**订阅服务**

# 简述

以太坊的订阅:以太坊中，如果A要获取B的订阅事件，需要拿到B的实例化对象，但是这时候如果B也要获取A的实例化对象，就会造成包循环依赖问题，造成代码的耦合度更高

改进版：为了解决代码循环依赖，也方便后期的消息队列的改善，订阅方和提供方仅关心订阅事件的消息码（string）不关心内在处理

使用场景：订阅方和提供方都会使用，订阅方到该服务订阅事件，服务方将消息发到该服务

# 模块设计

## 数据结构定义

type subevent struct {

    handleFeed map[**string**]\*event.Feed *//map key为strig，value为Feed事件*

    scope event.SubscriptionScope *//*

    listEvent []**string** *//订阅事件消息码集合*

}

* 全局变量的初始化函数

函数描述：var localSubEvent \*subevent = Run() 主要用途是初始化一个全局的指针

函数信息：

* 函数名：func Run() \*subevent {}
* 函数输入：无
* 函数输出：subevent的实例化对象

详细设计：handleFeed的初始化，listEvent事件的初始化,遍历listEvent数组对map进行赋值，返回初始化后的实例对象

func Run() \*subevent {

    subevent := &subevent{handleFeed: make(map[**string**]\*event.Feed)}

    subevent.listEvent = []**string**{ *//添加各种消息事件的消息码*

        "CA\_NewID",

    }

    for \_, v := range subevent.listEvent {

        subevent.handleFeed[v] = new(event.Feed)

    }

    return subevent

}

* 订阅事件函数

函数描述：订阅方调用该函数，将消息码和通道注册进来，内部Feed再进行注册

函数信息：

* 函数名：func SubscribeEvent(aim string, ch interface{}) (event.Subscription, error) {}
* 函数输入：消息码字符串，通道
* 函数输出:sub事件（取消订阅用），错误信息

详细设计：先判断map[aim]是否为空，为空报异常，正常则用Feed订阅通道（以太坊event实现）

func SubscribeEvent(aim **string**, ch interface{}) (event.Subscription, **error**) {

    \_, ok := localSubEvent.handleFeed[aim]

    if ok == false {

        return nil, SubscribeEventFailedNoThisEvent

    }

    Feed := localSubEvent.handleFeed[aim]

    return localSubEvent.scope.Track(Feed.Subscribe(ch)), nil

}

* 发送数据函数

函数描述：提供方调用该函数，将消息码和数据注册进来，内部Feed再进行发送

函数信息

* 函数名：func PostEvent(aim string, data interface{}) error {
* 函数输入：消息码，数据
* 函数输出：错误信息

详细设计：先判断map[aim]是否为空，为空报异常，正常则用Feed发送数据（以太坊event实现）

func PostEvent(aim **string**, data interface{}) **error** {

    \_, ok := localSubEvent.handleFeed[aim]

    if ok == false {

        return PostEventFailedNoThisEvent

    }

    Feed := localSubEvent.handleFeed[aim]

    Feed.Send(data)

    return nil

}

# 注意事项

1. 消息码定义规则：采用包名+消息，例如CA模块要提供身份到来消息，则可用CA\_NewID来表示
2. 订阅方和提供方定义好实用规则：双方实用前约定消息码，消息结构，不能订阅错误（代码中自行检查）
3. 接口的验证：网络组与算法组都准备好各自接口，11号下午对接口
4. 消息码分开维护问题：网络组和算法组维护不同的消息码集合，放入同一文件夹不同文件下
5. 若有人恶意改代码调用其他模块的Feed乱发消息怎么办：经讨论这版先内部规范好订阅规则，外部恶意改代码问题，如有更好建议可以给出