# Dcm设备接入代码编写指南

作者：李哲弘

时间:2017-07-26 10:22:00

1. **Netty 框架整理**

配置入口

/src/test/java/org/bdc/dcm/dataTabConf.json(收发数据格式配置,全局收发api配置入口)

/src/test/java/org/bdc/dcm/serverConf.json(启动Netty 各种service的配置)

代码入口

/src/test/java/org/bdc/dcm/ExampleService.java(利用”配置入口”启动整体Netty)

自定义启动加载类(用于加载serverConf)->规定了几种service启动的方法

|  |
| --- |
| org.bdc.dcm.netty.HttpClient  org.bdc.dcm.netty.HttpServer  org.bdc.dcm.netty.NettyBoot  org.bdc.dcm.netty.TcpClient  org.bdc.dcm.netty.TcpServer  org.bdc.dcm.netty.UdpConnect |

通道初始化

org.bdc.dcm.netty.channel.http.{ServerType}{DataType}ChannelInitializer

类型枚举配置

org.bdc.dcm.vo.e.DataType -> 对应dataTabConf

org.bdc.dcm.vo.e.ServerType -> 对应serverConf

自定义逻辑层

org.bdc.dcm.netty.channel.http.\* 写关于http初始化

org.bdc.dcm.netty.channel.tcp.\* 写关于 tcp 初始化

org.bdc.dcm.netty.channel.udp.\* 写关于 udp 初始化

1. 模块编写
2. **确定模块 名字(XX,XXJsn),用于org.bdc.dcm.vo.e.DataType编写,第一个是**统一的名称**，第二个是为了与json信息结合**
3. **确定需要的访问(读写)的硬件寄存器结构 将其写在配置**dataTabConf
   1. Id 寄存器地址
   2. Name 寄存器中文标识
   3. form 字段类型用于程序解析
   4. units 字段单位
   5. dataLevel 是否生效
4. 配置serverConf 文件列表
   1. Key 解释

serverType 服务器类型 请在org.bdc.dcm.vo.e.ServerType中查找下标

**dataType 数据类型 请在**org.bdc.dcm.vo.e.DataType中查找下标

**initSendingData 初始化发送的十六进制数据**

**delaySendingTime 循环发送间隔**

dataLevel配置是否生效

**filterMacs [“000000000000000”]接收全部信息**

1. **编写自定义字段类型转换程序**
   1. **org.bdc.dcm.netty.{**dataType**}. {**dataType**}TypeConvert.java**
2. **确定项目硬件协议**
   1. **硬件tcp**
      1. **完成Netty对航宇数据中心编解码包装**
         1. **org.bdc.dcm.netty.channel.tcp.Tcp{**dataType**}ChannelInitializer**
         2. **org.bdc.dcm.netty.framer.{**dataType**}FrameDecoder**
         3. **org.bdc.dcm.netty.coder.tcp.Tcp{**dataType**}Decoder**
         4. **org.bdc.dcm.netty.coder.tcp.Tcp{**dataType**}Encoder**
      2. **完成具体航宇对接设备的编解码**
         1. **org.bdc.dcm.data.coder.{dataType}Decoder**
            1. **关键代码(确定包结构和mac)**

|  |
| --- |
| DataPack dataPack = new DataPack();  dataPack.setMac(mac);  dataPack.setOnlineStatus(1);  dataPack.setDatalevel(Datalevel.NORMAL);  dataPack.setDataPackType(DataPackType.Info); |

* + - 1. **org.bdc.dcm.data.coder.{**dataType**}Encoder**
    1. **编写handler用于具体的逻辑层**
       1. **org.bdc.dcm.netty.handler.{**dataType**}DataHandler**
    2. **编写HttyService 用于接收测试程序的json信息，以调试和硬件tcp协议(这里模拟要接入数据中心服务的异步控制命令接收)**
       1. **Netty接入**

**org.bdc.dcm.netty.channel.http.{**ServerType**}{**dataType**}JsnChannelInitializer**

* + - * 1. **默认需写(只需要写**Encoder**)**

|  |
| --- |
| super.initChannel(ch);  ChannelPipeline pipeline = ch.pipeline();  pipeline.addLast("decoder", new HttpRequestDecoder());  pipeline.addLast("httpAggregator", new HttpObjectAggregator(65536));  pipeline.addLast("clientDecoder", new {ServerType}{dataType}JsnDecoder(getNettyBoot()));  pipeline.addLast("encoder", new HttpResponseEncoder());  pipeline.addLast("clientEncoder", new {ServerType}{dataType}JsnEncoder(getNettyBoot(), new HttpResponseFactory()));  pipeline.addLast("httpChunked", new ChunkedWriteHandler());  pipeline.addLast("dataHandler", new DataHandler(getNettyBoot())); |

* + - 1. **编解码**
         1. **org.bdc.dcm.netty.coder.http.{**ServerType**}{**dataType**}JsnDecoder**

**org.bdc.dcm.data.coder.{**dataType**}JsnDecoder**

**依据”数据中心接口说明书V0.1”解析**

|  |
| --- |
| **DataPack dataPack = new DataPack();**  **dataPack.setToMac(mac);**  **dataPack.setOnlineStatus(1);**  **dataPack.setDatalevel(Datalevel.NORMAL);**  **dataPack.setDataPackType(DataPackType.Cmd);** |

* + - * 1. **org.bdc.dcm.netty.coder.http.{**ServerType**}{**dataType**}JsnEncoder**

**org.bdc.dcm.data.coder.{**dataType**}JsnEncoder**

**随便回复任意字符串 这里只是返回测试的结果 “OK”**

* + 1. **编写HttpClient用于异步回复(这里模拟要接入数据中心服务的异步控制命令发送)**
       1. **org.bdc.dcm.netty.channel.http.{**ServerType**}{**dataType**}JsnChannelInitializer**

|  |
| --- |
| **super.initChannel(ch);**  **ChannelPipeline pipeline = ch.pipeline();**  **pipeline.addLast("decoder", new HttpResponseDecoder());**  **pipeline.addLast("encoder", new HttpRequestEncoder());**  **pipeline.addLast("clientEncoder", new {**ServerType**}{dataType}JsnEncoder(getNettyBoot(), new HttpRequestFactory()));**  **pipeline.addLast("dataHandler", new DataHandler(getNettyBoot()));** |

* + - 1. **编解码**
         1. **org.bdc.dcm.netty.coder.http.{**ServerType**}{**dataType**}JsnEncoder**

**org.bdc.dcm.data.coder.{**dataType**}Jsn1Encoder**

**这里”1”主要是为了防止重复标识是用于测试的client的解码器**

* 1. **硬件http(借鉴tcp)**
  2. **硬件udp(借鉴tcp)**

1. **硬件通讯**
   1. 立创

TcpService

功能：接收本服务器发送给硬件的信息

HttpService

功能：异步收到接入端系统控制命令

HttpClient

功能：异步回复接入端系统硬件信息

* 1. 乐清

TcpService

功能：接收本服务器发送给硬件的信息

HttpService

功能：异步收到接入端系统控制命令

HttpClient

功能：异步回复接入端系统硬件信息

**前端控制**

测试环境

工具：Postmain

消息：json (请与相关框架人员获取获取请求内容模板)

1. **接入端信息接收**

测试环境

工具：tcpserver助手

消息：所有包接收

正式环境

1. **DataHandler**职责：
   1. cmd包：修改socketaddress 找到mac 和 tomac 一致进行转发
   2. Info包：发送给channelGroup中其他的channel 目标是让client异步回复给接入服务器信息，但是为为什么全部转发呢，那岂不是其他网关硬件都会收到？错！因为在serverConf如果是全0那么才会开这个们哦
2. **整体流程**
   1. **Netty主程序**
   2. **Netty<->航宇编解码**
   3. **航宇Dcm编解码**
   4. **HttpServer测试接入服务**
   5. **HttpClinent测试发送数据服务**