Creating a Client

On this page

Using `Mongo::Client`

To start a Ruby driver connection, create a `Mongo::Client` object. Provide a list of hosts and options or a connection URI to the `Mongo::Client` constructor. The client's oselected database defaults to `admin`.

To create a client to a standalone server, provide one host in the seed list. Optionally, you can force the cluster topology to be standalone without going through the auto-discovery steps.

```
Mongo::Client.new([ '127.0.0.1:27017' ], :database => 'mydb')
Mongo::Client.new([ '127.0.0.1:27017' ], :database => 'mydb', :connect => :direct)
Mongo::Client.new('mongodb://127.0.0.1:27017/mydb')
```

To connect to a replica set, pass one or more hosts and the replica set name. The driver's auto-discovery feature finds all members of the replica set if they are not all provided.

```
Mongo::Client.new([ '127.0.0.1:27017', '127.0.0.1:27018' ], :database => 'mydb', replica_set:
Mongo::Client.new('mongodb://127.0.0.1:27017,127.0.0.1:27018/mydb?replicaSet=myapp')
```

To create a client to a sharded cluster, pass one or more mongos hosts. The auto-discovery feature can determine that the servers are `mongos` instances, but if you would like to bypass the auto-discovery, pass the `sharded` option to the client.

```
Mongo::Client.new([ '127.0.0.1:27017' ], :database => 'mydb')
Mongo::Client.new([ '127.0.0.1:27017' ], :database => 'mydb', :connect => :sharded)
Mongo::Client.new('mongodb://127.0.0.1:27017/mydb?connect=sharded')
```

Client Options

A number of different options can be passed to a `Mongo::Client` to configure driver behavior, either by providing them in the options hash to the constructor or by providing them in the URI.

Since the URI options are required in camel case, which is not the Ruby standard, the following table shows the option in the URI and its corresponding option if passed to the constructor in Ruby.

> **NOTE:**
> The options passed directly should be symbols.

The options are explained in detail in the Connection URI reference.

> **NOTE:**
> Options that are set in **milliseconds** in the URI are represented as a `float` in Ruby and the units are **seconds**.

URI Options Conversions

URI Option	Ruby Option
replicaSet=String	`:replica_set => String`
connect=String	`:connect => Symbol`
ssl=Boolean	`:ssl => true|false`
connectTimeoutMS=Integer	`:connect_timeout => Float`
socketTimeoutMS=Integer	`:socket_timeout => Float`

URI Option	Ruby Option
serverSelectionTimeoutMS=Integer	`:server_selection_timeout => Float`
localThresholdMS=Integer	`:local_threshold => Float`
maxPoolSize=Integer	`:max_pool_size => Integer`
minPoolSize=Integer	`:min_pool_size => Integer`
waitQueueTimeoutMS=Integer	`:wait_queue_timeout => Float`
w=Integer|String	`{ :write => { :w => Integer|String }}`
wtimeoutMS=Integer	`{ :write => { :wtimeout => Float }}`
journal=Boolean	`{ :write => { :j => true|false }}`
fsync=Boolean	`{ :write => { :fsync => true|false }}`
readPreference=String	`{ :read => { :mode => Symbol }}`
readPreferenceTags=Strings	`{ :read => { :tag_sets => Array<String> }}`
authSource=String	`:auth_source => String`
authMechanism=String	`:auth_mech => Symbol`
authMechanismProperties=Strings	`{ :auth_mech_properties => { :service_realm => String, :canonicalize_host_name => true|false, :service_name => String } }`

Ruby Options

Option	Description	Type	Default
`:replica_set`	When connecting to a replica set, this is the name of the set to filter servers by.	`String`	none
`:ssl`	Tell the client to connect to the servers via SSL.	`Boolean`	false

Option	Description	Type	Default
`:ssl_cert`	The certificate file path used to identify the connection against MongoDB. This option, if present, takes precedence over the values of :ssl_cert_string and :ssl_cert_object.	`String`	none
`:ssl_cert_string`	A string containing the PEM-encoded certificate used to identify the connection against MongoDB. This option, if present, takes precedence over the value of :ssl_cert_object.	`String`	none
`:ssl_cert_object`	The OpenSSL::X509::Certificate used to identify the connection against MongoDB.	`OpenSSL::X509::Certificate`	none
`:ssl_key`	The private keyfile used to identify the connection against MongoDB. Note that even if the key is stored in the same file as the certificate, both need to be explicitly specified. This option, if present, takes precedence over the values of :ssl_key_string and :ssl_key_object.	`String`	none
`:ssl_key_string`	A string containing the PEM-encoded private key used to identify the connection against MongoDB. This parameter, if present, takes precedence over the value of option :ssl_key_object.	`String`	none
`:ssl_key_object`	The private key used to identify the connection against MongoDB.	`OpenSSL::PKey`	none
`:ssl_key_pass_phrase`	A passphrase for the private key.	`String`	none

Option	Description	Type	Default
`:ssl_ca_cert`	The file path containing a set of concatenated certification authority certifications used to validate certs passed from the other end of the connection. One of :ssl_ca_cert, :ssl_ca_cert_string or :ssl_ca_cert_object (in order of priority) is required for :ssl_verify.	`String`	none
`:ssl_ca_cert_string`	A string containing a set of concatenated certification authority certifications used to validate certs passed from the other end of the connection. One of :ssl_ca_cert, :ssl_ca_cert_string or :ssl_ca_cert_object (in order of priority) is required for :ssl_verify.	`String`	none
`:ssl_ca_cert_object`	An array of OpenSSL::X509::Certificate representing the certification authority certifications used to validate certs passed from the other end of the connection. One of :ssl_ca_cert, :ssl_ca_cert_string or :ssl_ca_cert_object (in order of priority) is required for :ssl_verify.	`Array<OpenSSL::X509::Certificate>`	none
`:ssl_verify`	Whether or not to do peer certification validation.	`Boolean`	false
`:connect_timeout`	The number of seconds to wait to establish a socket connection before raising an exception.	`Float`	10 seconds
`:socket_timeout`	The number of seconds to wait for an operation to execute on a socket before raising an exception.	`Float`	5 seconds

Option	Description	Type	Default
`:max_pool_size`	The maximum size of the connection pool for each server.	`Integer`	5
`:min_pool_size`	The minimum number of connections in the connection pool for each server.	`Integer`	1
`:wait_queue_timeout`	The number of seconds to wait for a connection in the connection pool to become available.	`Float`	1
`:write`	Specifies write concern options as a `Hash`. Keys in the hash can be `:w`, `:wtimeout`, `:j`, `:fsync`. `{ :write => { :w => 2 } }`	`Hash`	`{ :w => 1 }`

`:read`	Specifies the read preference mode and tag sets for selecting servers as a `Hash`. Keys in the hash are `:mode` and `:tag_sets`.
	
	`{ :read =>`
	`  { :mode => :secondary,`
	`  :tag_sets => [ "berlin" ]`
	`  }`
	`}`
		`Hash`	`{ :mode => :primary }`

`:auth_source`	Specifies the authentication source.	`String`	For MongoDB 2.6 and later: **admin** if credentials are supplied, otherwise the current database

Option	Description	Type	Default
`:auth_mech`	Specifies the authenticaion mechanism to use. Can be one of: `:mongodb_cr`, `:mongodb_x509`, `:plain`, `:scram`.	`Symbol`	MongoDB 3.0 and later: `:scram` if user credentials are supplied but an `:auth_mech` is not. 2.6 and earlier: `:mongodb_cr`
`:auth_mech_properties`	Provides additional authentication mechanism properties.	`Hash`	none
`:user`	The name of the user to authenticate with.	`String`	none
`:password`	The password of the user to authenticate with.	`String`	none
`:connect`	Overrides the auto-discovery feature of the driver and forces the cluster topology to a specific type. Choices: `:direct`, `:replica_set` or `:sharded`.	`Symbol`	none
`:heartbeat_frequency`	The number of seconds for the server monitors to refresh server states asynchronously.	`Float`	10
`:database`	The name of the database to connect to.	`String`	admin
`:server_selection_timeout`	The number of seconds to wait for an appropriate server to be selected for an operation to be executed before raising an exception.	`Float`	30
`:local_threshold`	Specifies the maximum latency in seconds between the nearest server and the servers that can be available for selection to operate on.	`Float`	0.015

Details on Timeout Options

`connect_timeout`

On initialization of a connection to a server, this setting is the number of seconds to wait to connect before raising an exception. This timeout is also used when monitor threads ping their servers. The default is 10 seconds. See the socket timeout for monitoring specification for further explanation.

`socket_timeout`

The number of seconds to wait for an operation to execute on a socket before raising a timeout exception. It should take into account network latency and operation duration. The default is no value; the default is effectively infinity. Please consider using `max_time_ms` per-operation instead, as the `socket_timeout` does not stop the operation on the server; a long-running operation will continue to run on the server, beyond a socket timeout being reached. See the socket timeout for monitoring specification documentation for further information relating to server discovery and monitoring.

`server_selection_timeout`

The number of seconds to wait for the driver to find an appropriate server to which an operation can be sent before raising an exception. Defaults to 30. It should take the speed of elections during a failover into account. See the serverSelectionTimeoutMS specification for further information.

`local_threshold`

The maximum latency in seconds between the nearest server and the servers that can be considered available to send an operation to. Defaults to 0.015.

> NOTE:
> This is not the latency window between the driver and a server, but rather the latency between the nearest server and other servers. See the localThresholdMS specification.

`wait_queue_timeout`

The number of seconds to wait for a connection in the connection pool to become available. You should consider increasing this number if you are seeing many `Timeout` errors while using many threads or when operations are long-running. Defaults to 1 second.

`max_pool_size`

Maximum size of the connection pool for each server. Defaults to 5 connections.

`min_pool_size`

Minimum number of connections in the connection pool for each server. Increase this number to create connections when the pool is initialized and to reduce the overhead of creating new connections later on. Defaults to 1.

`max_time_ms`

Specified as an option on a particular operation. It defines a cumulative time limit in milliseconds for processing operations on a cursor. Consider using this option instead of a `socket_timeout`, if the operation should be interrupted on the server. See the CRUD specification for details on operations that support this option.