class MongoDB::Client

Class to define connections to servers

Table of Contents


```
package MongoDB { class Client { ... } }
```

Synopsis

```
my MongoDB::Client $client .= new(:uri<mongodb://>);
if $client.nbr-servers {
  my MongoDB::Database $d1 = $client.database('my_db1');
  my MongoDB::Collection $c1 = $d1.collection('my_cll1');
  my MongoDB::Collection $c2 = $client.collection('my_db2.my_cll2');
}
```

Description

This class is your most often used class. It maintains the connection to the servers specified in the given uri. In the background it herds a set of MongoDB::Server objects.

Readonly attributes

read-concern

```
has BSON::Document $.read-concern;
```

The read-concern is a structure to have some control over the read operations to which server the operations are directed to. Default is an empty structure. The structure will be explained elsewhere.

found-master

```
has Bool $.found-master = False;
```

While the client is processing the given uri it will set this flag when a master server is detected.

Methods

new

```
submethod BUILD ( Str:D :$uri, BSON::Document :$read-concern, Int )
```

Create a MongoDB::Client object.

Note. It is important to keep the following in mind to prevent memory leakage. The object must be cleaned up by hand before the variable is reused. This is because the Client object creates some background processes to keep an eye on the server and to update server object states and topology.

```
my MongoDB::Client $c .= new( ... );
# ... work with object
$c.cleanup;
```

Some help is given by the object creation. When it notices that the object (self) is defined along with some internal variables, it will destroy that object first before continuing. This also means that you must not use another MongoDB::Client object to create a new one!

```
my MongoDB::Client $c1, $c2;

# first time use, no leakage
$c1 .= new(...);

# In this proces $c1 will be destroyed!!
$c2 = $c1.new(...);

# This is ok however because we want to overwrite the object anyway
$c2 .= new(...);

# And this will result in memory leakage because $c2 was already defined.
$c2 = MongoDB::Client.new(...);
```

read-concern

Read concern will overwrite the default concern.

uri

Uri defines the servers and control options. The string is like a normal uri with mongodb as a protocol name. The difference however lies in the fact that more that one server can be defined. The uri definition states that at least a servername must be stated in the uri. Here in this package the absence of any name defaults to localhost. See also the MongoDB page to look for options and definition.

'Uri examples'

Example uri	Explanation
mongodb://	most simple specification, localhost using port 27017
mongodb://:65000	localhost on port 65000
mongodb://:56,:876	two servers localhost on port 56 and 876
mongodb://example.com	server example.com on port 27017
mongodb://pete:mypasswd@	server localhost:27017 on which pete must login using mypasswd
mongodb://pete:mypasswd@/mydb	same as above but login on database mydb
mongodb:///?replicaSet=myreplset	localhost:27017 must belong to a replica set named myreplset
mongodb://u1:pw1@nsa.us:666,my.datacenter.gov/nsa/?replicaSet=foryoureyesonly	User u1 with password pw1 logging in on database nsa on server nsa.us:666 and my.datacenter.gov:27017 which must both be member of a replica set named foryoureyesonly.

Note that the servers named in the uri must have something in common such as a replica set. Servers are refused when there is some problem between them e.g. both are master servers. In such situations another MongoDB::Client object should be created for the other server.

The options which can be used in the uri are in the following tables. See also this information for more details.

Section	Impl	Use
Replica set options		
replicaSet	done	Specifies the name of the replica set, if the mongod is a member of a replica set.
Connection options		
ssl		0 or 1. 1 Initiate the connection with TLS/SSL. The default value is false.
connectTimeoutMS		The time in milliseconds to attempt a connection before timing out.
socketTimeoutMS		The time in milliseconds to attempt a send or receive on a socket before the attempt times out.
Connect pool options		
maxPoolSize		The maximum number of connections in the connection pool. The default value is 100.
minPoolSize		The minimum number of connections in the connection pool. The default value is 0.
maxIdleTimeMS		The maximum number of milliseconds that a connection can remain idle in the pool before being removed and closed.
waitQueueMultiple		A number that the driver multiples the maxPoolSize value to, to provide the maximum number of threads allowed to wait for a connection to become available from the pool.
waitQueueTimeoutMS		The maximum time in milliseconds that a thread can wait for a connection to become available. For default values, see the MongoDB Drivers and Client Libraries documentation.
Write concern options		
w		Corresponds to the write concern w Option. The w option requests acknowledgement that the write operation has propagated to a specified number of mongod instances or to mongod instances with specified tags. You can specify a number, the string majority, or a tag set.
wtimeoutMS		Corresponds to the write concern wtimeout. wtimeoutMS specifies a time limit, in milliseconds, for the write concern. When wtimeoutMS is 0, write operations will never time out.

Section	Impl	Use
journal		Corresponds to the write concern j Option option. The journal option requests acknowledgement from MongoDB that the write operation has been written to the journal
Read concern options		
readConcernLevel		The level of isolation. Accepts either "local" or "majority".
Read preference option		
readPreference		Specifies the replica set read preference for this connection. The read preference values are the following: primary, primaryPreferred, secondary, secondaryPreferred, nearest
readPreferenceTags		Specifies a tag set as a comma-separated list of colon-separated key-value pairs
Authentication options		
authSource	part	Specify the database name associated with the user credentials, if the users collection do not exist in the database where the client is connecting. authSource defaults to the database specified in the connection string.
authMechanism		Specify the authentication mechanism that MongoDB will use to authenticate the connection. Possible values include: SCRAM-SHA-1, MONGODB-CR, MONGODB-X509, GSSAPI (Kerberos), PLAIN (LDAP SASL)
gssapiServiceName		Set the Kerberos service name when connecting to Kerberized MongoDB instances. This value must match the service name set on MongoDB instances.

nbr-servers

```
method nbr-servers ( --> Int )
```

Return number of servers found processing the uri in new(). When called directly after new() it may not have the proper count yet caused by delays in processing especially when processing replicasets.

server-status

```
method server-status ( Str:D $server-name --> ServerStatus )
```

Return the status of some server. The defined values are shown in the table and when it applies.

Server state	When
SS-Mongos	Field 'msg' in returned resuld of ismaster request is 'isdbgrid'.
SS-RSGhost	Field 'isreplicaset' is set. Server is in a initialization state.
SS-RSPrimary	Replicaset primary server. Field 'setName' is the replicaset name and 'ismaster' is True.
SS-RSSecondary	Replicaset secondary server. Field 'setName' is the replicaset name and 'secondary' is True.
SS-RSArbiter	Replicaset arbiter. Field 'setName' is the replicaset name and 'arbiterOnly' is True.
SS-RSOther	An other type of replicaserver is found. Possibly in transition between states.
SS-Standalone	Any other server being master or slave.
SS-Unknown	Servers which are down or with errors.
SS-PossiblePrimary	not implemeted

client-topology

```
method client-topology ( --> TopologyType ) {
```

Return the topology of the set of servers represents. A table of types is shown next;

Topology type	When
TT-Single	The first server with no faulty responses will set the topology to single. Any new SS-Standalone server will flip the topology to TT-Unknown
TT-ReplicaSetNoPrimary	When there are no primary servers found (yet) in a group of replicaservers, the topology is one of replicaset without a primary. When only one server is provided in the uri, the topology would first be TT-Single. Then the Client will gather more data from the server to find the primary and or other secondary servers. The topology might then change into this topology or the TT-ReplicaSetWithPrimary described below.
TT-ReplicaSetWithPrimary	When in a group of replica servers a primary is found, this topology is selected.
TT-Sharded	When mongos servers are provided in the uri, this topology applies. When there is only one server, the type would become TT-Single.
TT-Unknown	Any set of servers which are SS-Unknown will set the topology to TT-Unknown. Depending on the problems of these servers their states can change, and with that, the topology. When there is a set of servers which are not mixable, the topology becomes also TT-Unknown. Examples are more than one standalone server, mongos and replica servers, replicaservers from different replica sets etc.

select-server

```
multi method select-server ( Str:D :$servername! --> MongoDB::Server )

multi method select-server (
  BSON::Document :$read-concern is copy
  --> MongoDB::Server
)
```

The first method tries to get a specific server while the second is running through a selection mechanism using the server state and client topology.

Select a server for operations. It returns a Server object. In single server setups it is always the server you want to have. When however selecting a server from a replicaset the server is selected according to several rules such as read-concern, operation type (read or write) and round trip time to the server. When read-concern is not defined, the data is taken from this Clients read-concern. **Note**, this method is used internally and most of the time of no concern to the user.

database

```
method database (
  Str:D $name, BSON::Document :$read-concern
  --> MongoDB::Database
)
```

Create a Database object. In mongodb a database and its collections are only created when data is written in a collection.

The read-concern when defined will overide the one of the Client. If not defined, the structure of the client is taken.

collection

```
method collection (
  Str:D $full-collection-name, BSON::Document :$read-concern
  --> MongoDB::Collection
)
```

A shortcut to define a database and collection at once. The names for the database and collection are given in the string full-collection-name. This is a string of two names separated by a dot '.'.

When the read-concern is defined it overides the one from Client. If not defined, the structure of the client is taken.

cleanup

method cleanup ( )

Stop any background work on the Server object as well as the Monitor object. Cleanup structures so the object can be cleaned further by the GC later.