recv data compatible

一、环境配置

本项目所需要的第三方插件在docker环境中启动。

项目基础运行环境;

- jdk 1.8
- springboot 2.5.13
- netty 4.1.73.Final
- fastjson 2.0.9
- maven 3.8.8

> docker安装：docker在本地安装，可自行在网络上下载最新版本进行安装。
> 项目运行中所依赖的第三方插件环境需要在docker中运行起来。

> rabbitmq安装

```
1.拉取镜像。
docker pull rabbitmq
# 创建容器
docker run -di --name rabbitmq -p 4369:4369 -p 5671:5671 -p 5672:5672 -p
15671:15671 -p 15672:15672 -p 25672:25672 rabbitmq
2.进入容器并开启管理功能。
# 进入容器
docker exec -it rabbitmq /bin/bash
# 开启 RabbitMQ 管理功能
rabbitmq-plugins enable rabbitmq_management
    访问：http://192.168.10.10:15672/ 使用 guest 登录账号密码拉取镜像。
3.访问：http://192.168.10.10:15672/ 使用 guest 登录账号密码
```

> redis安装

```
1.拉取镜像。
docker pull redis
2.创建容器。
docker run -di --name redis -p 6379:6379 redis
```

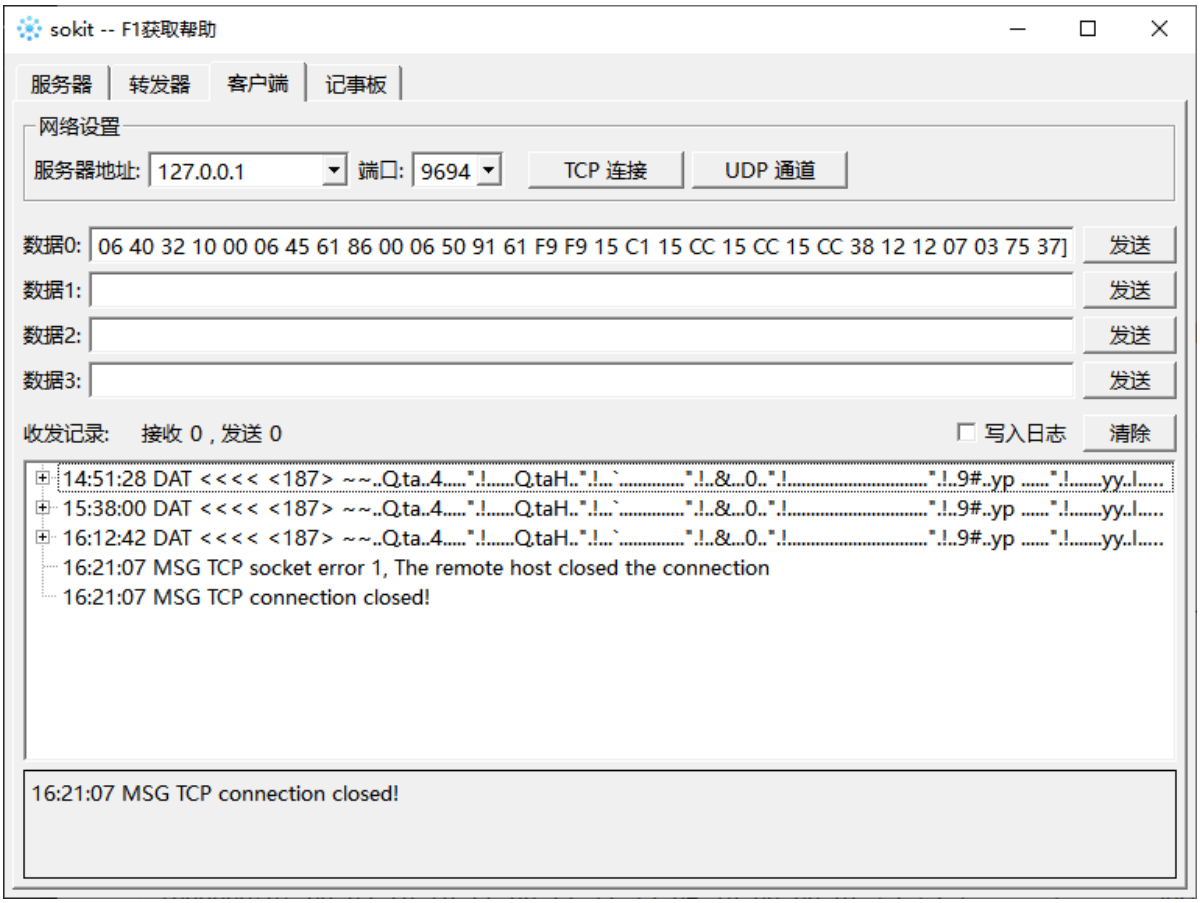
> mongodb安装(3.11.0)

```
1.拉取镜像。
docker pull mongo
2.创建容器。
docker run -di --name mongo -p 27017:27017 mongo
```

二、运行流程

接受数据->解析数据->存储数据
1.往消息队列里面放入数据
2.平台监听rabbitmq队列里面的数据，并获取数据，并进行解析
3.解析完后，将数据按指定格式存入数据库

使用sokit.exe等发包工具与数据解析后端项目建立TCP连接，并向服务器端发送待解析的报文。如下图所示：



测试使用的报文为RTU报文，如下所示：

```
[7E 7E 01 00 51 10 74 61 FF FF 34 00 AA 02 01 F3 22 06 21 12 01 10 F1 F1 00 51 10
74 61 48 F0 F0 22 06 21 11 05 F4 60 00 00 05 00 00 00 00 00 00 00 00 00 F0 F0 22
06 21 12 00 26 19 00 00 30 F0 F0 22 06 21 11 05 F5 C0 02 D4 02 D4 03 1D 03 1D 03
1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D F0 F0 22 06 21 12 00 39 23 00 00 79
70 20 19 00 00 05 F0 F0 22 06 21 11 15 04 18 00 00 0F 79 79 02 11 49 80 94 02 11
90 16 56 02 11 90 16 09 02 11 90 16 09 7D 7D 00 06 35 02 35 00 06 40 32 10 00 06
45 61 86 00 06 50 91 61 F9 F9 15 C1 15 CC 15 CC 15 CC 38 12 12 07 03 75 37]
```

通过服务端解析后得到的信息是：

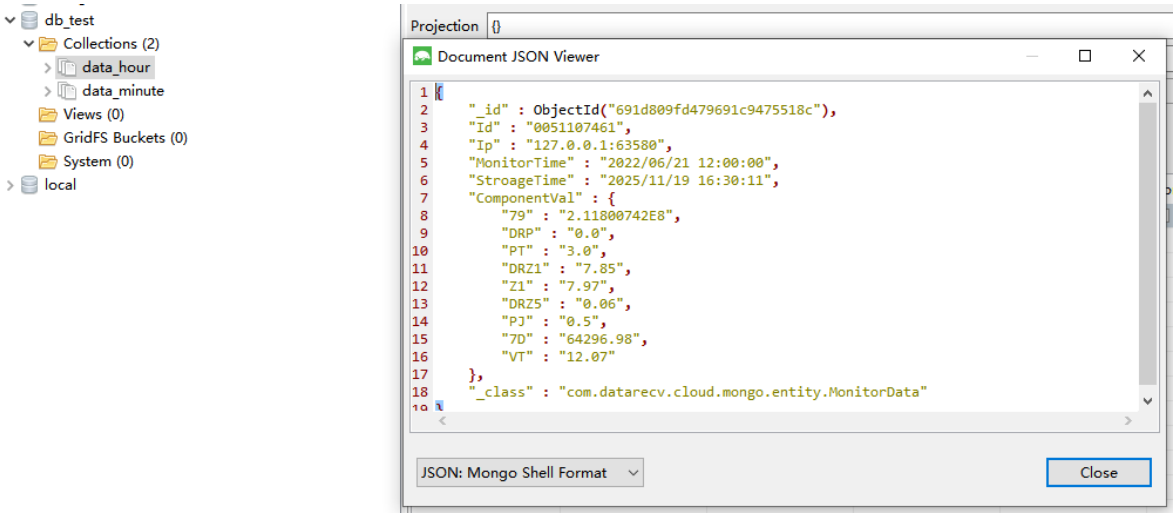
```
2025-11-19 14:39:17.795  INFO 15868 --- [ntLoopGroup-3-1]
c.d.c.s.decode.DelimiterFrameDecoder     : 报文:          +-----------------------
-------------------------+
         |  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f |
+--------+-------------------------------------------------+----------------+
|00000000| 7e 7e 01 00 51 10 74 61 ff ff 34 00 aa 02 01 f3 |~~..Q.ta..4.....|
|00000010| 22 06 21 12 01 10 f1 f1 00 51 10 74 61 48 f0 f0 |".!......Q.taH..|
|00000020| 22 06 21 11 05 f4 60 00 00 05 00 00 00 00 00 00 |".!...`.........|
|00000030| 00 00 00 f0 f0 22 06 21 12 00 26 19 00 00 30 f0 |.....".!..&...0.|
|00000040| f0 22 06 21 11 05 f5 c0 02 d4 02 d4 03 1d 03 1d |.".!............|
|00000050| 03 1d 03 1d 03 1d 03 1d 03 1d 03 1d 03 1d 03 1d |................|
|00000060| f0 f0 22 06 21 12 00 39 23 00 00 79 70 20 19 00 |..".!..9#..yp ..|
|00000070| 00 05 f0 f0 22 06 21 11 15 04 18 00 00 0f 79 79 |....".!.......yy|
|00000080| 02 11 49 80 94 02 11 90 16 56 02 11 90 16 09 02 |..I......V......|
|00000090| 11 90 16 09 7d 7d 00 06 35 02 35 00 06 40 32 10 |....}}..5.5..@2.|
|000000a0| 00 06 45 61 86 00 06 50 91 61 f9 f9 15 c1 15 cc |..Ea...P.a......|
|000000b0| 15 cc 15 cc 38 12 12 07 03 75 37                |....8....u7     |
+--------+-------------------------------------------------+----------------+
2025-11-19 14:39:17.797  INFO 15868 --- [ntLoopGroup-3-1]
c.d.c.s.decode.DelimiterFrameDecoder     : >>>in:a82744a7>剩余未读的包len:187>>包
len:187>>>>
2025-11-19 14:39:44.195  INFO 15868 --- [ntLoopGroup-3-1]
c.d.cloud.server.decode.ProtocolWrapper  : >>>>>len:187>>>>
2025-11-19 14:39:49.681  INFO 15868 --- [ntLoopGroup-3-1]
c.d.cloud.server.decode.ProtocolWrapper  : 收到的报文>>>>>>7e 7e 01 00 51 10 74 61
ff ff 34 00 aa 02 01 f3 22 06 21 12 01 10 f1 f1 00 51 10 74 61 48 f0 f0 22 06 21
11 05 f4 60 00 00 05 00 00 00 00 00 00 00 00 00 f0 f0 22 06 21 12 00 26 19 00 00
30 f0 f0 22 06 21 11 05 f5 c0 02 d4 02 d4 03 1d 03 1d 03 1d 03 1d 03 1d 03 1d 03
1d 03 1d 03 1d 03 1d f0 f0 22 06 21 12 00 39 23 00 00 79 70 20 19 00 00 05 f0 f0
22 06 21 11 15 04 18 00 00 0f 79 79 02 11 49 80 94 02 11 90 16 56 02 11 90 16 09
02 11 90 16 09 7d 7d 00 06 35 02 35 00 06 40 32 10 00 06 45 61 86 00 06 50 91 61
f9 f9 15 c1 15 cc 15 cc 15 cc 38 12 12 07 03 75 37
2025-11-19 14:39:51.106  INFO 15868 --- [ntLoopGroup-3-1]
c.datarecv.cloud.rabbit.MessageProducer  : 发送消息开始
2025-11-19 14:39:51.114  INFO 15868 --- [ntLoopGroup-3-1]
c.datarecv.cloud.rabbit.MessageProducer  : send message is:
{"checkCode":"7537","clientId":"0051107461","contentMessageRequest":
{"commonFactor":{"38":"12.07"},"elementResultS":
[{"f0":"2206211105","step":1,"stepFlag":false,"value":{"F4":
["0","0","5","0","0","0","0","0","0","0","0","0"]}},
{"f0":"220621120O","step":1,"stepFlag":false,"value":{"26":["3.0"]}},
{"f0":"2206211105","step":1,"stepFlag":false,"value":{"F5":
["724","724","797","797","797","797","797","797","797","797","797","797"]}},
{"f0":"220621120O","step":1,"stepFlag":false,"value":{"39":["7.97"],"2O":
["0.5"]}},{"f0":"2206211115","step":15,"stepFlag":true,"value":{"79":
["211498094","211901656","211901609","211901609"],"7D":
["6350235","6403210","6456186","6509161"],"F9":
["5.569","5.58","5.58","5.58"]}}],"sendingTime":"220621120110","serialNumber":"01
f3","stationClassificationCode":"48","stationCode":"0051107461"},"end":"03","ip":
"/127.0.0.1:63580","messageId":0,"messageRequest":
{"centerAddress":"01","functionCode":"34","identificationAndLength":"00aa","lengt
h":170,"password":"ffff","startCharacter":"02","telemetryStationAddress":"0051107
461","upAndDownIdentification":0},"serialNo":0}
2025-11-19 14:39:51.115  INFO 15868 --- [ntLoopGroup-3-1]
c.datarecv.cloud.rabbit.MessageProducer  : 发送消息结束
```

经过通过服务端解析后会发送到rabbitmq相应的队列中，这里rtu解析出的报文数据会被放在队列`cloud_in_RTU_SL_561`。存储的数据格式为：


```
{"checkCode":"7537","clientId":"0051107461","contentMessageRequest":
{"commonFactor":{"38":"12.07"},"elementResultS":
[{"f0":"2206211105","step":1,"stepFlag":false,"value":{"F4":
["0","0","5","0","0","0","0","0","0","0","0","0"]}},
{"f0":"2206211200","step":1,"stepFlag":false,"value":{"26":["3.0"]}},
{"f0":"2206211105","step":1,"stepFlag":false,"value":{"F5":
["724","724","797","797","797","797","797","797","797","797","797","797"]}},
{"f0":"2206211200","step":1,"stepFlag":false,"value":{"39":["7.97"],"20":
["0.5"]}},{"f0":"2206211115","step":15,"stepFlag":true,"value":{"79":
["211498094","211901656","211901609","211901609"],"7D":
["6350235","6403210","6456186","6509161"],"F9":
["5.569","5.58","5.58","5.58"]}}],"sendingTime":"220621120110","serialNumber":"01
f3","stationClassificationCode":"48","stationCode":"0051107461"},"end":"03","ip":
"/127.0.0.1:63580","messageId":0,"messageRequest":
{"centerAddress":"01","functionCode":"34","identificationAndLength":"00aa","lengt
h":170,"password":"ffff","startCharacter":"02","telemetryStationAddress":"0051107
461","upAndDownIdentification":0},"serialNo":0}
```


项目运行时的日志输入如下：

通过data-consumer服务，可以监控rabbit相关队列中是否有数据，并进行消费，将rabbit中的数据转换为数据库中需要存储的数据。如下：



三、项目代码模块

```
  <modules>
    <module>rtu-recv</module>
    <module>dtu-recv</module>
    <module>entry-recv</module>
    <module>comm-recv</module>
    <module>data-consumer</module>
    <module>data-alarm</module>
  </modules>
```

项目包含六个模块。

- comm-recv: 协议解析的一些基础功能，包括分隔符处理，消息解析或下发的基本格式；
- rtu-recv: rtu协议的具体解析方法和数据的封装方法；
- dtu-recv: dtu协议的具体解析方法和数据的封装方法；
- entry-recv: 项目使用netty接收数据的主模块；
- data-consumer: 从rabbitmq中取出解析数据进行处理，并存储到数据库；
- data-alarm: 对接收到数据进行实时逻辑判断，做出报警处理。

四、协议类型配置

框架在TCP写一下监听同一端口，通过接收的报文的头和相应的报文分隔符来区分不同类型报文，具体的配置如下：

```
##-------------------------------------------------------------------------##
protocol.delimiter[0].value=\r\n
protocol.delimiter[0].delimiterType=str
protocol.delimiter[0].save=true
protocol.delimiter[0].separator=true

protocol.delimiter[1].value=7e,7e
protocol.delimiter[1].delimiterType=arr
protocol.delimiter[1].save=false
protocol.delimiter[1].separator=true

##-------------------------------------

protocol.protocol[0].delimiter=##
protocol.protocol[0].length=2
protocol.protocol[0].start=true
protocol.protocol[0].hex=false
protocol.protocol[0].protocolType=DTU_212
protocol.protocol[0].ProtocolAnalysisInterface=com.datarecv.cloud.server.decode.D
tu212PacketDecode

protocol.protocol[1].delimiter=7e7e ##分隔符为7e7e
protocol.protocol[1].startIndex=0  ##开始的下标为0
protocol.protocol[1].length=2 ##分隔符的长度为2
protocol.protocol[1].start=true
protocol.protocol[1].hex=true ##报文的数据为hex十六进制
protocol.protocol[1].protocolType=RTU_SL561 ##协议的类型，对应rabbitmq的队列名
```

```
protocol.protocol[1].ProtocolAnalysisInterface=com.datarecv.cloud.server.decode.R
tuMessageDecoder ##映射到具体的协议解析类
##------------------------------------------------------------------------------##
```