Package ‘RcppRedis’

March 8, 2022

Type    Package
Title   'Rcpp' Bindings for 'Redis' using the 'hiredis' Library
Version 0.2.0
Date    2022-03-08
Author  Dirk Eddelbuettel and Bryan W. Lewis
Maintainer Dirk Eddelbuettel <edd@debian.org>
Description Connection to the 'Redis' key/value store using the
   C-language client library 'hiredis' (included as a fallback) with
   'MsgPack' encoding provided via 'RcppMsgPack' headers. It now also
   includes the pub/sub functions from the 'redis' package.
SystemRequirements An available hiredis library (eg via package
   libhiredis-dev on Debian/Ubuntu, hiredis-devel on
   Fedora/RedHat, or directly from source from
   <https://github.com/redis/hiredis>) can be used but version
   1.0.2 is also included and built on demand if needed.
URL https://github.com/eddelbuettel/rcppredis,
   https://dirk.eddelbuettel.com/code/rcpp.redis.html
BugReports https://github.com/eddelbuettel/rcppredis/issues
License GPL (>= 2)
Imports methods, Rcpp (>= 0.11.0), RApiSerialize
LinkingTo Rcpp, RApiSerialize
Suggests RcppMsgPack, rredis, tinytest
NeedsCompilation yes
Repository CRAN
Date/Publication 2022-03-08 20:40:02 UTC

R topics documented:

  RcppRedis .................................................. 2
  redisMonitorChannels ..................................... 2
Redis Monitor Channels

Description

Listen for messages on subscribed Redis message channels.

Usage

redisMonitorChannels(context, type=c("rdata", "raw", "string"))

Arguments

context A valid Redis context (see example).

type The expected message value type.
Details

(From the Redis.io documentation): implement the Publish/Subscribe messaging paradigm where (citing Wikipedia) senders (publishers) are not programmed to send their messages to specific receivers (subscribers). Rather, published messages are characterized into channels, without knowledge of what (if any) subscribers there may be. Subscribers express interest in one or more channels, and only receive messages that are of interest, without knowledge of what (if any) publishers there are.

The `redisMonitorChannels` function may be called repeatedly in an event loop to service messages on all subscribed channels. When a message is received, the `redisMonitorChannels` function will attempt to evaluate a callback function with same name as the channel, with the message as its single argument. If no such function can be found, the message is returned. See the help page for `redisGetResponse` for a description of the message format.

WARNING: The `redisMonitorChannels` function blocks indefinitely until a message is received. Use the lower-level `listen` context method to simply poll channels for messages without evaluating function callbacks.

Value

The result of an evaluated function callback message, or if no matching callback exists, the message.

Author(s)

B. W. Lewis

References

http://redis.io/commands

Examples

```r
## Not run:
x <- new(Redis)
y <- new(Redis)

# Define a callback function to process messages from channel 1:
channel1 <- function(x) {
cat("Message received from channel 1: ",x,"\n")
}
# Define a callback function to process messages from channel 2:
channel2 <- function(x) {
cat("Message received from channel 2: ",x,"\n")
}

# Subscribe to the channels...
x$subscribe(c("channel1", "channel2"))
y$subscribe("channel2")
y$publish("channel2", pi)
redisMonitorChannels(x)
```
# Unsubscribe
x$unsubscribe(c("channel1", "channel2"))

## End(Not run)
Index

* package
  RcppRedis, 2

  Rcpp_Redis (RcppRedis), 2
  Rcpp_Redis-class (RcppRedis), 2
  RcppRedis, 2
  Redis (RcppRedis), 2
  redisMonitorChannels, 2

  serializeToChar (RcppRedis), 2
  serializeToRaw (RcppRedis), 2

  unserializeFromChar (RcppRedis), 2
  unserializeFromRaw (RcppRedis), 2