点点畅行道闸推送接口文档

1	2023年6月28日v0.1版本创建
2	2024年2月20日v0.2 新增流水号与优惠时长
3	2024年6月19日v0.3 新增电量

1接口约定

1.1访问方式

1.1.1协议

接口服务与用户的通信协议，使用HTTP协议，使用POST方式进行交互。

1.1.2接收地址

https：域名/ddcx/{version}/barrierGatePush

由对接方提供域名，点点畅行平台给予配置

1.1.3版本号

为保证API兼容性，将接口服务的版本号version放入URL

1.2密钥

密钥由点点畅行平台发放，两端共同使用

1.3数据格式

接口服务与客户端之间通信的数据采用JSON格式进行交互，JSON数据的属性见具体接口定义。

1.4响应数据格式

字段	描述
code	1成功、0失败
msg	返回信息

2接口签名说明

2.1签名生成说明

将待推送数据按照字段名的ASCII码从小到大排序，并和对应的值进行拼接，最后加上密钥，进行MD5加密，如：

```
key1=value1&key2=value2&key=9dkasj7jgh
```

2.2签名校验说明

接收到数据分为两部分，data和sign，对data数据进行提取后，按照2.1签名生成说明得到的字符串和sign（大写）进行校验，一致则通过

3.推送数据说明

序号	参数名	类型	说明
1	parkingId	String	停车场id
2	parkingName	String	停车场名称
3	carLicense	String	车牌
4	stationId	int	站点id
5	startTime	String	开始时间
6	endTime	String	结束时间
7	deviceId	int	充电桩id
8	deviceNumber	String	充电桩编码
9	electricMoney	int	电费单位：分
10	serviceMoney	int	服务费单位：分
11	serialNumber	String	流水号
12	discountDuration	int	优惠时长：分钟
13	electric	String	电量单位：0.01度

4签名校验&加密代码示例

4.1签名校验

```
    /**
     * 校验签名是否正确
     * @param data  data数据
     * @param pushKey 推送密钥
     * @param sign sign 数据
     * @return 签名校验结果
     */
    public static boolean checkSign(Map<String,Object> data, String pushKey,
String sign)
```

```
    {
        boolean bo=false;
        List<String> keys = new ArrayList<String>(data.keySet());
        Collections.sort(keys);
        StringBuilder preStr = new StringBuilder();
        for (int i = 0; i < keys.size(); i++) {
            String key = keys.get(i);
            Object value = data.get(key);
            if (!StringUtils.isEmpty(value))
            {
                if (value!="sign")
                {
                    preStr.append(key).append("=").append(value).append("&");
                }
            }
        }
        preStr.append("key=").append(pushKey);
        //MD5
        String str= EncryptUtils.Encrypt(preStr.toString()).toUpperCase();
        if (str.equals(sign))
        {
            bo=true;
        }
        return bo;
    }
```

4.2加密

4.2.1 JAVA加密

```
    /**
     * MD5加密
     * @param original 初始字符串
     * @return 加密后字符串
     */
    public static String Encrypt(String original) {
        byte[] bytes = null;
        try {
            MessageDigest digest = MessageDigest.getInstance("MD5");
            bytes = digest.digest(original.getBytes(StandardCharsets.UTF_8));
            if (bytes != null && bytes.length > 0) {
                StringBuilder sb = new StringBuilder();
                for (byte b : bytes) {
                    sb.append(Integer.toHexString(b & 0xFF));
                }
                return sb.toString();
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return null;
    }
```

4.2.2 .NET加密

```
/**
 * Java 和 .net 通用MD5加密
 * @param origin
 * @return
 */
public static String GeneralMD5Encrypt(String origin)
{
    StringBuffer sb = new StringBuffer();
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(origin.getBytes(StandardCharsets.UTF_8));
        byte[] result = md.digest();
        for (int i = 0; i < result.length; i++) {
            int val = (result[i] & 0x000000ff) | 0xffffff00;
            sb.append(Integer.toHexString(val).substring(6));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return sb.toString();
}
```