万威电池与云服务器间数据交互协议

文件状态：	文件标识：	PRD
[ ] 草稿	当前版本：	V1.9
[√] 正式发布	作　　者：	宋书群
[ ] 正在修改	完成日期：	2019-11-04

版 本 历 史

版本	作者	参与者	起止日期	备注
v0.1	宋书群	卢总 时总 沈总	2019.04.02 - 2019.04.09	初步定义完成
v1.0	宋书群	卢总 时总 沈总	2019.04.10 - 2019.04.15	讨论后形成正式标准 改动如下： 1. 充放电使能取消随机数相关 2. 电池升级中增加类型（BMS/通讯板） 3. 升级的结果中也增加类型 4. 充放电使能增加上线时下发，mode 5. 升级增加上线时下发，mode
v1.1	宋书群		2019.05.05 - 2019.05.15	1. frame 包中 serial_num 改名 serail_no 2. 将之前新威协议中的取值范围移植到本文档 3. 拆分使能包，把进水检测和睡眠指令拆到新包。 4. 结果上报电池规格信息 ICCID 用 20 个字节表示 5. 结果上报编码信息，code 注释为万威编码
v1.2	宋书群	王工	2019.05.16	1. 进水检测和睡眠使能取值改为 1，2 2. 蜂鸣器 status 取值为 1，2 3. 充放电使能结果上报第一个字段从 mode 改为来包序列号 4. 电池保护参数中的单体欠压保护限注释修改
v1.3	宋书群	时总	2019.05.30	1. Summery 下 soc 字段精度变为 1
v1.4	宋书群	王工	2019.06.25	1. 周期包改为多个包一起报
v1.5	宋书群		2019.07.11	1. 增加对加密和秘钥更新的说明 2. 结果上报更新秘钥两个包中 status 类型改为 u8
v1.6	宋书群	王工	2019.09.23	1. 增加设置和应答仓储模式指令 2. Hello 包增加海拔 3. 周期包增加 utc 和单体电压和均衡 4. 结果上报规格信息增加循环次数，万威

				编码，imei 和各种版本信息 5. 增加 MOS 温度差阈值设置指令
v1.7	宋书群	时总 王工	2019.10.10	1. 将 Hello 包中的信息放到概要包子包中，暂不删除 Hello 中的内容。 2. 结果上报电池规格信息中 times 改为 u16 charge_count 增加服务器通讯协议版本 protocolVer 字段；通讯模组类型 commModuleType 字段；通讯连接类型 commLinkType 字段；通讯板内核版本号 commKernelVer 字段 3. 周期上报中删除 soft_ver 4. 增加 BMS SOC 基准容量命令
v1.8	宋书群	王工 时总	2019.11.04	1. 去除 Hello 包中内容，增加一个 utc 2. 周期包中 index 去除，duration 去除 3. Soc 和 mos 用 u16 类型数据
v1.9	宋书群	刘工 王工 时总	2019.11.12	1. 增加配置信息设置 2. 增加配置信息请求 3. 增加保护参数设置

1.文档介绍

1.1. 文档目的

该文档描述了电池与云服务器之间交互的总体机制和详细协议及要求。该协议是电池设备进入万威平台的具体要求，符合该协议，技术上具备了接入万威平台的前提条件。

1.2. 文档范围

该文档涉及电池设备与云服务器之间交互的内容。

1.3. 读者对象

- 项目组相关成员
- 需要接入的电池厂商

1.4. 参考文档

具体的统计分析内容需要参考：

1.

2.

1.5. 术语与缩写解释

缩写、术语	解 释
请求	服务器向电池请求数据
反馈	服务器或电池收到请求设置等命令后告知对方已经收到数据
上报	电池向服务器上报数据
设置	服务器向电池设置参数
结果上报	电池向服务器保送请求或设置的结果状态
…	

2.数据格式

交互数据帧结构:

```
struct FRAME_STRUCT{
   u32   dst_battery_id;    //目的设备的编号
   u32   crc;               //crc 是整个数据 32bit 校验和。
   u16   trans_length;      //加密后 CMD_Frame 的实际长度
   u16   cmd_data;          //命令字
   u32   serial_no;         //流水单号
   u8    data[CMD_LENGTH]; //命令内容
};
```

2.1. 更新密钥请求

电池向服务器申请更新密钥。
第一次请求更新密钥时，因为设备中没有密钥，所以用设备 ID 的 MD5 算法计算一个值作为密钥，用这个密钥加密数据，服务端验证密钥是否正确，如果正确服务器生成一个新的密钥返回给电池，此后通讯都用最新的密钥加解密，直到下一次更新密钥。
更新密钥协议采用的是同步方式执行。

命令

```
#define CMD_BATTERY_KEY_UPDATE_REQ           0x101e
struct BATTERY_KEY_UPDATE_REQ {
   u16 random_encrypt_len;                  //密文长度
   u8[random_encrypt_len] random_encrypt;  //加密后的随机数
};
```

应答：

```
#define CMD_SVR_KEY_UPDATE_ACK               0x401e
struct SVR_KEY_UPDATE_ACK {
   u32 src_serial_no;                       //来包序列号
   u16 key_encrypt_len;                     //新密钥加密后长度
   u8[key_encrypt_len] new_key_encrypt;     //旧密钥加密新密钥数据
   u16 random_encrypt_len;                  //随机数加密后长度
```

```
    u8[random_encrypt_len] random_encrypt; //新加密后的随机数
};
```

2.2. 结果上报更新密钥

电池向服务器上报更新密钥结果。服务器端收到秘钥更新成功（status=0）后将设备秘钥更新为新生成的秘钥，如果失败，将设备秘钥跟新为旧秘钥。
如果电池未收到服务器回复的 401F 指令，则需要持续发送 101F，直到收到 401F 指令。
特别要注意，在未收到 401F 指令时，不能重新更新秘钥。

命令

```
#define CMD_BATTERY_KEY_UPDATE_RSLT           0x101f
struct BATTERY_KEY_UPDATE_RSLT {
    u32 src_serial_no;          //密钥更新请求包来包序列号
    u8 status;                  //0 成功
                                //1 失败
};
```

应答：

```
#define CMD_SVR_KEY_UPDATE_ACK                0x401f
struct SVR_KEY_UPDATE_ACK {
    u32 src_serial_no;          //命令包来包序列号
    u8 status;                  //0 成功
                                //1 失败
};
```

2.3. 上报心跳包

电池向服务器发送心跳包，以保活设备跟服务器的链接。注意，此协议是基于长链接的模式，因此心跳包仅作为定时发给服务器的，为了提高可靠性的，一种手段。
另外服务器端用心跳包建立连接，维护链接。

命令：

```
#define CMD_BATTERY_HELLO_RPT           0x1000
struct BATTERY_HELLO_RPT {
       u32 utc;        //utc 时间
```

```
                              //精度：1s
                              //单位：s
                              //0:无效数据

};
```

应答：

```
#define CMD_SVR_HELLO_ACK                    0x4000
struct SVR_HELLO_ACK {
   u32 src_serial_no;    //来包序列号
   u8 status;            //0 正常
                         //1 异常
};
```

2.4. 上报电池概要信息

电池每个周期采集概信息(10s)，在适当的时机上报电池概要信息。
其时机如下所列：
采集信息达到指定的条数时
采集信息达到指定的时间范围时
采集信息时发现超过阀值，需要马上报警时
命令：

```
#define CMD_BATTERY_SUMMERY_RPT         0x1001
struct BATTERY_SUMMERY_RPT {
   u16 count;            //包数
   struct {
      u8 soc;            //电量
                         //精度：1/100 偏移：0 单位：百分比
      u8 bms_status;     //BMS 工作状态
                         //0:搁置 1:放电 2:充电
      u32 bms_err_code;  //BMS 故障代码
      u32 err_warn;      //故障报警预留
      u16 total_voltage;//电池包总电压
                         //精度：1/10 偏移：0 单位：V
      u16 total_current;//电池包总电流
                         //精度：1/10 偏移：-3200 单位：A
      u16 max_item_voltage;//最大单体电压
                         //精度：1 偏移：0 单位：MV
```

```
u16 min_item_voltage;//最低单体电压
                    //精度：1 偏移：0 单位：MV
u16 avg_item_voltage;//平均单体电压
                    //精度：1 偏移：0 单位：MV
u8 max_temp;        //最高电池温度
                    //精度：1 偏移：-40 单位：℃
u8 min_temp;        // 最低电池温度
                    //精度：1 偏移：-40 单位：℃
u8 out_mos_temp;        //放电 MOS 温度
                    //精度：1 偏移：-40 单位：℃
u8 in_mos_temp;         //充电 MOS 温度
                    //精度：1 偏移：-40 单位：℃
u8 equalizing_resistance_temp;//均衡电阻温度
                    //精度：1 偏移：-40 单位：℃
u8 in_mos_status;       //充电 MOS 状态
                    //1 断开  2 吸合
u8 in_mos_status;       //放电 MOS 状态
                    //1 断开  2 吸合
u8 signal_intensity; //GPRS 信号强度
                    //取值 0 - 31
u8 balance;         //均衡
                //0:均衡
                //其他不均衡
u32 utc;        //utc 时间
                    //精度：1s
                    //单位：s
                    //0:无效数据
u8 updated;     //经纬度信号更新
                    //0 无更新
                    //1 有更新
u8 location_type; //定位方式
                    //1 GPS
                    //2 基站
u32 longitude; //经度
                    //正数东经 负数西经
                    //精度：1/1000000
                    //偏移量： 0
u32 latitude;  //纬度
                    //正数北纬 附属南纬
                    //精度：1/1000000
```

```
                        //偏移量： 0
    u16 speed;      //速度
                        //精度：百分之一
                        //单位：KM/H
     u16 altitude;  //海拔
                        //精度:3M
                        //单位:M
                        //0:无效数据

    u8 battery_count;                   //电池串数
    u16[battery_count] valtages;        //电池电压
                                        //精度：1 偏移：0 单位：MV
  }summery_info[num];
};
```

注：

序号 index：每包从 0 开始，假设周期为十秒，每十秒采集一次序号加一，如果有一个采集周期没有数据，下一个数据的需要要加上 2 了，同理如果三次未采集下一条加 4

应答：

```
#define CMD_SVR_SUMMERY_ACK    0x4001
struct SVR_SUMMERY_ACK   {
   u32 src_serial_no;       //来包序列号
   u8 status;               //0 正常
                            //1 异常
};
```

2.5. 请求电池规格信息

云服务器请求电池将自己的规格信息发送到服务器。

命令：

```
#define CMD_SVR_STANDARD_REQ         0x1002
struct SVR_STANDARD_REQ {
}
```

应答：

```
#define CMD_BATTERY_STANDARD_ACK     0x4002
struct BATTERY_STANDARD_ACK {
   u32 src_serial_no;       //来包序列号
   u8 status;               //0 正常
                            //1 异常
};
```

2.6. 结果上报电池规格信息

电池在收到【请求电池规格信息】后收集规格信息，并立即上报到云服务器。

命令：

```
#define CMD_BATTERY_STANDARD_RSLT       0x1003
struct BATTERY_STANDARD_RSLT {
   u32 src_serial_no;    //来包序列号
   u8 year;              //生产日期年 17 代表 2017
   u8 month;             //生产日期月
   u8 day;               //生产日期日
   u8 count;             //单体电池的节数
   u16 capacity;         //电池包额定容量
                         //精度：1/100 偏移：0 单位：AH
   u8 temp_probe_count; //单体温度探头个数
   u8 battery_type;      //电池类型
                         //1  232 通讯    2  485 通讯
   u8 exchange_type;     //与整车通讯方式
   u8[20] iccid;         //ICCID 每个字节代表手机号的一个数字
   u16 charge_count;     //充电次数
   u8 length;            //编码长度
   u8[length] code;      //万威编码
   u8[15] IMEI;         //IMEI 通讯设备全球唯一编号
   u8 comm_hard_ver;     //通讯板硬件版本号
   u8 comm_soft_ver;     //通讯板软件版本号
   u8 bms_hard_ver;         //BMS 硬件版本
   u8 bms_soft_ver;         //BMS 软件版本
   u8 protocol_ver;      //服务器通讯协议版本
   u8 comm_module_type; //通讯模组类型：2:2G、4:4G
   u8 comm_link_type;       //通讯连接类型：2:2G、3:3G、4:4G
```

```
    u8 comm_kernel_ver;      //通讯板内核版本号
};
```

注：

组包时需要携带【请求电池规格信息】的 serial_no。

应答：

```
#define CMD_SVR_STANDARD_ACK      0x4003
struct SVR_STANDARD_ACK  {
    u32 src_serial_no;       //来包序列号
    u8 status;               //0 正常
                             //1 异常
};
```

注：

组包时需要携带【上报电池规格信息】的 serial_no。

2.7. 请求电池包详细信息

云服务器请求电池将自己的详细信息发送到服务器。

命令：

```
#define CMD_SVR_DETAIL_REQ        0x1004
struct SVR_DETAIL_REQ {
}
```

应答：

```
#define CMD_BATTERY_DETAIL_ACK    0x4004
struct BATTERY_DETAIL_ACK {
    u32 src_serial_no;       //来包序列号
    u8 status;               //0 正常
                             //1 异常
};
```

2.8. 结果上报电池详细信息

电池在收到【请求电池详细信息】后收集规格信息，并立即上报到云服务器。

命令：

```
#define CMD_BATTERY_DETAIL_RSLT         0x1005
struct BATTERY_DETAIL_RSLT {
   u32 src_serial_no;                  //来包序列号
   u8 battery_count;                   //电池个数
   u16[battery_count] valtages;        //电池电压
                                       //精度：1 偏移：0 单位：MV
   u8 temp_detective_count;            //温度探测器个数
   u8[temp_detective_count] temp;      //电池温度
                                       //精度：1 偏移：-40 单位：℃
};
```

注：
组包时需要携带【请求电池详细信息】的 serial_no。

应答：

```
#define CMD_SVR_DETAIL_ACK     0x4005
struct SVR_DETAIL_ACK {
   u32 src_serial_no;    //来包序列号
   u8 status;               //0 正常
                            //1 异常
};
```

注：
组包时需要携带【上报电池详细信息】的 serial_no。

2.9. 请求电池历史信息

云服务器请求电池将自己的历史信息发送到服务器。

命令：

```
#define CMD_SVR_HISTORY_REQ       0x1006
struct SVR_HISTORY_REQ {
}
```

应答：

```
#define CMD_BATTERY_HISTORY_ACK  0x4006
struct BATTERY_HISTORY_ACK {
```

```
   u32 src_serial_no;       //来包序列号
   u8 status;               //0 正常
                            //1 异常
};
```

2.10. 结果上报电池历史信息

电池在收到【请求电池历史信息】后收集规格信息，并立即上报到云服务器。

命令：

```
#define CMD_BATTERY_HISTORY_RSLT        0x1007
struct BATTERY_HISTORY_RSLT {
   u32 src_serial_no;             //来包序列号
   u16 charge_count;              //充电次数
   u32 total_discharge_ah;        //累计放电 AH 数
                                  //单位：AH
   u16 hightest_item_valtages;    //历史最高单体电压
                           //精度：1 偏移：0  单位：MV
   u16 lowest_item_valtages;      //历史最低单体电压
                           //精度：1 偏移：0  单位：MV
   u16 hightest_temp;             //历史最高温度
                           //精度：1 偏移：-40  单位：℃
   u16 lowest_temp;               //历史最低温度
                           //精度：1 偏移：-40  单位：℃
   u16 bigest_discharge_current; //历史最大放电电流
                           //精度：1/10 偏移：-3200  单位:A
   u16 bigest_charge_current;     //历史最大充电电流
                           //精度：1/10 偏移：-3200  单位:A
};
```

注：
组包时需要携带【请求电池历史信息】的 serial_no。

应答：

```
#define CMD_SVR_HISTORY_ACK    0x4007
struct SVR_HISTORY_ACK   {
   u32 src_serial_no;       //来包序列号
   u8 status;               //0 正常
```

```
                                        //1 异常
};
```

注：

组包时需要携带【上报电池历史信息】的 serial_no。

2.11. 请求电池编码信息

云服务器请求电池将自己的编码信息发送到服务器。

命令：

```
#define CMD_SVR_CODE_REQ          0x1008
struct SVR_CODE_REQ {
}
```

应答：

```
#define CMD_BATTERY_CODE_ACK      0x4008
struct BATTERY_CODE_ACK {
   u32 src_serial_no;    //来包序列号
   u8 status;            //0 正常
                         //1 异常
};
```

2.12. 结果上报电池编码信息

电池在收到【请求电池编码信息】后收集规格信息，并立即上报到云服务器。

命令：

```
#define CMD_BATTERY_CODE_RSLT        0x1009
struct BATTERY_CODE_RSLT {
   u32 src_serial_no;    //来包序列号
   u8 length;            //编码长度
   u8[length] code;      //万威编码
};
```

注：

组包时需要携带【请求电池编码信息】的 serial_no。

应答：

```
#define CMD_SVR_CODE_ACK              0x4009
struct SVR_CODE_ACK   {
   u32 src_serial_no;       //来包序列号
   u8 status;               //0 正常
                            //1 异常
};
```

注：
组包时需要携带【上报电池编码信息】的 serial_no。

2.13. 设置电池充放电使能

云服务器设置电池充放电使能信息。

命令：

```
#define CMD_SVR_ENABLED_SET       0x100a
struct SVR_ENABLED_SET {
   u8 mode;                     //下发模式 0 超时模式； 1 暂存模式；
   u8 charge;            //控制充电使能
                                //0x0:不控制，归 BMS 自动控制
                                //0x01：强制吸合充电 MOS
                                //0x02：强制断开充电 MOS
   u8 discharge;         //控制放电使能
                                //0x0:不控制，归 BMS 自动控制
                                //0x01：强制吸合充电 MOS
                                //0x02：强制断开充电 MOS
}
```

应答：

```
#define CMD_BATTERY_ENABLED_ACK      0x400a
struct BATTERY_ENABLED_ACK {
   u32 src_serial_no;    //来包序列号
   u8 status;            //0 正常
                         //1 异常
};
```

2.14. 结果上报设置电池充放电使能状态

电池在收到【设置电池充放电使能】后设置，并将结果上报到云服务器。

命令：

```
#define CMD_BATTERY_ENABLED_RSLT          0x100b
struct BATTERY_ENABLED_RSLT {
   u32 src_serial_no;    //来包序列号
   u8 charge;            //控制充电使能
                                 //0x0:不控制，归 BMS 自动控制
                                 //0x01：强制吸合充电 MOS
                                 //0x02：强制断开充电 MOS
   u8 discharge;         //控制放电使能
                                 //0x0:不控制，归 BMS 自动控制
                                 //0x01：强制吸合充电 MOS
                                 //0x02：强制断开充电 MOS
};
```

注：
组包时需要携带【设置电池充放电使能】的 serial_no。

应答：

```
#define CMD_SVR_ENABLED_ACK     0x400b
struct SVR_ENABLED_ACK {
   u32 src_serial_no;        //来包序列号
   u8 status;                //0 正常
                             //1 异常
};
```

注：
组包时需要携带【上报设置电池充放电使能结果】的 serial_no。

2.15. 设置电池保护参数

云服务器设置电池保护参数。

命令：

```
#define CMD_SVR_PARAMS_SET        0x100c
struct SVR_PARAMS_SET {
   u16 max_item_voltage;        //单体欠压保护限值
            //精度：1mv/bit     偏移量：0
            //出厂默认值：2800mv
            //修改范围：  2000mv- 4200mv
   u16 overcurrent_discharge; //放电过流限值
            //精度：0.1A/bit     偏移量：3000
            //出厂默认值：65A
            //修改范围：  0-150A
   u16 max_temp;                //电池温度过高保护限值
            //精度：1℃/bit 偏移量：-40
            //电池温度过高保护出厂默认值：60℃
            //电池温度过低保护出厂默认值：-20℃
            //MOS 温度过高出厂默认值：85℃
            //修改范围：  -30-100℃
   u8 max_mos_temp;             //MOS 温度过高限制
            //精度：1℃/bit 偏移量：-40
            //电池温度过高保护出厂默认值：60℃
            //电池温度过低保护出厂默认值：-20℃
            //MOS 温度过高出厂默认值：85℃
            //修改范围：  -30-100℃
}
```

应答：

```
#define CMD_BATTERY_PARAMS_ACK    0x400c
struct BATTERY_PARAMS_ACK {
   u32 src_serial_no;    //来包序列号
   u8 status;            //0 正常
                         //1 异常
};
```

2.16. 结果上报设置电池保护参数

电池在收到【设置电池保护参数】后设置参数，并将结果上报到云服务器。

命令：

```
#define CMD_BATTERY_PARAMS_RSLT 0x100d
struct BATTERY_PARAMS_RSLT {
    u32 src_serial_no;            //来包序列号
    u16 max_item_voltage;         //单体电压保护限值
              //精度：1mv/bit     偏移量：0
              //例如：欠压保护限值高字节为 0xA0，欠压保护限值低字节为 0xF0；
              //则欠压保护限值为 0xAF0；十进制值即为 2800mv
              //出厂默认值：2800mv
              //修改范围： 2000mv- 4200mv
    u16 overcurrent_discharge; //放电过流限值
              //精度：0.1A/bit     偏移量：3000
              //例如想设置设置放电过流限值为 50A；则
              //放电过流限值 = （50+3000）* 10 = 30500；
              //放电过流限值高字节为 0x77，放电过流限值低字节为 0x24。
              //出厂默认值：65A
              //修改范围： 0-150A
    u16 max_temp;                 //电池温度过高保护限值
              //精度：1℃/bit 偏移量：-40
              //电池温度过高保护出厂默认值：60℃
              //电池温度过低保护出厂默认值：-20℃
              //MOS 温度过高出厂默认值：85℃
              //修改范围： -30-100℃
    u8 max_mos_temp;              //MOS 温度过高限制
              //精度：1℃/bit 偏移量：-40
              //电池温度过高保护出厂默认值：60℃
              //电池温度过低保护出厂默认值：-20℃
              //MOS 温度过高出厂默认值：85℃
              //修改范围： -30-100℃
};
```

注：

组包时需要携带【设置电池保护参数】的 serial_no。

应答：

```
#define CMD_SVR_PARAMS_ACK      0x400d
struct SVR_PARAMS_ACK  {
    u32 src_serial_no;     //来包序列号
    u8 status;             //0 正常
```

```
                                    //1 异常
};
```

注：

组包时需要携带【上报设置电池保护参数结果】的 serial_no。

2.17. 设置电池蜂鸣器报警

云服务器请求电池将自己的规格信息发送到服务器。

命令：

```
#define CMD_SVR_BUZZER_SET        0x100e
struct SVR_BUZZER_SET {
   u8 status;  //2 开启
               //1 关闭
   u8 seconds; //报警时长，单位秒
}
```

应答：

```
#define CMD_BATTERY_BUZZER_ACK    0x400e
struct BATTERY_BUZZER_ACK {
   u32 src_serial_no;       //来包序列号
   u8 status;               //0 正常
                            //1 异常
};
```

2.18. 结果上报设置电池蜂鸣器报警信息

电池在收到【设置电池蜂鸣器报警】后收集规格信息，并立即上报到云服务器。

命令：

```
#define CMD_BATTERY_BUZZER_RSLT       0x100f
struct BATTERY_BUZZER_RSLT {
   u32 src_serial_no;     //来包序列号
   u8 seconds;            //报警时长，单位秒
};
```

注：

组包时需要携带【设置电池蜂鸣器报警】的 serial_no。

应答：

```
#define CMD_SVR_BUZZER_ACK      0x400f
struct SVR_BUZZER_ACK  {
   u32 src_serial_no;        //来包序列号
   u8 status;                //0 正常
                             //1 异常
};
```

注：
组包时需要携带【上报设置电池蜂鸣器报警结果】的 serial_no。

2.19. 设置电池升级

云服务器设置电池升级。

命令：

```
#define CMD_SVR_UPGRADE_SET       0x1010
struct SVR_UPGRADE_SET {
   u8 mode;     //下发模式 0 超时模式； 1 暂存模式；
   u8 type;     //目标类型 0 BMS； 1 通讯板
   u8 length;   //地址长度
   u16[length] bin_url; //bin 地址 URL
}
```

应答：

```
#define CMD_BATTERY_UPGRADE_ACK      0x4010
struct BATTERY_UPGRADE_ACK {
   u32 src_serial_no;       //来包序列号
   u8 status;               //0 正常
                            //1 异常
};
```

2.20. 结果上报电池升级

电池在收到【设置电池升级信息】后，执行升级动作，完成后立即上报到云服务器。

命令：

```
#define CMD_BATTERY_UPGRADE_RSLT          0x1011
struct BATTERY_UPGRADE_RSLT {
   u32 src_serial_no;     //来包序列号
   u8 type;     //目标类型 0 BMS； 1 通讯板
   u8 result;             //升级结果
   u16 soft_ver;          //软件版本
};
```

注：
组包时需要携带【设置电池升级】的 serial_no。

应答：

```
#define CMD_SVR_UPDATE_ACK             0x4011
struct SVR_UPDATE_ACK  {
   u32 src_serial_no;        //来包序列号
   u8 status;                //0 正常
                             //1 异常
};
```

注：
组包时需要携带【上报电池升级结果】的 serial_no。

2.21. 设置电池进水检测与睡眠使能

云服务器设置电池充放电使能信息。

命令：

```
#define CMD_SVR_ENABLED_SET       0x1012
struct SVR_ENABLED_SET {
   u8 mode;                       //下发模式 0 超时模式； 1 暂存模式；
   u8 water_check;    //控制保护板进水检测功能
                                  //0x01：关闭进水检测功能
                                  //0x02：进水功能开启
   u8 sleep;          //控制保护板不去休眠
                                  //0x01：保护板不允许休眠
                                  //0x02：保护板的休眠功能由自己控制
}
```

应答：

```
#define CMD_BATTERY_ENABLED_ACK      0x4012
struct BATTERY_ENABLED_ACK {
   u32 src_serial_no;    //来包序列号
   u8 status;            //0 正常
                         //1 异常
};
```

2.22. 结果上报设置电池进水检测与睡眠使能状态

电池在收到【设置电池进水检测与睡眠使能】后设置，并将结果上报到云服务器。

命令：

```
#define CMD_BATTERY_ENABLED_RSLT         0x1013
struct BATTERY_ENABLED_RSLT {
   u32 src_serial_no; //来包序列号
   u8 water_check;    //控制保护板进水检测功能
                              //0x01：关闭进水检测功能
                              //其它：进水功能开启
   u8 sleep;          //控制保护板不去休眠
                              //0x01：保护板不允许休眠
                              //其它：保护板的休眠功能由自己控制

};
```

注：
组包时需要携带【设置电池进水检测与睡眠使能】的 serial_no。

应答：

```
#define CMD_SVR_ENABLED_ACK     0x4013
struct SVR_ENABLED_ACK {
   u32 src_serial_no;        //来包序列号
   u8 status;                //0 正常
                             //1 异常
};
```

注：
组包时需要携带【上报设置电池充放电使能结果】的 serial_no。

2.23. 设置仓储模式(暂不支持)

云服务器设置电池进入仓储模式。

命令：

```
#define CMD_SVR_STORAGE_SET        0x1014
struct SVR_STORAGE_SET {
   u8 store; //1 进入仓储模式，通讯板并断电
}
```

应答：

```
#define CMD_BATTERY_STORAGE_ACK    0x4014
struct BATTERY_STORAGE_ACK {
   u32 src_serial_no;        //来包序列号
   u8 status;                //0 正常
                             //1 异常
};
```

2.24. 应答仓储模式(暂不支持)

电池在收到【设置仓储模式】命令后，响应服务器，随后 BMS 给通讯板断电进入最低功耗。

命令：

```
#define CMD_BATTERY_STORAGE_RSLT         0x1015
struct BATTERY_STORAGE_RSLT {
   u32 src_serial_no;     //来包序列号
   u8 store;           //1:进入仓储
};
```

注：

组包时需要携带【设置仓储模式】的 serial_no。

应答：

```
#define CMD_SVR_STORAGE_ACK     0x4015
struct SVR_STORAGE_ACK {
   u32 src_serial_no;        //来包序列号
   u8 status;                //0 正常
                          //1 异常
};
```

注：

2.25. 设置 BMS SOC 基准容量

云服务器设置 BMS SOC 基准容量。

命令：

```
#define CMD_SVR_BMS_SOC_SET        0x1016
struct SVR_BMS_SOC_SET {
   u16 soc;  //soc 基准值
}
```

应答：

```
#define CMD_BATTERY_BMS_SOC_ACK    0x4016
struct BATTERY_BMS_SOC_ACK {
   u32 src_serial_no;        //来包序列号
   u8 status;                //0 正常
                             //1 异常
};
```

2.26. 应答设置 BMS SOC 基准容量

电池在收到【设置仓储模式】命令后，响应服务器，随后 BMS 给通讯板断电进入最低功耗。

命令：

```
#define CMD_BATTERY_BMS_SOC_RSLT          0x1017
struct BATTERY_BMS_SOC_RSLT {
   u32 src_serial_no;     //来包序列号
   u16 soc;
};
```

注：

组包时需要携带【设置仓储模式】的 serial_no。

应答：

```
#define CMD_SVR_BMS_SOC_ACK     0x4017
struct SVR_BMS_SOC_ACK {
   u32 src_serial_no;        //来包序列号
   u8 status;                //0 正常
                          //1 异常
};
```

注：

2.27. 设置 MOS 阈值

云服务器设置电池 MOS 阈值。
命令：

```
#define CMD_SVR_MOS_SET          0x1018
struct SVR_MOS_SET {
    u16 mos;  //MOS 阈值
}
```

应答：

```
#define CMD_BATTERY_MOS_ACK      0x4018
struct BATTERY_MOS_ACK {
    u32 src_serial_no;          //来包序列号
    u8 status;                  //0 正常
                                //1 异常
};
```

2.28. 结果上报设置 MOS 阈值

电池在收到【设置 MOS 阈值】命令后，反馈设置结果。
命令：

```
#define CMD_BATTERY_MOS_RSLT        0x1019
struct BATTERY_MOS_RSLT {
    u32 src_serial_no;     //来包序列号
    u16 mos;
};
```

注：
组包时需要携带【设置 MOS 阈值】的 serial_no。
应答：

```
#define CMD_SVR_MOS_ACK       0x4019
struct SVR_MOS_ACK  {
    u32 src_serial_no;          //来包序列号
    u8 status;                  //0 正常
                                //1 异常
};
```

注：

2.29. 设置电池配置信息

命令：

```
#define CMD_SVR_CONFIG_SET      0x1020
struct SVR_CONFIG_SET {
   u16 config;
};
```

应答：

```
#define CMD_BATTERY_CONFIG_ACK     0x4020
struct BATTERY_CONFIG_ACK {
   u32 src_serial_no;        //来包序列号
   u8 status;                //0 正常
                             //1 异常
};
```

2.30. 结果上报电池配置信息

命令：

```
#define CMD_BATTERY_CONFIG_RSLT    0x1021
struct BATTERY_CONFIG_RSLT{
   u32 src_serial_no;        //来包序列号
   u16 config;
};
```

应答：

```
#define CMD_SVR_CONFIG_ACK      0x4021
struct SVR_CONFIG_ACK {
   u32 src_serial_no;        //来包序列号
   u8 status;                //0 正常
                             //1 异常
};
```

2.31. 请求电池配置信息

云服务器请求电池将自己的规格信息发送到服务器。

命令：

```
#define CMD_SVR_CONFIG_REQ        0x1022
```

```
struct SVR_CONFIG_REQ {
}
```

应答：

```
#define CMD_BATTERY_CONFIG_ACK    0x4022
struct BATTERY_CONFIG_ACK {
   u32 src_serial_no;       //来包序列号
   u8 status;               //0 正常
                            //1 异常
};
```

2.32. 结果上报电池配置信息

命令：

```
#define CMD_BATTERY_CONFIG_RSLT         0x1023
struct BATTERY_CONFIG_RSLT {
   u32 src_serial_no;    //来包序列号
   u16 config;              //
};
```

注：

应答：

```
#define CMD_SVR_COFIG_ACK      0x4023
struct SVR_CONFIG_ACK {
   u32 src_serial_no;       //来包序列号
   u8 status;               //0 正常
                            //1 异常
};
```

注：

2.33. 设置电池保护参数

命令：

```
#define CMD_SVR_PROTECTPARAM_SET      0x1024
struct SVR_PROTECTPARAM_SET {
```

```
    u16 item_less_voltage;          //单体欠压保护限值
    u16 item_less_voltage_release; //单体欠压恢复限值
    u8 charge_low_temp;             //充电低温保护值
    u8 charge_low_temp_release;     //充电低温保护释放值
};
```

应答：

```
#define CMD_BATTERY_PROTECTPARAM_ACK     0x4024
struct BATTERY_PROTECTPARAM_ACK {
    u32 src_serial_no;        //来包序列号
    u8 status;                //0 正常
                              //1 异常
};
```

2.34. 结果上报电池保护参数

命令：

```
#define CMD_BATTERY_PROTECTPARAM_RSLT    0x1025
struct BATTERY_PROTECTPARAM_RSLT{
    u32 src_serial_no;        //来包序列号
    u16 item_less_voltage;          //单体欠压保护限值
    u16 item_less_voltage_release; //单体欠压恢复限值
    u8 charge_low_temp;             //充电低温保护值
    u8 charge_low_temp_release;     //充电低温保护释放值

};
```

应答：

```
#define CMD_SVR_PROTECTPARAM_ACK      0x4025
struct SVR_PROTECTPARAM_ACK {
    u32 src_serial_no;        //来包序列号
    u8 status;                //0 正常
                              //1 异常
};
```

3.可靠性

3.1. 在线状态及时侦知

服务端根据 hello 包主动判断设备是否在线。当 APP 端操作设备不在线的时候给出正确提示，当 APP 端显示设备在线，但数据到达服务器后设备短线的情况，服务器给出正确的反馈。

3.2. 包序号对应机制

由于本协议是异步协议，在正式应答时需要就应答包与请求/设置包之间的关系。

3.3. 重发机制

本协议规定谁发出，谁负责数据可靠送达。
没有收到确切回包，则传输不认定为成功。规定时间内未收到反馈的包，发送端需要做重发处理，重发一定时间后依然不成功，需要做错误处理。

4.安全性

4.1. AES 加解密

设备与服务器通讯采用密文传输，加密方式选用 AES 加密。每设备使用自己的密钥。
AES 的类型：ECB/pkcs5padding,128 位秘钥长度。
注意事项：

- 秘钥中使用数字和小写字母，不允许用大写字母
- 数据包加密的时候从 cmd_data 开始加密，trans_length 是加密后的数据长度，解密也是也一样。
- 为了验证 AES 解密后数据是否正确，加密时需要先计算原文（从 cmd_data 往后的数据）的 CRC32 值，并将这个 CRC32 值放到原文的前面组成新的原文，作为加密的原始输入。解密时，用密文解出 CRC32 和原文，并且计算原文的 CRC32 值，和收到的 CRC32 值对比是否相同，相同说明解密正确，不同说明解密错误。

4.2. CRC 校验

用于校验数据包是否完整

4.3. Serial_no 运用规则

Serail_no 规定为不断增大，如果设备收到的 Serail_no 不大于之前的 serail_no，则认定为不安全。
更新密钥时 serial_no 重设起始值

4.4. 随机数校验

更新密钥时使用随机数交互验证，如果随机数与解密的数据不一致，电池端认为不安全。