**ws连接地址：ws://127.0.0.1:8086/masterpiece**

注意使用websocket连接时的subprotocol需填写**private:7**

数据帧结构：

业务数据(payload)包裹在数据帧结构中。

|  |  |  |  |
| --- | --- | --- | --- |
| 字节偏移 | 字段 | 长度(字节,byte) | 说明 |
| 0 | 帧头 | 2 | 固定不变 0x6D37 |
| 2 | 协议版本 | 1 | 用于协议升级扩展,当前版本为0x00 |
| 3 | fnCode | 1 | 功能码，心跳=1 Order=2 OrderResp=3 |
| 4 | 数据总长度 | 2 | 数据长度N |
| 6 | 数据payload | N | protobuf编码的二进制数据 |
| 6 + N | 校验码 | 2 | CRC-16/MODBUS |

业务数据使用[ProtoBuf v3版本](https://protobuf.dev/programming-guides/proto3/)封装，.proto文件内容见后文

CRC校验计算从帧头至payload (包含帧头和payload)关于CRC校验，参考<http://www.ip33.com/crc.html>

Proto文件：

//1. 心跳命令,客户端主动向服务器上传，服务器收到后立即原样回复客户端。建议间隔1分钟。  
message Heartbeat {  
}  
  
//2 通用命令请求  
message Order {  
 string uuid = 1;//请求id  
 string service = 2;//约定的service名称(每个可控设备抽象为一个service)  
 string data = 3;//命令数据, jsonRpc 调用  
}  
  
//3 通用命令响应  
message OrderResp {  
 string uuid = 1;//请求id  
 string data = 4;//jsonRpc响应  
}

## JsonRpc

在已有项目轻量使用了JsonRpc，有库支持，可阅读性好，可以方便的在mqtt、http、串口、自定义tcp等各种协议中传输，总体评价简单好用。

Deamon程序作为被调用端实现以下接口，UI程序进行登录

@ServiceDescription(model = "mpdaemon", desc = "daemon登录服务")  
interface MPDaemonService {  
 //登录失败请退出程序  
 @FunctionDescription("登录")  
 @ReturnTypeDescription("是否登录成功")  
 fun authorized(  
 @ParamDescription("开发代号") id: String,  
 @ParamDescription("软件版本") version: String,  
 @ParamDescription("设备分类(例如UI为MPUI)") group: String,  
 @ParamDescription("协议版本") protoVersion: Int  
 ): Boolean  
  
}

UI程序作为被调用端需要实现以下RPC接口， Deamon程序负责调用这些接口

@ServiceDescription(model = "mpui", desc = "ui控制指令服务")  
interface MPUiService {  
  
 @FunctionDescription(desc = "复位屏幕显示")  
 fun reset(): Boolean  
  
 @FunctionDescription(desc = "切换屏幕显示")  
 fun setScreen(screenId: String): Boolean  
  
 @FunctionDescription(desc = "设置文字")  
 fun setText(resourceId: String, content: String): Boolean  
  
 @FunctionDescription(desc = "图标切换")  
 fun setIcon(resourceId: String, iconId: String): Boolean  
  
 @FunctionDescription(desc = "颜色切换")  
 fun setColor(resourceId: String, color: String): Boolean  
}

### 服务名称：daemon登录服务

ProtoBuf中service字段: mpdaemon

#### 1.登录

请求示例: { "method": "authorized", "id": id, "params": [arg0,arg1,arg2,arg3]}

id: 随机生成int,再转为string, 例如"104567"

参数介绍:

arg0 String 开发代号（现在随意填写，以后会对每一种程序统一分配）  
  
arg1 String 软件版本  
  
arg2 String 设备分类（现在随意填写，以后会对每一种程序统一分配，例如UI为MPUI)  
  
arg3 int 协议版本 填0

返回示例： {"jsonrpc":"2.0","id":请求的id,"result":执行结果}

结果解释： 是否登录成功

结果类型： boolean

### 服务名称：ui控制指令服务

ProtoBuf中service字段: mpui

#### 1.切换屏幕显示

请求示例: { "method": "setScreen", "id": id, "params": [arg0]}

id: 随机生成int,再转为string, 例如"104567"

参数介绍:

arg0 String

返回示例： {"jsonrpc":"2.0","id":请求的id,"result":执行结果}

结果解释：

结果类型： boolean

#### 2.设置文字

请求示例: { "method": "setText", "id": id, "params": [arg0,arg1]}

id: 随机生成int,再转为string, 例如"104567"

参数介绍:

arg0 String   
  
arg1 String

返回示例： {"jsonrpc":"2.0","id":请求的id,"result":执行结果}

结果解释：

结果类型： boolean

#### 3.图标切换

请求示例: { "method": "setIcon", "id": id, "params": [arg0,arg1]}

id: 随机生成int,再转为string, 例如"104567"

参数介绍:

arg0 String   
  
arg1 String

返回示例： {"jsonrpc":"2.0","id":请求的id,"result":执行结果}

结果解释：

结果类型： boolean

#### 4.颜色切换

请求示例: { "method": "setColor", "id": id, "params": [arg0,arg1]}

id: 随机生成int,再转为string, 例如"104567"

参数介绍:

arg0 String   
  
arg1 String "#RGB" or "#RGBA"

返回示例： {"jsonrpc":"2.0","id":请求的id,"result":执行结果}

结果解释：

结果类型： boolean

#### 5.复位屏幕显示

请求示例: { "method": "reset", "id": id, "params": []}

id: 随机生成int,再转为string, 例如"104567"

返回示例： {"jsonrpc":"2.0","id":请求的id,"result":执行结果}

结果解释：

结果类型： boolean

## 开发测试

提供masterpiece-proxy-1.0.0-masterpiece-proxy.jar 作为模拟Daemon进程服务器, 提供给前端开发使用， 用于协议对接和效果验证。

该程序模拟了连接的协议，并且提供了手动向UI发送命令的交互web界面

运行需要先安装Java11以上

运行执行命令：

java -jar masterpiece-proxy-1.0.0-masterpiece-proxy.jar

然后可以在浏览器中访问`<http://127.0.0.1:8086/index.html>`