

Introduction

A golang interface to the CZMQ v4.2 API.

Install

Dependencies

- libsodium
- libzmq
- czmq

For CZMQ master

```
1 go get github.com/zeromq/goczmq
```

A Note on Build Tags The CZMQ library includes experimental classes that are not built by default, but can be built by passing `--enable-drafts` to configure. Support for these draft classes are being added to goczmq. To build these features against a CZMQ that has been compiled with `--enable-drafts`, use `go build -tags draft`.

For CMZQ = 4.2

```
1 go get gopkg.in/zeromq/goczmq.v4
```

Note: CZMQ 4.2 is has not been released yet.

For CZMQ Before 4.0

```
1 go get gopkg.in/zeromq/goczmq.v1
```

Usage

Direct CZMQ Sock API

Example

```
1 package main
2
3 import (
4     "log"
5
6     "github.com/zeromq/goczmq"
7 )
8
9 func main() {
10     // Create a router socket and bind it to port 5555.
11     router, err := goczmq.NewRouter("tcp://*:5555")
12     if err != nil {
13         log.Fatal(err)
14     }
15     defer router.Destroy()
16
17     log.Println("router created and bound")
18
19     // Create a dealer socket and connect it to the router.
20     dealer, err := goczmq.NewDealer("tcp://127.0.0.1:5555")
21     if err != nil {
22         log.Fatal(err)
23     }
24     defer dealer.Destroy()
25
26     log.Println("dealer created and connected")
27
28     // Send a 'Hello' message from the dealer to the router.
29     // Here we send it as a frame ([]byte), with a FlagNone
30     // flag to indicate there are no more frames following.
31     err = dealer.SendFrame([]byte("Hello"), goczmq.FlagNone)
32     if err != nil {
33         log.Fatal(err)
34     }
35
36     log.Println("dealer sent 'Hello'")
37
38     // Receive the message. Here we call RecvMessage, which
39     // will return the message as a slice of frames ([][]byte).
40     // Since this is a router socket that support async
41     // request / reply, the first frame of the message will
42     // be the routing frame.
43     request, err := router.RecvMessage()
44     if err != nil {
45         log.Fatal(err)
46     }
```

```

47
48     log.Printf("router received '%s' from '%v'", request[1], request
49               [0])
50     // Send a reply. First we send the routing frame, which
51     // lets the dealer know which client to send the message.
52     // The FlagMore flag tells the router there will be more
53     // frames in this message.
54     err = router.SendFrame(request[0], goczmq.FlagMore)
55     if err != nil {
56         log.Fatal(err)
57     }
58
59     log.Printf("router sent 'World'")
60
61     // Next send the reply. The FlagNone flag tells the router
62     // that this is the last frame of the message.
63     err = router.SendFrame([]byte("World"), goczmq.FlagNone)
64     if err != nil {
65         log.Fatal(err)
66     }
67
68     // Receive the reply.
69     reply, err := dealer.RecvMessage()
70     if err != nil {
71         log.Fatal(err)
72     }
73
74     log.Printf("dealer received '%s'", string(reply[0]))
75 }

```

Output

```

1 2015/05/26 21:52:52 router created and bound
2 2015/05/26 21:52:52 dealer created and connected
3 2015/05/26 21:52:52 dealer sent 'Hello'
4 2015/05/26 21:52:52 router received 'Hello' from '[0 103 84 189 175]'
5 2015/05/26 21:52:52 router sent 'World'
6 2015/05/26 21:52:52 dealer received 'World'

```

io.ReadWriter support

Example

```

1 package main
2
3 import (
4     "log"
5
6     "github.com/zeromq/goczmq"
7 )

```

```

8
9 func main() {
10     // Create a router socket and bind it to port 5555.
11     router, err := gocmq.NewRouter("tcp://*:5555")
12     if err != nil {
13         log.Fatal(err)
14     }
15     defer router.Destroy()
16
17     log.Println("router created and bound")
18
19     // Create a dealer socket and connect it to the router.
20     dealer, err := gocmq.NewDealer("tcp://127.0.0.1:5555")
21     if err != nil {
22         log.Fatal(err)
23     }
24     defer dealer.Destroy()
25
26     log.Println("dealer created and connected")
27
28     // Send a 'Hello' message from the dealer to the router,
29     // using the io.Write interface
30     n, err := dealer.Write([]byte("Hello"))
31     if err != nil {
32         log.Fatal(err)
33     }
34
35     log.Printf("dealer sent %d byte message 'Hello'\n", n)
36
37     // Make a byte slice and pass it to the router
38     // Read interface. When using the ReadWriter
39     // interface with a router socket, the router
40     // caches the routing frames internally in a
41     // FIFO and uses them transparently when
42     // sending replies.
43     buf := make([]byte, 16386)
44
45     n, err = router.Read(buf)
46     if err != nil {
47         log.Fatal(err)
48     }
49
50     log.Printf("router received '%s'\n", buf[:n])
51
52     // Send a reply.
53     n, err = router.Write([]byte("World"))
54     if err != nil {
55         log.Fatal(err)
56     }
57
58     log.Printf("router sent %d byte message 'World'\n", n)

```

```

59
60     // Receive the reply, reusing the previous buffer.
61     n, err = dealer.Read(buf)
62     if err != nil {
63         log.Fatal(err)
64     }
65
66     log.Printf("dealer received '%s'", string(buf[:n]))
67 }

```

Output

```

1 2015/05/26 21:54:10 router created and bound
2 2015/05/26 21:54:10 dealer created and connected
3 2015/05/26 21:54:10 dealer sent 5 byte message 'Hello'
4 2015/05/26 21:54:10 router received 'Hello'
5 2015/05/26 21:54:10 router sent 5 byte message 'World'
6 2015/05/26 21:54:10 dealer received 'World'

```

Thread safe channel interface

Example

```

1 package main
2
3 import (
4     "log"
5
6     "github.com/zeromq/goczmq"
7 )
8
9 func main() {
10     // Create a router channeler and bind it to port 5555.
11     // A channeler provides a thread safe channel interface
12     // to a *Sock
13     router := goczmq.NewRouterChanneler("tcp://*:5555")
14     defer router.Destroy()
15
16     log.Println("router created and bound")
17
18     // Create a dealer channeler and connect it to the router.
19     dealer := goczmq.NewDealerChanneler("tcp://127.0.0.1:5555")
20     defer dealer.Destroy()
21
22     log.Println("dealer created and connected")
23
24     // Send a 'Hello' message from the dealer to the router.
25     dealer.SendChan <- [][]byte{[]byte("Hello")}
26     log.Println("dealer sent 'Hello'")
27
28     // Receive the message as a [][]byte. Since this is

```

```
29 // a router, the first frame of the message wil
30 // be the routing frame.
31 request := <-router.RecvChan
32 log.Printf("router received '%s' from '%v'", request[1], request
    [0])
33
34 // Send a reply. First we send the routing frame, which
35 // lets the dealer know which client to send the message.
36 router.SendChan <- [][]byte{request[0], []byte("World")}
37 log.Printf("router sent 'World'")
38
39 // Receive the reply.
40 reply := <-dealer.RecvChan
41 log.Printf("dealer received '%s'", string(reply[0]))
42 }
```

Output

```
1 2015/05/26 21:56:43 router created and bound
2 2015/05/26 21:56:43 dealer created and connected
3 2015/05/26 21:56:43 dealer sent 'Hello'
4 2015/05/26 21:56:43 received 'Hello' from '[0 12 109 153 35]'
5 2015/05/26 21:56:43 router sent 'World'
6 2015/05/26 21:56:43 dealer received 'World'
```

GoDoc

godoc

See Also

- Peter Kleiweg's zmq4 bindings

License

This project uses the MPL v2 license, see LICENSE