充电站通信协议

版本信息

版本	修订日期	修订人	版本说明
1.0.0	2018.02.28	japy	初始版本

概述

本协议规范了充电站到后台之间的通信协议。充电站跟后台采用MQTT协议进行消息传输，每条消息采用protobuffer数据格式进行封装。

充电站跟后台之间的消息通信需要MQTT服务器进行转发，充电站和后台都作为MQTT客户端登陆到MQTT服务器，通过本协议协定的发布和订阅主题进行通信。

每条消息被称为应用协议数据单元APDU。

应用协议数据单元APDU

APDU格式如下表定义。因为大部分消息需要一应一答，消息序列号用于消息请求和应答的同步，每条不同的请求消息，要保证其消息序列号不一样，而某条请求消息对应的应答消息的消息序列号，应该与该请求消息的消息序列号一样。

APDU消息定义：

```
message APDU
{
    MessageID messageId = 1;     ///消息代码,参考 MessageID 定义
    int32 sequenceId = 2;        ///定义消息的唯一ID，应答报文中此字段必须与请
求报文中的一致，用来匹配请求和应答
    bytes payload = 3;           ///消息经protocol buffer序列化后的数据，实
际的消息内容
    int32 timestamp = 4;         ///消息unix时间戳
}
```

消息代码定义

消息	消息名	消息代码	消息类型	发送方
启动通知请求	BootNotificationReq	0x01	命令	充电站
远程设置参数请求	ChangeConfigurationReq	0x03	命令	后台
获取设备参数请求	GetConfigurationReq	0x04	命令	后台
远程控制请求	RemoteControlReq	0x05	命令	后台
远程升级请求	UpdateFirmwareReq	0x06	命令	后台
远程升级状态通知请求	FirmwareStatusNotificationReq	0x07	命令	充电站
获取诊断日志请求	GetDiagnosticsReq	0x08	命令	后台
诊断日志状态通知请求	DiagnosticsStatusNotificationReq	0x09	命令	充电站
注册请求	DeviceRegistrationReq	0x0A	注册	充电站
远程启动充电请求	RemoteStartTransactionReq	0x10	命令	后台
远程停止充电请求	RemoteStopTransactionReq	0x11	命令	后台
充电开始通知请求	StartTransactionReq	0x12	命令	充电站
充电结束通知请求	StopTransactionReq	0x13	命令	充电

				站
鉴权请求	AuthorizeReq	0x14	命令	充电站
预约充电请求	ReserveNowReq	0x15	命令	后台
取消预约请求	CancelReservationReq	0x16	命令	后台
更新用户卡缓存请求	UpdateIdCardCacheReq	0x1A	命令	后台
遥测请求	TelemetryReq	0x21	遥测	充电站
充电过程信息请求	ChargingInfoReq	0x23	遥测	充电站
触发消息重发请求	TriggerMessageReq	0x24	命令	后台
获取离线订单请求	GetTransactionsReq	0x30	命令	后台
上报离线订单请求	TransactionReq	0x31	命令	充电站
获取告警纪录请求	GetWarningReq	0x32	命令	后台
上报告警请求	WarningReq	0x33	命令	充电站
设置计费模版请求	SetTariffReq	0x34	命令	双向
启动通知应答	BootNotificationConf	0x81	命令	后台
				充

远程设置参数应答	ChangeConfigurationConf	0x83	命令	电站
获取设备参数应答	GetConfigurationConf	0x84	命令	充电站
远程控制应答	RemoteControlConf	0x85	命令	充电站
获取诊断日志应答	GetDiagnosticsConf	0x88	命令	充电站
注册应答	DeviceRegistrationConf	0x8A	注册	后台
远程启动充电应答	RemoteStartTransactionConf	0x90	命令	充电站
远程停止充电应答	RemoteStopTransactionConf	0x91	命令	充电站
充电开始通知应答	StartTransactionConf	0x92	命令	后台
充电结束通知应答	StopTransactionConf	0x93	命令	后台
鉴权应答	AuthorizeConf	0x94	命令	后台
预约充电应答	ReserveNowConf	0x95	命令	充电站
取消预约应答	CancelReservationConf	0x96	命令	充电站
更新用户卡缓存应答	UpdateIdCardCacheConf	0x9A	命令	充电站

获取离线订单应答	GetTransactionsConf	0xB0	命令	充电站
上报离线订单应答	TransactionConf	0xB1	命令	后台
获取告警记录应答	GetWarningConf	0xB2	命令	充电站
设置计费模版应答	SetTariffConf	0xB4	命令	充电站
错误应答	MessageError	0xFF	命令	双向

消息及类型定义

所有的消息及类型定义详细见 charger.proto文件

MQTT主题

MQTT协议规定了三种消息QOS，0，1，2。本协议只用到QOS0和QOS2。在本协议所有的消息中，包含三类消息，注册消息，命令消息和遥测消息。

注册消息主题路径，QOS为2：

充电站发布路径：/U/R/[设备序列号]
充电站订阅路径：/D/R/[设备序列号]

命令消息主题路径，QOS为2:

充电站发布路径：/U/C/[设备识别号]
充电站订阅路径：/D/C/[设备识别号]

遥测消息主题路径，QOS为0:

充电站发布路径：/U/M/[设备识别号]
充电站订阅路径：/D/M/[设备识别号]

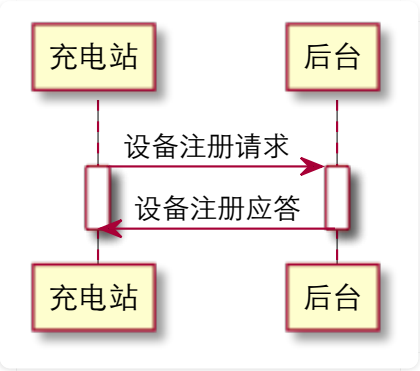
设备序列号为设备本身生产时所烧录的序列号。设备识别号是设备通过向后台注册所获取的设备在平台的身份识别号。

设备管理

设备接入到后台，需要经过以下几个步骤：
1. 在设备管理平台创建该产品型号，并获得设备注册码
2. 设备持设备注册码进行设备注册
3. 设备持设备识别号登陆后台
以上，注册码通常是某一产品型号的设备为同一个注册码，运营商获得注册码后，保存在设备端，设备注册时持注册码进行注册。
设备第一次接入后台需要进行设备注册。获取到设备识别号后，每次上电或者重连时直接登录即可。

设备注册

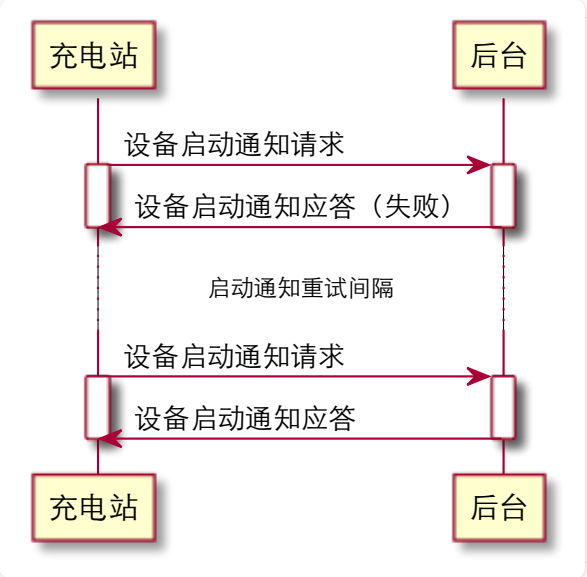
设备登陆MQTT服务器后，在进行正常的业务之前，需要取得后台的设备识别号。设备可以通过向后台发送”设备注册请求“消息获取合法的设备识别号，如果后台验证设备合法，通过”设备注册应答“消息，返回设备识别号给设备。
充电桩发布设备注册请求消息到主题路径 “/U/R/[设备序列号]” ，充电桩订阅设备注册应答消息的主题路径为 “/D/R/[设备序列号]”
设备注册时，需要填入设备型号，设备序列号，供应商代码，注册码信息。
后台验证设备合法的条件：
1. 该设备所持的注册码验证合法
2. 该设备序列号还未注册过



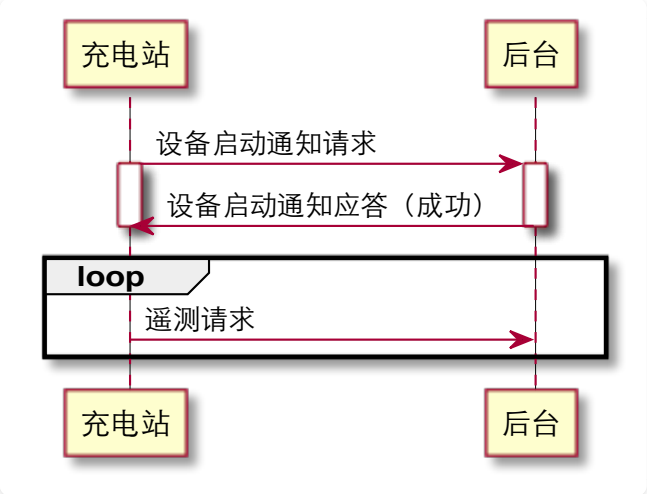
设备登陆

获得了设备识别号的设备。每次上电和重启，或者离线重连时，发送“启动通知请求”消息到后台进行登陆。中心平台根据“启动通知请求”对设备进行登陆确认，确认通过后设备和后台之间才能进行后续的消息交互。确认失败，充电站将在固定的时间间隔尝试重新登录。

1. 充电站登陆确认失败交互流程



1. 充电站登陆成功交互流程



连接保持

充电站连接到MQTT服务器后，通过MQTT协议的心跳消息，来维持设备连接状态。后台直接通过MQTT服务器获取设备的在线或离线状态。在业务消息层，不需要特别的消息来维持连接状态。

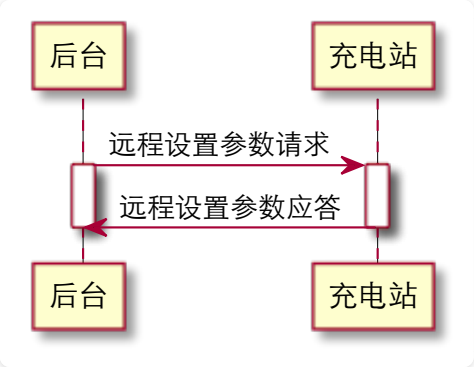
远程设置参数

后台可以通过“远程设置参数请求”消息对设备进行参数设置，支持一次性设置多个参数。每个参数为键，值对。

KeyValue

字段	类型	描述

key	string	参数名
readonly	bool	是否只读，只读参数无法修改
value	string	参数值



远程参数设置示例：

1. 设置单个参数，设置心跳间隔时间为10秒

```
{
    .configurationKey={
        .key="HeartbeatInterval",
        .value="10",
    }
}
```

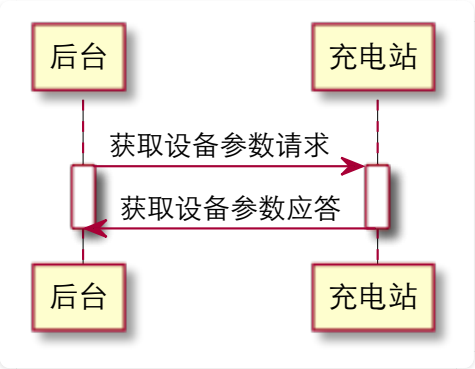
2. 设置多个参数，设置服务器地址为“test.com:1884”,用户名为“test” ，密码为“test”

```
{
    .configurationKey={
        {
            .key="ServerUrl",
            .value="test.com:1884",
        },
        {
            .key="User",
            .value="test",
        },
        {
            .key="Password",
            .value="test",
        }
```

```
        }
    }
```

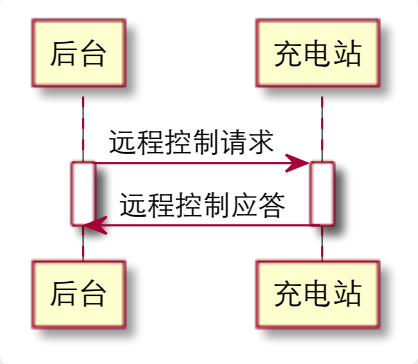
获取远程设备参数

后台可以通过“获取设备参数请求”消息获取设备的参数信息。支持一次性获取多个参数。



远程控制

后台通过“远程控制请求”对充电站进行远程控制。



标准远程控制命令

1. 远程重启
   命令名：Reset
   参数：类型int，0-重启系统，1-重启程序

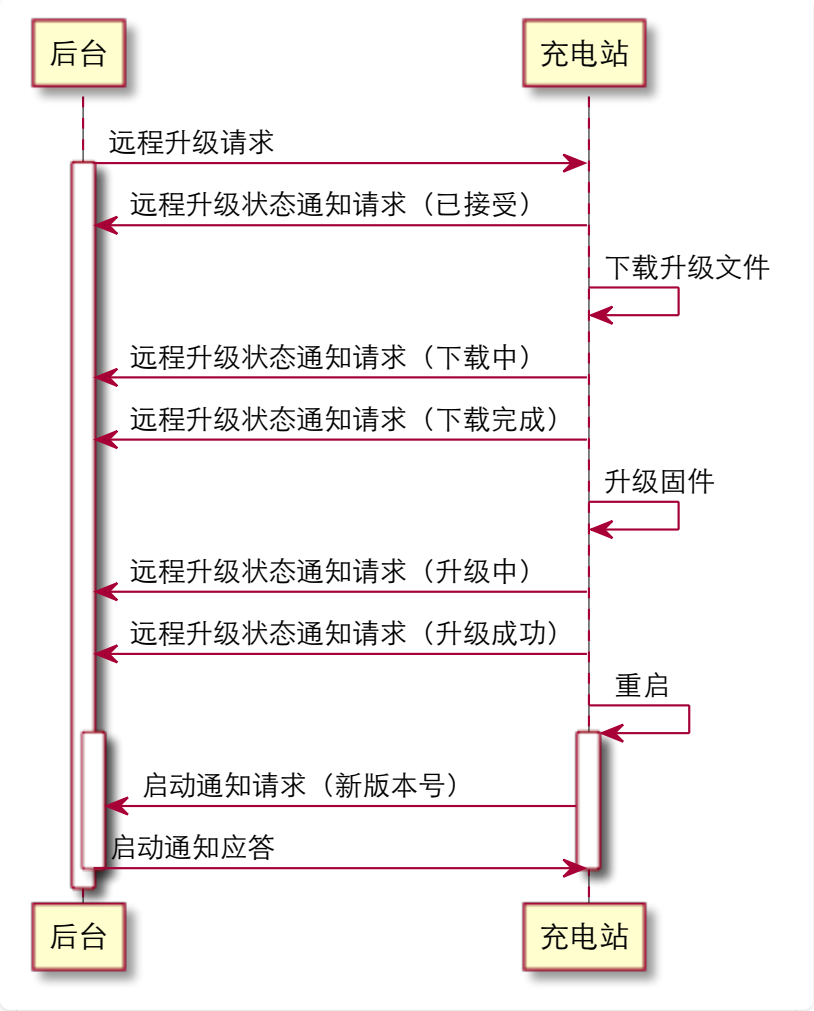
2. 开启充电
   命令名：RemoteStart
   参数：类型int，充电接口编号

3. 结束充电
   命令名：RemoteStart
   参数：类型int，充电接口编号

4. 启动停止调试
   命令名：Debug
   参数：类型int，0-关闭调试，1-开启调试

远程升级

后台通过“远程升级请求”消息，推送升级信息给充电站，充电站根据“远程升级请求”消息提供的FTP服务器地址，镜像文件名，通过ftp协议下载升级镜像文件并执行升级，充电站会校验文件的md5值。充电站通过“远程升级状态通知”消息通知升级过程中的状态。升级过程状态包括已接受升级推送，下载中，下载完成，下载失败，升级中。充电站升级成功后，将重启并发送启动通知请求，后台根据启动通知请求的版本号确认升级是否成功。



获取设备诊断日志

充电服务

充电站支持多种鉴权和支付方式发起充电。鉴权方式包括在线扫码充电，蓝牙扫码充电，刷卡充电，车辆识别充电。

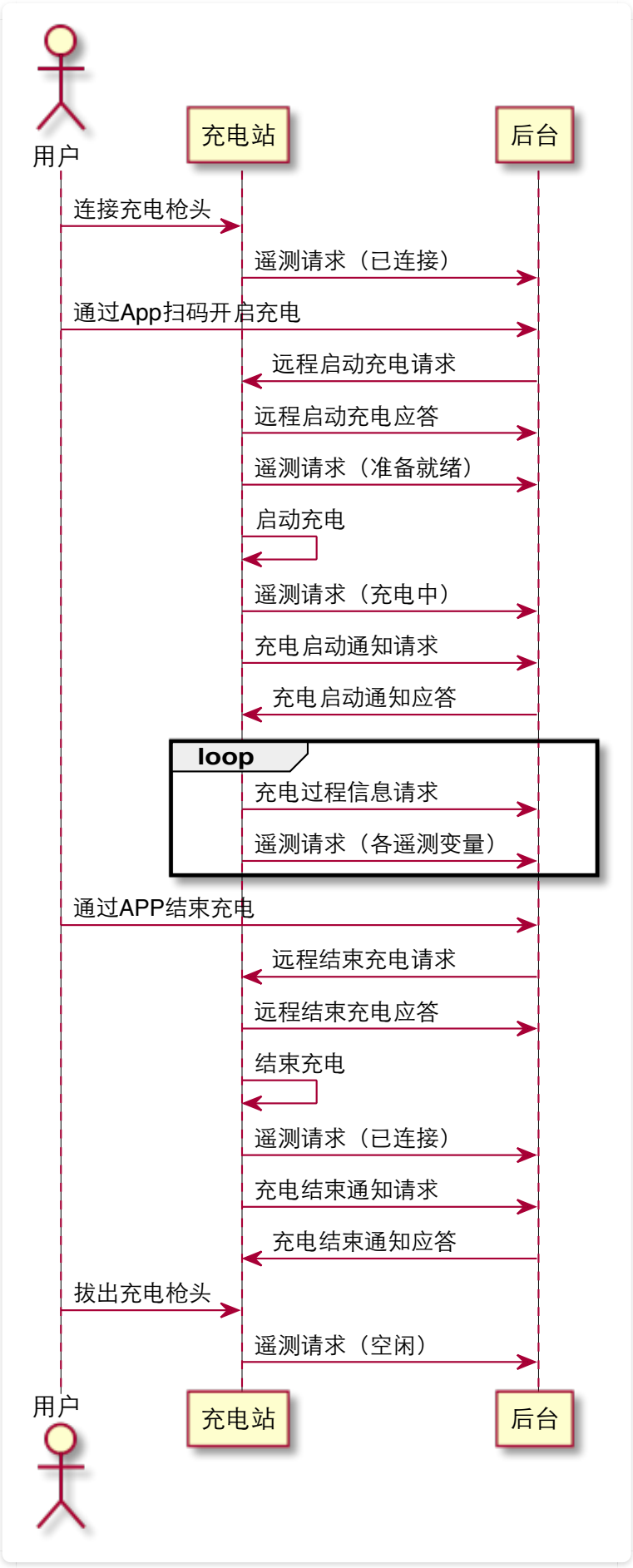
充电模式

在协议上，支持按电量，按金额，按时长，自动充满模式。在实现上，实际只需要按金额和自动充满两种模式即可。按电量和按时长归根到底是可以转换成按金额充电的。
一般而言，

在线扫码充电

用户通过APP扫充电桩上的二维码开启充电。无论是预先支付充电金额，还是充完之后再支付充电金额，充电桩和后台之间的消息交互流程是一样的。
用户扫码充电时，可能先连接好线缆再扫码启动充电，也可能先扫码打开充电接口，再连接线缆。一般对于直流充电桩，必须要先连接线缆才允许启动充电，具体APP实现时，可以做这个限制，连接好枪头之后，才允许扫码开启。
有一些充电接口可能无法判断充电线缆的连接状态，比如国标两孔 / 三孔插座，在打开充电接口前，充电接口是无法判断是否连接了充电设备的。在这种情况下，充电站可以设计成不考虑连接状态，收到远程开启充电请求，就打开充电接口。

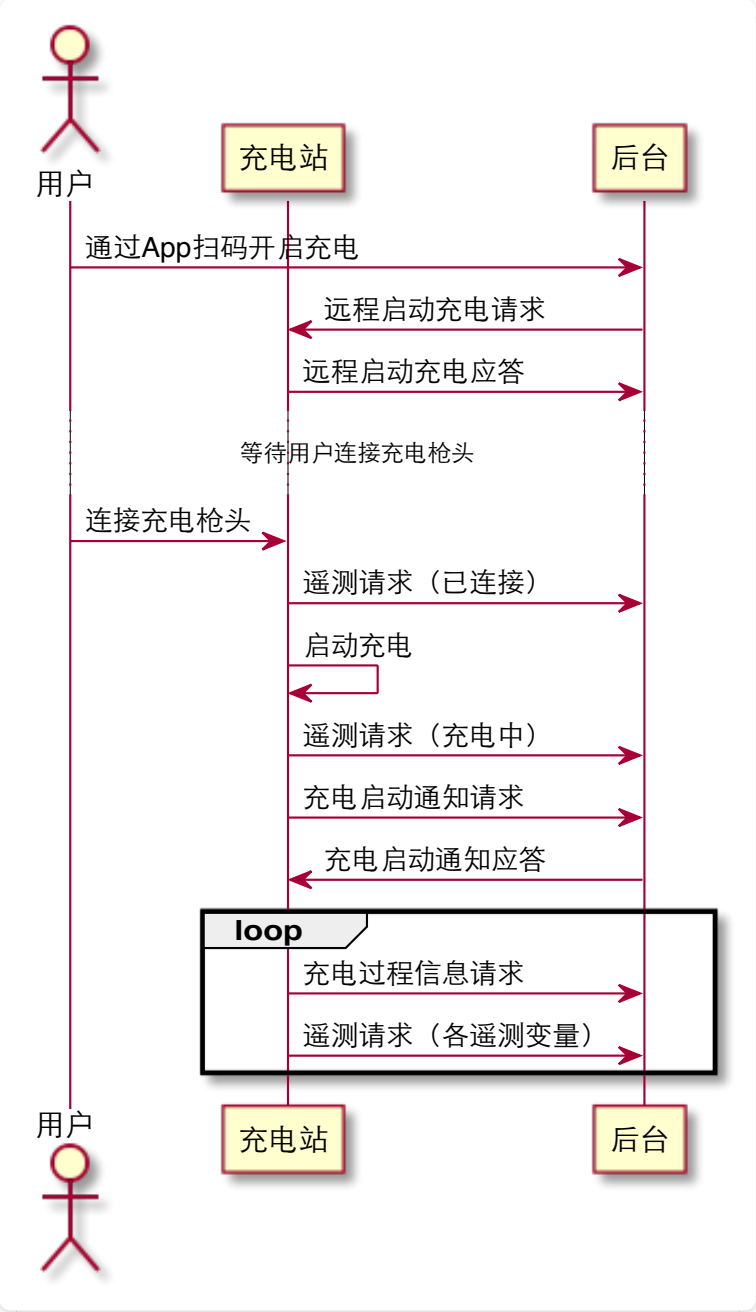
先连接枪头扫码开启充电时序图



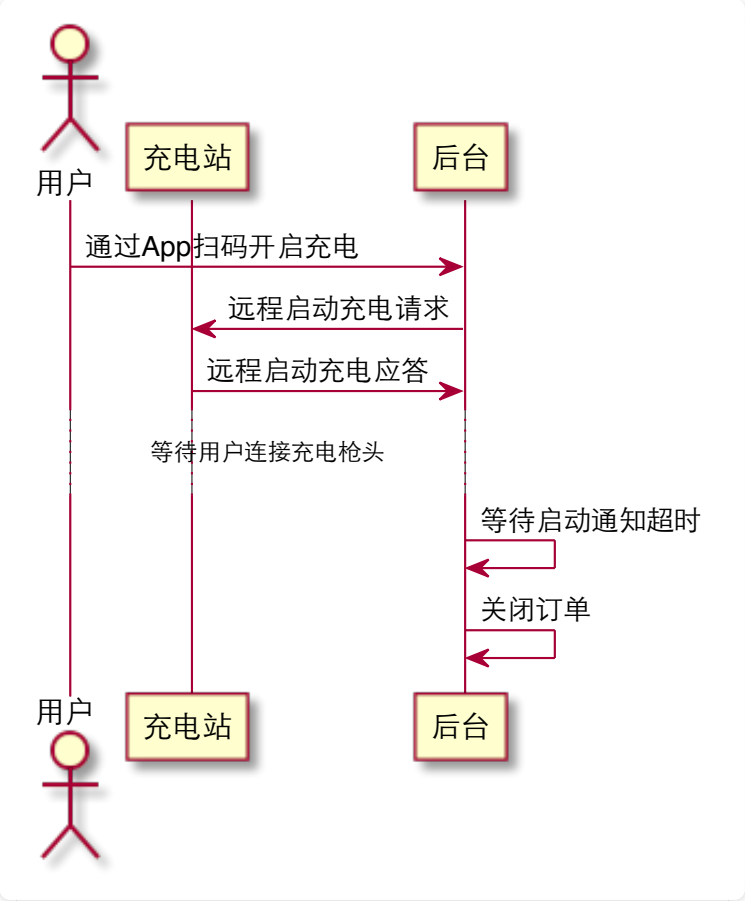
对于某些没有锁止接口的充电接口，用户可以直接拔出枪头结束充电。则上图中的远程结束充电请求及应答的流程是没有的，充电站检测到充电线缆连接断开后，直接结束充电，发送充电结束通知请求到后台。

对于充电过程中，无用户干预的正常结束（充满，预付金额已用完），故障结束。充电站停止充电后，发送充电结束通知请求到后台。

先扫码再连接枪头充电时序图



充电站和后台需要协定等待用户连接充电枪头的超时时间，超时时间到，且桩是在线的，后台仍然未收到充电启动通知请求，则本次充电结束。
等待用户连接充电枪头超时时序图：

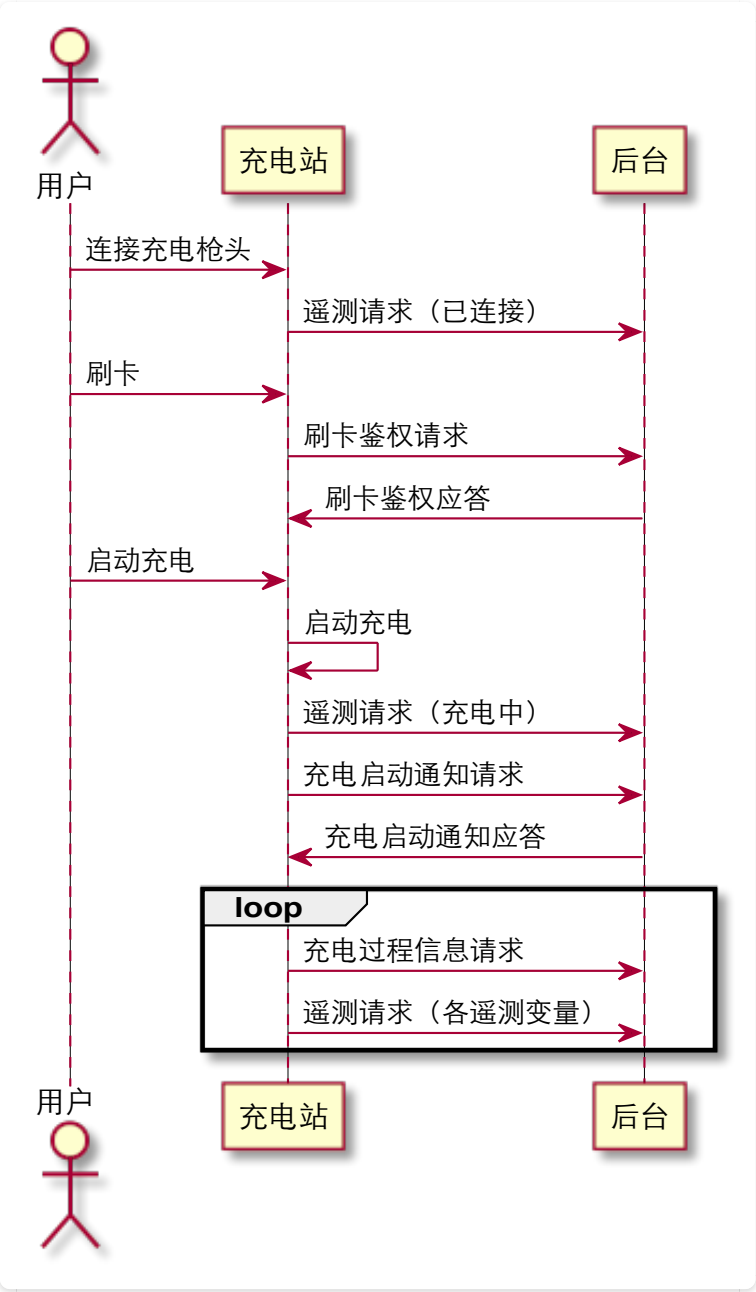


刷卡充电

这里所指的卡是指身份鉴权卡，账户余额和有效期都存储在后台。

在线鉴权

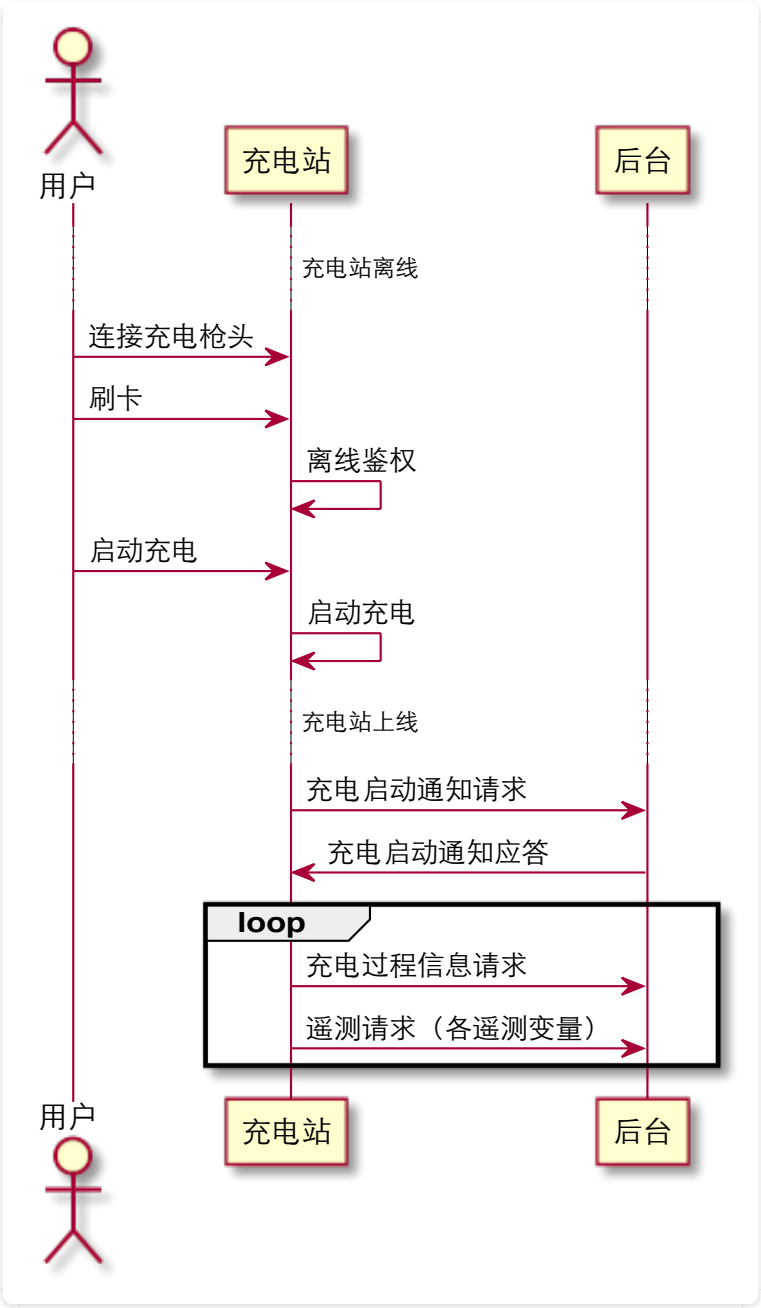
当充电站在线时，用户刷卡通过后台鉴权。



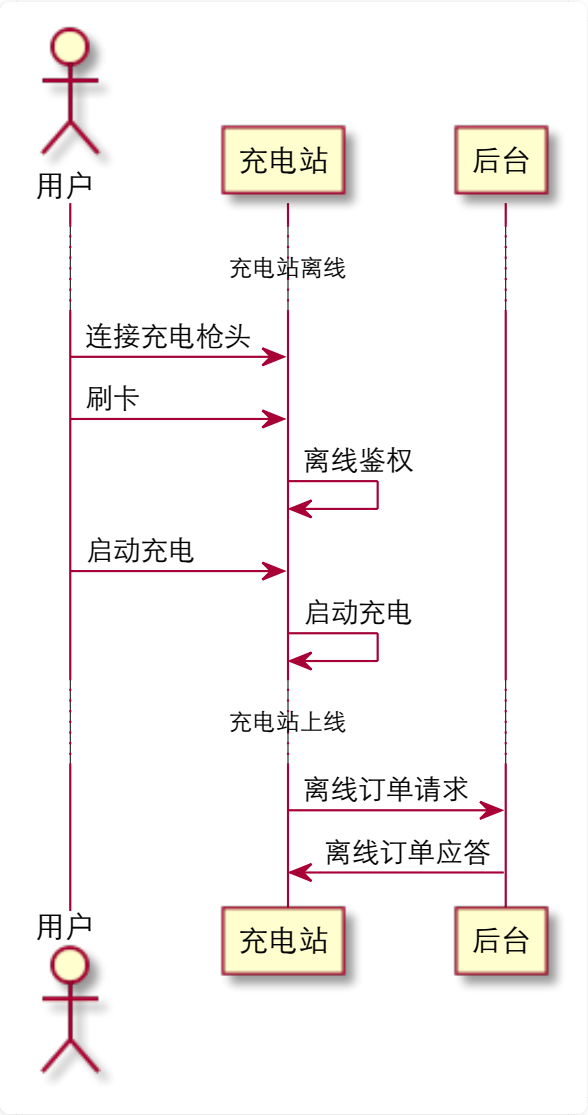
离线鉴权

当充电站离线时，用户刷卡充电时，如果充电站缓存了该卡片的鉴权信息，则可以通过本地离线进行鉴权，鉴权通过开启充电订单。当充电站再次上线时，如果订单还在充电中，充电站上报该订单的充电启动通知，如果订单已经结束，则订单将会做为一个离线订单上报后台。
有两种途径更新本地用户卡鉴权缓存。一是每次用户刷卡时，通过“刷卡鉴权请求”更新，二是后台通过“更新用户卡缓存请求”进行更新。

充电站重新上线时订单未结束



充电站重新上线时订单已结束



离线订单及离线问题

充电站可能在任何环节离线，后台需要考虑每条消息的同步和超时问题。就充电订单而言，完整的充电订单消息交互包括，启动充电，充电开始通知，充电过程信息，结束充电请求，充电结束。每一条消息都包括请求和应答。充电站可能在这些消息交互之间的任何时候离线。
约定：

1. 当充电站离线后重新上线时，所有还在进行中的充电订单，充电站重新发送“充电开始通知请求”，后台应答“充电开始通知应答”后，充电站才发送后续的充电过程信息。
2. 当充电站离线重新上线时，发送离线订单请求到后台
3. 充电站未收到后台“充电结束通知应答”的订单，视为离线订单

计费模版

每个计费模版都有计费模版ID号，后台和充电站通过计费模版ID号区分不同的计费模版。设

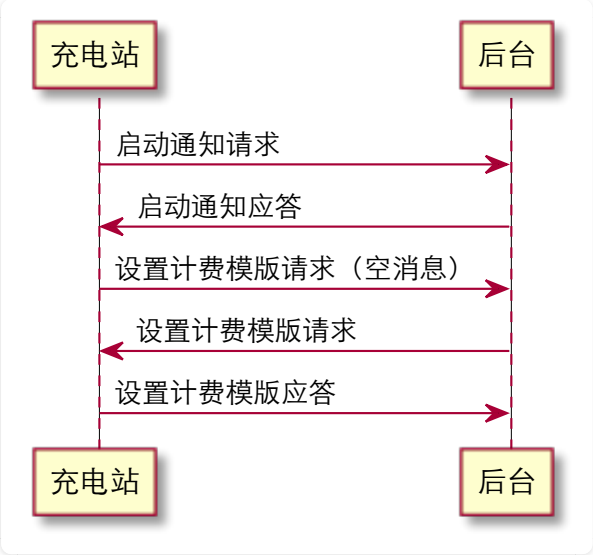
置计费模版消息定义如下：

```
message SetTariffReq
{
    int32 tariffid = 1;                          ///计费模版id号
    string description = 2;                      ///描述
    repeated TariffCharge chargetariffs = 3;     ///按电量充电费用
    repeated TariffTime timetariffs = 4;         ///按时长充电费用
    repeated TariffParking parkingtariffs = 5;   ///停车费
}
```

一般而言，对于充电站，只支持上述计费模版中的，“按电量计费”，“按时长计费”，“停车费”中的一种。

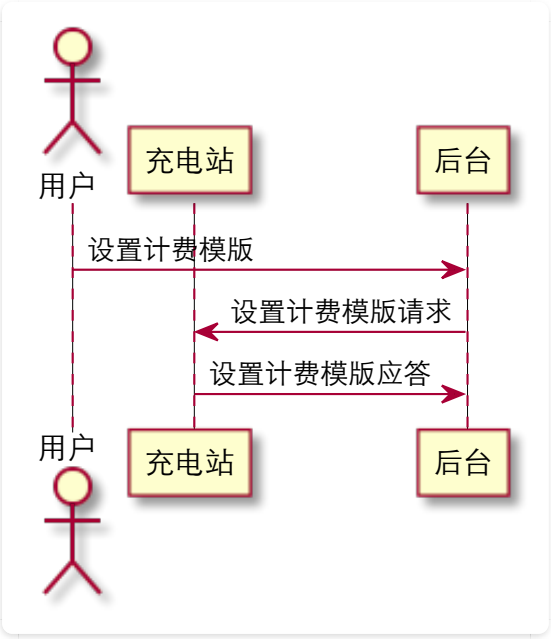
有两种方式更新充电站的计费模版：

1. 充电站发送“设置计费模版请求”主动请求更新计费模版，一般情况下，可以在每次登陆后发送



   以上，充电站向后台发送“设置计费模版请求”消息时，APDU的消息ID为“设置计费模版请求”，payload为空

2. 用户在后台设置更新计费模版



按电量计费

按电量计费的充电服务费可能包含三种费用，电费，服务费，按时长占用费（预约，延时等），支持分时段电价计费。按电量计费模版定义如下：

```
message TariffCharge{
    int32 timestart = 1;          ///分段费率开始时间，以24小时内分钟数表示。例如420表示此费率从7:00（420/60）开始
    int32 timeend = 2;            ///分段费率结束时间，同上
    string tag = 3;               ///分段标签，比如"尖"，"峰"，"平"，"谷"
    int32 elecprice = 4;          ///按度电价，分辨率0.1RMB分
    int32 serviceprice = 5;       ///按度服务费价格，分辨率0.1RMB分
    int32 occupyprice = 6;        ///按小时占用费用
}
```

计费模版举例：

1. 电费0.8元，服务费0.8元

```
{
    .tariffid=1,
    .TariffCharge={
        .elecprice=800,
        .serviceprice=800,
    }
}
```

2. 峰期时段，9:00-11:30，14:00-16:30，19:00-21:00，电价1.052，服务费1.00
   平期时段，7:00-9:00，11:30-14:00，16:30-19:00，21:00-23:00，电价0.699，服务费1.00
   谷期时段，23:00-次日7:00，电价0.255，服务费1.00

```
{
    .tariffid=2,
    .TariffCharge={
        {
            .timestart=0,
            .timeend=420,
            .tag="valley",
            .elecprice=255,
            .serviceprice=1000,
        },
        {
            .timestart=420,
            .timeend=540,
            .tag="flat",
            .elecprice=699,
            .serviceprice=1000,
        },
        {
            .timestart=540,
            .timeend=690,
            .tag="tip",
            .elecprice=1052,
            .serviceprice=1000,
        },
        {
            .timestart=690,
            .timeend=840,
            .tag="flat",
            .elecprice=699,
            .serviceprice=1000,
        },
        {
            .timestart=840,
            .timeend=990,
            .tag="tip",
            .elecprice=1052,
            .serviceprice=1000,
        },
        {
            .timestart=990,
```

```
            .timeend=1140,
            .tag="flat",
            .elecprice=699,
            .serviceprice=1000,
        },
        {
            .timestart=1140,
            .timeend=1260,
            .tag="tip",
            .elecprice=1052,
            .serviceprice=1000,
        },
        {
            .timestart=1260,
            .timeend=1380,
            .tag="flat",
            .elecprice=699,
            .serviceprice=1000,
        },
        {
            .timestart=1380,
            .timeend=1440,
            .tag="valley",
            .elecprice=255,
            .serviceprice=1000,
        },
    }
}
```

按时长计费

按充电时长进行计费，支持按不同功率不同价格计费，以及支持按不同时段不同价格计费。计费模版定义如下：

```
message TariffTime{
    int32 powerstart = 1;        ///分段费率起始功率，分辨率W
    int32 powerend = 2;          ///分段费率结束功率，分辨率W
    int32 timestart = 3;         ///分段费率开始时间，以24小时内分钟数表示。例如420表示此费率从7:00（420/60）开始
    int32 timeend = 4;           ///分段费率结束时间，以24小时内分钟数表示。例如480表示此费率到8:00（480/60）结束
    int32 price = 5;             ///每小时充电价格，分辨率0.1分。
}
```

计费模版举例：

1. 1元 / 4小时

```
{
    .tariffid=11,
    .TariffTime={
        .price=250,
    }
}
```

2. 0 ~ 200w 1元 / 4小时，200w ~ 400w 1元 / 3小时，400w以上 1元 / 2小时

```
{
    .tariffid=12,
    .TariffTime={
        {
            .powerstart=0,
            .powerend=200,
            .price=250,
        },
        {
            .powerstart=200,
            .powerend=400,
            .price=333,
        },
        {
            .powerstart=400,
            .powerend=9999,
            .price=500,
        }
    }
}
```

设备监控

设备通过充电过程信息请求上报充电接口充电中信息，通过遥测请求上报充电站状态变化及周期性测量值，通过上报告警纪录请求消息上报充电站告警信息。

充电过程信息

当充电接口启动充电后，处于充电中状态的订单，以固定的间隔时间，上报充电过程信息。通常，对于交流充电，充电过程信息仅包含以下信息：

1. 充电纪录号（桩端记录号）
2. 充电接口号
3. 已充时间
4. 已充电量
5. 消费金额
6. 充电功率
7. 剩余充电时间

对于直流充电，包含了更多跟BMS交互的信息：

1. BMS充电握手阶段信息
2. BMS充电参数配置阶段信息
3. BMS充电中交互信息
4. BMS充电结束信息

遥测

充电站遥测请求消息定义如下：

```
message TelemetryReq
{
    Components component = 1;  ///组件
    int32 connectorId = 2;     ///充电接口，若测量组件属于某个充电接口，如某个接口的电表，填对应的充电接口号
    repeated SampledValue values = 5;   ///遥测值集合
    uint64 warningBitmap = 6;  ///故障告警位表，此组件当前所有告警位标志
    int32 recordId = 7;        ///相关联充电订单号
}
```

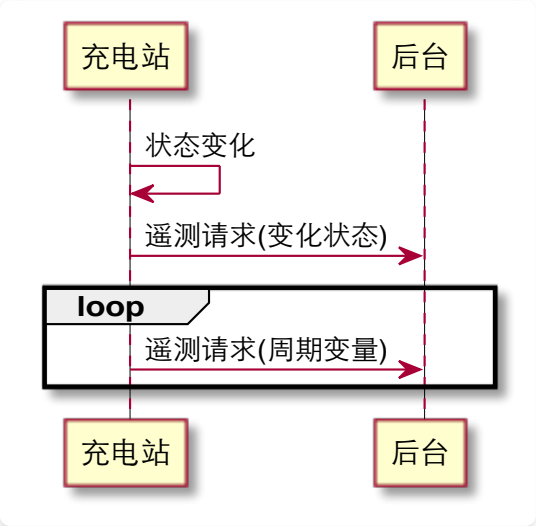
其中的values为遥测值集合，values为SampledValue类型。SampledValue类型定义如下：

```
message SampledValue
{
    int32 measurand = 1;            ///遥测变量代码
    string vendorEx = 2;            ///供应商扩展遥测变量名
```

```
    int32 value = 3;                ///遥测值
}
```

对于协议标准所支持的遥测变量，measurand直接填入标准的遥测变量代码即可。遥测变量代码在“measure.proto”文件中定义。如果是非协议标准所支持的遥测变量，通过vendorEx填入遥测变量名。后台可以通过遥测变量名知道所遥测的变量是什么。

遥测请求消息时序图



当充电站某些状态发生变化时，通过遥测请求消息上报状态。所有充电站应该包含以下两个变化状态：

1. 可用充电接口数量
   例如上报可用充电接口数量为2

```
{
    .component=CP_None,
    .values={
        .measurand=ConnectorAvailable,
        .value=2,
    }
}
```

2. 充电接口状态
   例如上报充电接口1的充电接口状态为空闲

```
{
    .component=CP_Connector,
    .values={
```

```
            .measurand=ConnectorStatus,
            .value=CHS_Available,
        }
    }
```

周期性上报的遥测变量信息。周期性上报的遥测变量视具体的充电站所支持的遥测变量而定。以下是几个例子：

1. 上报充电接口1，温度 = 30度，电压 = 220.0V，电流 = 2.0A，电量 = 5.01kwh，功率 = 320.10w，接触器状态 = 开

```
{
    .component=CP_Connector,
    .connectorId=1,
    .values={
        {
            .measurand=Temperature,
            .value=30,
        },
        {
            .measurand=Voltage,
            .value=2200,
        },
        {
            .measurand=Current,
            .value=20,
        },
        {
            .measurand=Electricity,
            .value=501,
        },
        {
            .measurand=PowerWatts,
            .value=32010,
        },
        {
            .measurand=ContactorStatus,
            .value=1,
        },
    }
}
```

2. 系统遥测，RSSI移动信号值为30

```
{
    .component=CP_System,
    .values={
        .measurand=RSSI,
        .value=30,
    }
}
```

告警

当桩端触发告警时，发送告警消息到后台。标准告警代码在“warning.proto”的文件中定义