

dsHasTable  

*Check remote table exists*

**Description**

Check if a remote table exists in the data repository. Returns a logical indicating the existence of a remote table accessible through this connection.

**Usage**

```r
dsHasTable(conn, table)
```
Arguments

conn    An object that inherits from `DSConnection-class`.
table    the table fully qualified name

See Also

Other DSConnection generics: `DSConnection-class`, `dsAggregate()`, `dsAssignExpr()`, `dsAssignResource()`, `dsAssignTable()`, `dsDisconnect()`, `dsGetInfo()`, `dsHasResource()`, `dsIsAsync()`, `dsListMethods()`, `dsListPackages()`, `dsListResources()`, `dsListSymbols()`, `dsListTables()`, `dsListWorkspaces()`, `dsRmSymbol()`, `dsRmWorkspace()`, `dsSaveWorkspace()`

Examples

```r
## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
                  "username", "password", "https://opal.example.org")
dsHasTable(con, "test.CNSIM")
dsDisconnect(con)
## End(Not run)
```

dIsAsync  

Asynchronous result support

Description

When a `DSResult-class` object is returned on aggregation or assignment operation, the raw result can be accessed asynchronously, allowing parallelization of DataSHIELD calls over multiple servers. The returned named list of logics will specify if asynchronicity is supported for: aggregation operation (`aggregate`), table assignment operation (`assignTable`), resource assignment operation (`assignResource`) and expression assignment operation (`assignExpr`).

Usage

`dsIsAsync(conn)`

Arguments

`conn`    An object that inherits from `DSConnection-class`.

See Also

Other DSConnection generics: `DSConnection-class`, `dsAggregate()`, `dsAssignExpr()`, `dsAssignResource()`, `dsAssignTable()`, `dsDisconnect()`, `dsGetInfo()`, `dsHasResource()`, `dsHasTable()`, `dsListMethods()`, `dsListPackages()`, `dsListResources()`, `dsListSymbols()`, `dsListTables()`, `dsListWorkspaces()`, `dsRmSymbol()`, `dsRmWorkspace()`, `dsSaveWorkspace()`
dsListMethods

Get the DataSHIELD methods

description

Get the list of DataSHIELD methods that have been configured on the remote data repository.

Usage

dsListMethods(conn, type = "aggregate")

Arguments

- conn: An object that inherits from `DSConnection-class`
- type: Type of the method: "aggregate" (default) or "assign".

Value

A data.frame with columns: name, type ('aggregate' or 'assign'), class ('function' or 'script'), value, package, version.

See Also

Other DSConnection generics: `DSConnection-class`, `dsAggregate()`, `dsAssignExpr()`, `dsAssignResource()`, `dsAssignTable()`, `dsDisconnect()`, `dsGetInfo()`, `dsHasResource()`, `dsHasTable()`, `dsIsAsync()`, `dsListPackages()`, `dsListResources()`, `dsListSymbols()`, `dsListTables()`, `dsListWorkspaces()`, `dsRmSymbol()`, `dsRmWorkspace()`, `dsSaveWorkspace()`

Examples

```r
## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
                 "username", "password", "https://opal.example.org")
dsIsAsync(con)
dsDisconnect(con)

## End(Not run)
```

```r
## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
                 "username", "password", "https://opal.example.org")
dsListMethods(con)
dsDisconnect(con)

## End(Not run)
```
dsListPackages

Get the DataSHIELD packages

Description
Get the list of DataSHIELD packages with their version, that have been configured on the remote data repository.

Usage

dsListPackages(conn)

Arguments

conn An object that inherits from DSConnection-class.

Value
A data.frame with columns: name, version.

See Also
Other DSConnection generics: DSConnection-class, dsAggregate(), dsAssignExpr(), dsAssignResource(), dsAssignTable(), dsDisconnect(), dsGetInfo(), dsHasResource(), dsHasTable(), dsIsAsync(), dsListMethods(), dsListResources(), dsListSymbols(), dsListTables(), dsListWorkspaces(), dsRmSymbol(), dsRmWorkspace(), dsSaveWorkspace()

Examples

## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
    "username", "password", "https://opal.example.org")
dsListPackages(con)
dsDisconnect(con)

## End(Not run)

dsListResources

List remote resources

Description
List remote resources from the data repository. Returns the unquoted names of remote resources accessible through this connection.

Usage

dsListResources(conn)
dsListSymbols

List symbols

Description

After assignments have been performed, some symbols live in the DataSHIELD R session on the server side.

Usage

dsListSymbols(conn)

Arguments

conn An object that inherits from DSConnection-class.

See Also

Other DSConnection generics: DSConnection-class, dsAggregate(), dsAssignExpr(), dsAssignResource(), dsAssignTable(), dsDisconnect(), dsGetInfo(), dsHasResource(), dsHasTable(), dsIsAsync(), dsListMethods(), dsListPackages(), dsListSymbols(), dsListTables(), dsListWorkspaces(), dsRmSymbol(), dsRmWorkspace(), dsSaveWorkspace()
dsListTables

Examples

## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
    "username", "password", "https://opal.example.org")
dsAssignTable(con, "D", "test.CNSIM")
dsListSymbols(con)
dsDisconnect(con)

## End(Not run)

dListTables List remote tables

Description

List remote tables from the data repository. Returns the unquoted names of remote tables accessible through this connection.

Usage

dListTables(conn)

Arguments

conn An object that inherits from DSConnection-class.

See Also

Other DSConnection generics: DSConnection-class, dsAggregate(), dsAssignExpr(), dsAssignResource(), dsAssignTable(), dsDisconnect(), dsGetInfo(), dsHasResource(), dsHasTable(), dsIsAsync(), dsListMethods(), dsListPackages(), dsListResources(), dsListSymbols(), dsListWorkspaces(), dsRmSymbol(), dsRmWorkspace(), dsSaveWorkspace()

Examples

## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
    "username", "password", "https://opal.example.org")
dListTables(con)
dsDisconnect(con)

## End(Not run)
dsListWorkspaces  
*Get the DataSHIELD workspaces*

Description
Get the list of DataSHIELD workspaces, that have been saved on the remote data repository.

Usage
```r
dsListWorkspaces(conn)
```

Arguments
- `conn`: An object that inherits from `DSConnection-class`.

Value
A data.frame with columns: name, lastAccessDate, size.

See Also
Other DSConnection generics: `DSConnection-class, dsAggregate(), dsAssignExpr(), dsAssignResource(), dsAssignTable(), dsDisconnect(), dsGetInfo(), dsHasResource(), dsHasTable(), dsIsAsync(), dsListMethods(), dsListPackages(), dsListResources(), dsListSymbols(), dsListTables(), dsRmSymbol(), dsRmWorkspace(), dsSaveWorkspace()

Examples
```r
## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
    "username", "password", "https://opal.example.org")
dsListWorkspaces(con)
dsDisconnect(con)

## End(Not run)
```

---

DSLoginBuilder  
*DataSHIELD login details builder*

Description
DataSHIELD login details builder
DataSHIELD login details builder

Format
A R6 object of class DSLoginBuilder
DSLoginBuilder

Details

Helper class for creating a valid data frame that can be used to perform `datashield.login`. See also `newDSLoginBuilder`.

Methods

Public methods:

- `DSLoginBuilder$new()`
- `DSLoginBuilder$append()`
- `DSLoginBuilder$build()`
- `DSLoginBuilder$clone()`

Method `new()`: Create a new DSLoginBuilder instance.

Usage:

```r
DSLoginBuilder$new(logins = NULL, .silent = FALSE)
```

Arguments:

- `logins` A valid login details data frame to initiate the builder, optional.
- `.silent` Do not warn user when non secure HTTP urls are encountered. Default is FALSE.

Returns: A DSLoginBuilder object.

Method `append()`: Append login information for a specific server.

Usage:

```r
DSLoginBuilder$append(
  server,
  url,
  table = "",
  resource = "",
  driver = "OpalDriver",
  user = "",
  password = "",
  token = "",
  options = ""
)
```

Arguments:

- `server` The server name (must be unique).
- `url` The url to connect to the server or a R symbol name.
- `table` The table path that identifies the dataset in the server.
- `resource` The resource path that identifies the resource reference in the server.
- `driver` The `DSDriver-class` name to build the `DSConnection-class`.
- `user` The user name in the user credentials.
- `password` The user password in the user credentials.
- `token` The personal access token (ignored when user credentials are not empty).
options Any options (R code to be parsed) that could be relevant for the DS connection object.

Method `build()`: Build the DSLoginBuilder instance.

Usage:
DSLoginBuilder$build()

Returns: The DataSHIELD logindata data.frame

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

Usage:
DSLoginBuilder$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

---

DSObject-class

**DSObject class**

Description

Base class for all other DataSHIELD classes (e.g., drivers, connections). This is a virtual Class: No objects may be created from it.

Details

More generally, DataSHIELD defines a very small set of classes and generics that allows users and applications perform meta-analysis with a common interface. The virtual classes are ‘DSDriver’ that individual drivers extend, ‘DSConnection’ that represent instances of DataSHIELD-aware data repository connections, and ‘DSResult’ that represent the result of a DataSHIELD operation. These three classes extend the basic class of ‘DSObject’, which serves as the root or parent of the class hierarchy.

Implementation notes

An implementation MUST provide methods for the following generics:

- dsGetInfo

It MAY also provide methods for:

- summary Print a concise description of the object. The default method invokes ‘dsGetInfo(dsObj)’ and prints the name-value pairs one per line. Individual implementations may tailor this appropriately.

See Also

Other DS classes: DSConnection-class, DSDriver-class, DSResult-class
Examples

```r
## Not run:
drv <- DSOpal::Opal()
con <- dsConnect(drv, "username", "password", "https://opal.example.org")

rs <- dsAssign(con, "Project.TableA")
is(drv, "DSObject") ## True
is(con, "DSObject") ## True
is(rs, "DSObject")  ## True

dsDisconnect(con)

## End(Not run)
```

---

**DSResult-class**  
*DSResult class*

**Description**

This virtual class describes the result and state of execution of a DataSHIELD request (aggregation or assignment).

**Implementation notes**

Individual drivers are free to allow single or multiple active results per connection. The default show method displays a summary of the query using other DS generics.

**See Also**

Other DS classes: **`DSConnection-class`, `DSDriver-class`, `DSObject-class`**  
Other DSResult generics: **`dsFetch()`, `dsGetInfo()`**

---

**dsRmSymbol**  
*Remove a symbol*

**Description**

After removal, the data identified by the symbol will not be accessible in the DataSHIELD R session on the server side.

**Usage**

dsRmSymbol(conn, symbol)
Arguments

conn An object that inherits from DSConnection-class.
symbol Name of the R symbol.

See Also

Other DSConnection generics: DSConnection-class, dsAggregate(), dsAssignExpr(), dsAssignResource(), dsAssignTable(), dsDisconnect(), dsGetInfo(), dsHasResource(), dsHasTable(), dsIsAsync(), dsListMethods(), dsListPackages(), dsListResources(), dsListSymbols(), dsListTables(), dsListWorkspaces(), dsRmWorkspace(), dsSaveWorkspace()

Examples

## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
  "username", "password", "https://opal.example.org")
dsAssignTable(con, "D", "test.CNSIM")
dsRmSymbol(con, "D")
dsDisconnect(con)

## End(Not run)

dsRmWorkspace Remove a DataSHIELD workspace

Description

Remove a DataSHIELD workspace from the remote data repository. Ignore if no such workspace exists.

Usage

dsRmWorkspace(conn, name)

Arguments

conn An object that inherits from DSConnection-class.
name Name of the workspace

See Also

Other DSConnection generics: DSConnection-class, dsAggregate(), dsAssignExpr(), dsAssignResource(), dsAssignTable(), dsDisconnect(), dsGetInfo(), dsHasResource(), dsHasTable(), dsIsAsync(), dsListMethods(), dsListPackages(), dsListResources(), dsListSymbols(), dsListTables(), dsListWorkspaces(), dsRmSymbol(), dsSaveWorkspace()
dsSaveWorkspace

Save the DataSHIELD R session in a workspace

Description

Save the DataSHIELD R session in a workspace on the remote data repository.

Usage

dsSaveWorkspace(conn, name)

Arguments

conn An object that inherits from DSConnection-class.
name Name of the workspace

See Also

Other DSConnection generics: DSConnection-class, dsAggregate(), dsAssignExpr(), dsAssignResource(),
    dsAssignTable(), dsDisconnect(), dsGetInfo(), dsHasResource(), dsHasTable(), dsIsAsync(),
    dsListMethods(), dsListPackages(), dsListResources(), dsListSymbols(), dsListTables(),
    dsListWorkspaces(), dsRmSymbol(), dsRmWorkspace()

Examples

## Not run:
con <- dbConnect(DSOpal::Opal(), "server1",
    "username", "password", "https://opal.example.org")
dsSaveWorkspace(con, "foo")
dsListWorkspaces(con)
dsRmWorkspace(con, "foo")
dsListWorkspaces(con)
dsDisconnect(con)

## End(Not run)
newDSLoginBuilder Create a new DataSHIELD login details builder

Description

Shortcut function to create a new DSLoginBuilder instance. The data frame that is being built can be used to perform datashield.login.

Usage

newDSLoginBuilder(logins = NULL, .silent = FALSE)

Arguments

logins A valid login details data frame to initiate the builder, optional.
.silent Do not warn user when non secure HTTP urls are encountered. Default is FALSE.

Examples

{
  builder <- newDSLoginBuilder()
  builder$append(server="server1", url="https://opal-demo.obiba.org", table="datashield.CNSIM1", user="administrator", password="password")
  builder$append(server="server2", url="dslite.server", table="CNSIM2")
  builder$append(server="server3", url="http://molgenis.example.org", table="CNSIM3", token="123456789")
  builder$append(server="server4", url="dslite.server", table="CNSIM4")
  logindata <- builder$build()
}
Index

datashield.aggregate, 3
datashield.assign, 4
datashield.assign.expr, 5
datashield.assign.resource, 6
datashield.assign.table, 4, 7
datashield.connections, 8, 8, 10
datashield.connections_default, 8, 9, 10
datashield.connections_find, 9, 10, 10
datashield.errors, 11
datashield.login, 6, 7, 11, 16, 18, 19, 35, 40
datashield.logout, 13
datashield.method.status, 14
datashield.methods, 14
datashield.pkg_check, 15
datashield.pkg_status, 15
datashield.resource_status, 16
datashield.resources, 16
datashield.rm, 17
datashield.symbols, 17
datashield.table_status, 18
datashield.tables, 17
datashield.workspace.rm, 19
datashield.workspace_save, 19
datashield.workspaces, 18
dsAggregate, 20, 21–25, 27–34, 38, 39
dsAssignExpr, 20, 20, 22–25, 27–34, 38, 39
dsAssignResource, 20, 21, 21, 23–25, 27–34, 38, 39
dsAssignTable, 20–22, 22, 24, 25, 27–34, 38, 39
dsConnect, 23, 26, 27
DSConnection-class, 24
dsDisconnect, 20–24, 25, 27–34, 38, 39
DSDriver-class, 26
dsFetch, 26, 27, 37
dsGetInfo, 20–26, 27, 28–34, 36–39
dsHasResource, 20–25, 27, 28, 29–34, 38, 39
dsHasTable, 20–25, 27, 28, 29–34, 38, 39
dsIsAsync, 20–25, 27–29, 29, 30–34, 38, 39
dsListMethods, 20–25, 27–29, 30, 31–34, 38, 39
dsListPackages, 20–25, 27–30, 31, 32–34, 38, 39
dsListSymbols, 20–25, 27–32, 32, 33, 34, 38, 39
dsListTables, 20–25, 27–32, 33, 34, 38, 39
dsListWorkspaces, 20–25, 27–33, 34, 38, 39
DSLoginBuilder, 34, 40
DSObject-class, 36
DSResult-class, 37
dsRmSymbol, 20–25, 27–34, 37, 38, 39
dsRmWorkspace, 20–25, 27–34, 38, 39
dsSaveWorkspace, 20–25, 27–34, 38, 39
newDSLoginBuilder, 35, 40
summary, 36