总线系统接口说明

1 IConnect

```
/*
 * 总线连接接口，用于向总线注册一个新的连接
 * @param url: 总线的url地址
 * @return *Conn: 连接的实例，用于调用其他方法
 * @return error: 错误类型
 */
func IConnect(url string) (*Conn, error)
```

- e.g. 使用场景

  ```
  nc, err := IConnect("nats://127.0.0.1:4222")
  ```

2 IClose

```
/*
 * 关闭连接接口，用于关闭已经注册的连接
 * @class *Conn: 连接的实例
 */
func (nc *Conn) IClose()
```

- e.g. 使用场景

  ```
  nc, err := IConnect("nats://127.0.0.1:4222")
  defer nc.IClose()
  ```

3 IPublish

```
/*
 * 消息发送接口，用于向总线服务器发送一条消息
 * @class *Conn: 连接的实例
 * @param subj: 消息发送的主题，接收消息也在该主题上接收
 * @param data: 消息的文本
 * @return error: 错误类型
 */
func (nc *Conn) IPublish(subj string, data []byte) error
```

- e.g. 使用场景

```
err := nc.IPublish("subj1", []byte("hello world"))
```

4 ISubscribe

```
/*
 * 消息接收接口，用于接收总线服务器中的消息
 * @class *Conn: 连接的实例
 * @param subj: 消息接收的主题，发送消息在其上发送
 * @param cb: 消息处理函数，其中定义了对得到消息结构体的处理方式
 * @return Subscription: 接收状态的描述结构体
 * @return error: 错误类型
 */
func (nc *Conn) ISubscribe(subj string, cb MsgHandler) (*Subscription, error)
```

- e.g. 使用场景

```
// 接收消息并打印
sub, err = nc.ISubscribe("subj1", func(m *Msg) {
    fmt.Printf("Received a message: %s\n", string(m.Data))
})
```

```
// 接收消息请求并回复
sub, err := nc.ISubscribe("subj1", func(m *Msg) {
    m.IRespond([]byte("reply! "))
    fmt.Printf("request: %v\n", string(m.Data))
})
```

5 IRequest

```
/*
 * 消息请求接口，用于向另一个使用者发起消息请求
 * @class *Conn: 连接的实例
 * @param subj: 消息请求的主题
 * @param data: 消息的文本
 * @param timeout: 请求的等待时限
 * @return []byte: 接收返回的消息
 * @return error: 错误类型
 */
func (nc *Conn) IRequest(subj string, data []byte, timeout time.Duration) ([]byte,
error)
```

- e.g. 使用场景

```
data, err := nc.IRequest("subj1", []byte("request"), 5*time.Second)
```

6 IRespond

```
/*
 * 消息请求接口，用于向另一个使用者发起消息请求
 * @class *Msg: 消息结构体
 * @param data: 回复消息的文本
 * @return error: 错误类型
 */
func (m *Msg) IRespond(data []byte) error
```

- e.g. 使用场景

```
sub, err := nc.ISubscribe("subj1", func(m *Msg) {
    m.IRespond([]byte("reply! "))
    fmt.Printf("request: %v\n", string(m.Data))
})
```