目录

一、身份认证以及加解密方式

密钥的产生和体系

产生

数据密钥应具备随机产生特性，密钥产生后要检查密钥的有效性，弱密钥和半弱密钥需被剔除。

运营商加入信息交换时，必须申请独立的密钥文件，密钥可由运营商协商产生。

体系

每个运营商与平台交互前需要分配平台标识 (OperatorID)、平台密钥 (OperatorSecret)、消息密钥（DataSecret）、消息密钥初始化向量（DataSecretIV）和签名密钥（SigSecret）。

1)平台标识（OperatorID）：固定9位（在【企业查】等网站上可以找到），运营商的组织机构代码，作为运营商的唯一标示。

2) 平台密钥 (OperatorSecret):固定16位，可采用32H、48H 和64H，由 0-F 字符组成，为申请认证使用。

3)消息密钥（DataSecret）:固定16位，用于对所有接口中 Data 信息进行加密。

4)消息密钥初始化向量（DataSecretIV）： 固定16位，用户 AES 加密过程的混合加密。

5)签名密钥（SigSecret）：固定16位，可采用32H、48H 和64H，由 0-F 字符组成，为签名的加密密钥。

平台认证方式及规则

平台认证方法

平台认证宜采取身份认证和访问控制相结合的方式进行。

身份认证可采取用户名/口令认证、密钥认证或数字证书认证等方式进行；访问控制可采取 IP 访问控制、时间访问控制等多种手段结合。

用户身份认证成功后授予 Token，每次向服务端请求资源的时候需要带着服务端签发的 Token，服务端验证 Token 成功后，才返回请求的数据。Token 的有效期由服务方确定，最长不应超过7天，Token 丢失或失效后需要再次发起认证服务。

图 2　平台认证方式

数据传输方式及规则

接口调用方式

所有接口均使用 HTTP(S)/POST 方式传输参数，传输过程中应包含消息头和消息主体两部分。

消息头规范

消息头一般需包含内容类型和授权信息（Authorization）。

内容类型（Content-Type）字段用于标识请求中的消息主体的编码方式，本标准中所规范的信息交换内容均采用 JSON 的方式，参数信息采用 utf-8编码，因此需要配置消息头中的 Content-Type 为 application/json;charset=utf-8。

授权信息（Authorization）字段用于证明客户端有权查看某个资源，本标准中所规范的授权信息采用凭证（Token）的方式，因此需要在配置消息头中的 Authorization 为 Bearer Token。

请求参数规则

一般由运营商标识（OperatorID）、参数内容（Data）JSON 串、时间戳（TimeStamp）、自增序列（Seq）和数字签名（Sig）组成。

表 1　请求消息主体内容表

参数名	说明	举例	是否必填
OperatorID	运营商标识		
Data	各接口具体加密后的参数信息	8TRL3nAAPQyKFp7a+XaOZ+K+WmpSN2uf9m+ZYZawnwtSGIJdL2Fg2PyZPHCM+3UZUqC43vBdmWSrkFBMUDLm0g==	是
TimeStamp	时间戳	接口请求时时间戳信息,格式为 yyyyMMddHHmmss	是
Seq	自增序列	4 位自增序列取自时间戳，同一秒内按序列自增长，新秒重计。如 0001	是
Sig	参数签名	加签参数为 OperatorID+Data+TimeStamp+Seq	是

返回参数规则

数据传输接口的返回参数一般由返回值（Ret）、返回信息（Msg）、参数内容（Data）和数字签名（Sig）组成。

1)Ret:必填字段，返回编码参考下表。

2)Msg:必填字段，有错误表示具体错误信息，无错误返回成功信息。

3)Data:必填字段，参数内容,采用 utf-8编码，JSON 格式。

表 2　返回消息主体内容表

参数名	说明	举例	是否必填
Ret	返回参数编码		是
Msg	返回消息字符串		是
Data	业务加密数据	hrJar2kPUP7p1JvXmPvyJ7hBe21jHG+j6x8hWxFNG6+Z4WxYRhKtSb7wybOixfacVc3aDriNAvwlQjJS91LNzCOgSI8Hb57uPQ5pFmXcHvJIZP5qvxqA2HDXQVyJEovV5Lfoe7i/JdCYu8L3Gh/2AirwvFK2KQCrfYgR8phCws4q1Omc6RRnKQJdYchpi6pctewccNEdQB2sh0Yxm6imMxlCEsa0J6NZJdxBunpyNoY=	是
Sig	签名	Ret+Msg+Data	是

表 3　返回参数编码表

Ret 值	说明
-1	系统繁忙，此时请求方稍后重试
0	请求成功
4001	签名错误
4002	Token 错误
4003	POST 参数不合法,缺少必须的示例：OperatorID,sig,TimeStamp,Data，Seq 五个参数
4004	请求的业务参数不合法，各接口定义自己的必须参数
500	系统错误

批量数据传输

数据传输接口中的 Data 字段可为数组型的 JSON 格式，数据发送方可通过该字段实现批量数据的传输。

数据加解密规则

消息发送方需要对 Data 字段中涉及交易及隐私等数据利用消息密钥（DataSecret）进行加密，加密算法宜使用 AES 128位加密，加密模式采用 CBC，填充模式采用 PKCS5Padding 方式。

消息接收方收到消息之后，根据消息密钥（DataSecret）对消息体中的 Data 数据进行解密，校验参数合法性等后续业务处理。

数据加/解密示例（密钥、向量不作为固定值）

密钥：1234567890abcdef

初始向量：1234567890abcdef

明文信息：

示例：{"total":1,"stationStatusInfo":{"operationID":"123456789","stationID":"111111111111111","connectorStatusInfos":{"connectorID":1,"equipmentID":"10000000000000000000001","status":4,"currentA":0,"currentB":0,"currentC":0,"voltageA":0,"voltageB":0,"voltageC":0,"soc":10,}}}

秘文：

示例：il7B0BSEjFdzpyKzfOFpvg/Se1CP802RltKYFPfSLRxJ3jf0bVl9hvYOEktPAYW2nd7S8M
BcyHYyacHKbISq5iTmDzG+ivnR+SZJv3USNTYVMz9rCQVSxd0cLlqsJauko79NnwQ
JbzDTyLooYoIwz75qBOH2/xOMirpeEqRJrF/EQjWekJmGk9RtboXePu2rka+Xm51sy
BPhiXJAq0GfbfaFu9tNqs/e2Vjja/ltE1M0lqvxfXQ6da6HrThsm5id4ClZFIi0acRfrsPLRix
S/IQYtksxghvJwbqOsbIsITail9Ayy4tKcogeEZiOO+4Ed264NSKmk7l3wKwJLAFjCFog
Bx8GE3OBz4pqcAn/ydA=

参数签名规范

参数签名要求

参数签名采用 HMAC-MD5算法，采用 MD5 作为散列函数，通过签名密钥（SigSecret）对整个消息主体进行加密，然后采用 Md5信息摘要的方式形成新的密文，参数签名要求大写。

请求参数的签名顺序按照消息体顺序拼接后执行，拼接顺序为运营商标识（OperatorID）、参数内容（Data）、时间戳（TimeStamp）、自增序列（Seq）。

返回参数的签名顺序按照消息体顺序拼接后执行，拼接顺序为返回值（Ret）、返回信息（Msg）、参数内容（Data）。

参数签名方法

HMAC-MD5算法

1) 在签名密钥（SigSecret）后面添加0来创建一个长为64字节的字符串(str)；

2) 将上一步生成的字符串(str)与 ipad(0x36)做异或运算，形成结果字符串(istr)；

3)将消息内容 data 附加到第二步的结果字符串(istr)的末尾；

4)做 md5运算于第三步生成的数据流(istr)；

5) 将第一步生成的字符串(str)与 opad(0x5c)做异或运算，形成结果字符串(ostr)；

6)再将第四步的结果(istr)附加到第五步的结果字符串(ostr)的末尾；

7)做 md5运算于第六步生成的数据流(ostr)，输出最终结果(out)。

参数签名示例

签名密钥：1234567890abcdef

运营商标识（OperatorID）：123456789

参数信息（Data）：

il7B0BSEjFdzpyKzfOFpvg/Se1CP802RItKYFPfSLRxJ3jf0bVl9hvYOEktPAYW2nd7S8MBcyHYyacHK
bISq5iTmDzG+ivnR+SZJv3USNTYVMz9rCQVSxd0cLlqsJauko79NnwQJbzDTyLooYoIwz75qBOH2/xO
MirpeEqRJrF/EQjWekJmGk9RtboXePu2rka+Xm51syBPhiXJAq0GfbfaFu9tNqs/e2Vjja/ltE1M0lqvxfX
Q6da6HrThsm5id4ClZFIi0acRfrsPLRixS/IQYtksxghvJwbqOsbIsITail9Ayy4tKcogeEZiOO+4Ed264NS
Kmk7l3wKwJLAFjCFogBx8GE3OBz4pqcAn/ydA=

时间戳（TimeStamp）：20160729142400

自增序列（Seq）：0001

签名（Sig）：745166E8C43C84D37FFEC0F529C4136F

二、接口（信息交互均是 json 形式）

认证接口

此接口用于平台之间认证 Token 的申请，Token 作为全局唯一凭证，调用各接口时均需要使用。此接口也应实现加解密规范要求和验签要求。Token 的有效时间一般为7天。

接口定义

接口名称：query_token
接口使用方法：由服务端实现此接口，需求端调用。

输入参数

参数名称	定义	参数类型	描述
运营商标识	OperatorID	字符串	运营商组织机构代码
运营商密钥	OperatorSecret	字符串	运营商分配的唯一识别密钥

返回值

参数名称	定义	参数类型	描述
运营商标识	OperatorID	字符串	运营商组织机构代码
成功状态	SuccStat	整型	0:成功; 1:失败
获取的凭证	AccessToken	字符串	全局唯一凭证
凭证有效期	TokenAvailableTime	整型	凭证有效期，单位秒
失败原因	FailReason	整型	0:无; 1:无此运营商; 2:密钥错误; 3～99:自定义

到闸接口（宙晖调用道闸系统）

接口定义

方法名称：chargeorder

输入参数

字段	类型	含义	字段限制说明
parkId	String	停车场站 id	必填 0-100 字符
orderNo	String	订单号	必填 0-100 字符
plateNo	String	车牌号	必填 0-100 字符
startTime	String	充电开始时间	yyyy-MM-dd HH:mm:ss
endTime	String	充电结束时间	yyyy-MM-dd HH:mm:ss
saleValue	String	减免时长，单位为分钟	作为真正减免依据

返回值

参数名称	定义	参数类型	描述

状态	SuccStat	整型	0:成功; 1:失败
原因	FailReason	字符串	成功或失败原因

推送站点接口（宙晖调用流量方或者市政平台）

接口定义

方法名称：notification_stationInfo

输入参数

参数名称	定义	参数类型	描述
充电站信息列表	StationInfos	StationInfo[]	类型“StationInfo”

名称	字段	描述	必填	类型	长度
充电站 ID	StationID	运营商自定义的唯一编码	是	字符串	<=20 字符
运营商 ID	OperatorID	运营商 ID	是	字符串	9 字符
设备所属方 ID	EquipmentOwnerID	设备所属运营平台组织机构代码	是	字符串	9 字符
充电站名称	StationName	充电站名称的描述	是	字符串	<=50 字符
充电站国家代码	CountryCode	比如 CN	是	字符串	2 字符
充电站省市辖区编码	AreaCode	填写内容为参照 GB/T2260-2015	是	字符串	20 字符
详细地址	Address		是	字符串	<=50 字符
站点电话	StationTel	能够联系场站工作人员进行协助的联系电话	否	字符串	<=30 字符
服务电话	ServiceTel	平台服务电话，例如 400 的电话	是	字符串	<=30 字符
站点类型	StationType	1：公共 50：个人 100：公交（专用） 101：环卫（专用） 102：物流（专用） 103：出租车（专用）	是	整型	

		255：其他			
站点状态	StationStatus	0：未知 1：建设中 5：关闭下线 6：维护中 50：正常使用	是	整型	
车位数量	ParkNums	可停放进行充电的车位总数，默认：0 未知	是	整型	
经度	StationLng	GCJ-02 坐标系	是	浮点型	保留小数点后 6 位
纬度	StationLat	GCJ-02 坐标系	是	浮点型	保留小数点后 6 位
站点引导	SiteGuide	描述性文字，用于引导车主找到充电车位	否	字符串	<=100 字符
建设场所	Construction	1：居民区 2：公共机构 3：企事业单位 4：写字楼 5：工业园区 6：交通枢纽 7：大型文体设施 8：城市绿地 9：大型建筑配建停车场 10：路边停车位 11：城际高速服务区 255：其他	是	整型	
站点照片	Pictures	充电设备照片、充电车位照片、停车场入口照片	否	字符串数组	
使用车型描述	MatchCars	描述该站点接受的车大小以及类型，如大巴、物流车、私家乘用车、出租车等	否	字符串	<=100 字符
车位楼层及数量描述	ParkInfo	车位楼层以及数量信息	否	字符串	<=100 字符
营业时间	BusineHours	营业时间描述	否	字符串	<=100 字符
充电电费率	ElectricityFee	充电费描述	否	字符串	<=256 字符
服务费率	ServiceFee	服务费率描述	否	字符串	<=100 字符
停车费	ParkFee	停车费率描述	否	字符串	<=100 字符
支付方式	Payment	支付方式:刷卡、线上、现金 其中电子钱包类卡为刷卡，身份鉴权卡、微信/支付宝、APP 为线上	否	字符串	<=20 字符

是否支持预约	SupportOrder	充电设备是否需要提前预约后才能使用。0 为不支持预约、1 为支持预约。不填默认为 0	否	整型	
备注	Remark	其他备注信息	否	字符串	<=100 字符
充电设备信息列表	EquipmentInfos	该充电站所有充电设备信息对象集合	是	EquipmentInfo[]	

名称	字段	描述	必填	类型	长度/范围
设备编码	EquipmentID	设备唯一编码，对同一运营商，保证唯一	是	字符串	<=23 字符
设备生产商组织机构代码	ManufacturerID	设备生产商组织机构代码	是	字符串	9 字符
设备型号	EquipmentModel	由设备生厂商定义的设备型号	否	字符串	<=20 字符
设备生产日期	ProductionDate	YYYY-MM-DD	否	字符串	10 字符
设备类型	EquipmentType	1：直流设备 2：交流设备 3：交直流一体设备 4：无线设备 5：其他	是	整型	
充电设备接口列表	ConnectorInfos	该充电设备所有的充电设备接口的信息对象集合	是	ConnectorInfo[]	
充电设备经度	EquipmentLng	GCJ-02 坐标系	否	浮点型	保留小数点后 6 位
充电设备纬度	EquipmentLat	GCJ-02 坐标系	否	浮点型	保留小数点后 6 位
充电设备总功率	Power	单位:kW	是	浮点型	保留小数点后 1 位
充电设备名称	EquipmentName		否	字符串	<=30 字符
国家标准	NationalStandard	设备接口遵从的国家标准号	是	字符串	20 个字符

名称	字段	描述	必填	类型	长度/范围
充电设备接口编码	ConnectorID	充电设备接口编码，同一运营商内唯一	是	字符串	<=26 字符
充电设备接口名称	ConnectorName		否	字符串	30 字符
充电设备接口类型	ConnectorType	1：家用插座（模式 2） 2：交流接口插座（模式 3，连接方式 B ） 3：交流接口插头（带枪线，模式 3，连接方式 C） 4：直流接口枪头（带枪线，模	是	整型	

		式 4) 5：无线充电座 6：其他			
额定电压上限	VoltageUpperLimits	单位：V	是	整型	
额定电压下限	VoltageLowerLimits	单位：V	是	整型	
额定电流	Current	单位：A	是	整型	
额定功率	Power	单位：kW	是	浮点型	保留小数点后一位
车位号	ParkNo	停车场车位编号	否	字符串	10 字符
国家标准	NationalStandard	1:2011 2:2015	是	整型	

返回值

参数名称	定义	参数类型	描述
状态	SuccStat	整型	0:成功; 1:失败
原因	FailReason	字符串	成功或失败原因

三、代码示例（java）

加密

```
/**
 * 加密
 *
 * @param data            被加密数据
 * @param dataSecretIV 初始向量 AES 为 16bytes. DES 为 8bytes
 * @param dataSecret     私钥  AES 固定格式为 128/192/256 bits.即：16/24/32bytes。DES 固定格式为 128bits，即 8bytes。
 * @return
 */
public static String encrypt (String data, String dataSecretIV, String dataSecret) {
   try {
      if (data == null || data.isEmpty()) {
         return "";
```

```
        }
        IvParameterSpec zeroIv = new IvParameterSpec(dataSecretIV.getBytes());
        SecretKeySpec key = new SecretKeySpec(dataSecret.getBytes(), "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding"); //PKCS5Padding 比 PKCS7Padding
效率高，PKCS7Padding 可支持 IOS 加解密
        cipher.init(Cipher.ENCRYPT_MODE, key, zeroIv);
        byte[] encryptedData = cipher.doFinal(data.getBytes( "UTF-8"));

        return Base64.encode(encryptedData).replace("\n", "").replace("\r", "");
    } catch (Exception e) {
        log.error("AesUtil-encrypt-error: param:
encrypted:{},dataSecretIV:{},dataSecret:{}",data,dataSecretIV,dataSecret,e);
        return "";
    }
}
```

解密

```
/**
 * 解密
 */
public static String decrypt (String data, String dataSecretIV, String dataSecret) {
    try {
        if (data == null || data.isEmpty()) {
            return "";
        }
        byte[] byteMi =Base64.decode(data);
        IvParameterSpec zeroIv = new IvParameterSpec(dataSecretIV.getBytes());
        SecretKeySpec key = new SecretKeySpec(dataSecret.getBytes(), "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE, key, zeroIv);
        byte[] decryptedData = cipher.doFinal(byteMi);
        return new String(decryptedData, "UTF-8");
    } catch (Exception e) {
        return "";
```

```
    }
}
```

签名

```
/**
 * @param aValue 加密的原文，即源数据
 * @param sigSecret    签名密钥
 * @s 这是针对一条字符串进行加密的方法
 */
public static String hmacSign (String aValue, String sigSecret) {
    byte[] k_ipad = new byte[64];
    byte[] k_opad = new byte[64];
    byte[] keyb;
    byte[] value;
    keyb = sigSecret.getBytes(StandardCharsets.UTF_8);
    value = aValue.getBytes(StandardCharsets.UTF_8);

    Arrays.fill(k_ipad, keyb.length, 64, (byte) 54);
    Arrays.fill(k_opad, keyb.length, 64, (byte) 92);
    for (int i = 0; i < keyb.length; i++) {
        k_ipad[i] = (byte) (keyb[i] ^ 0x36);
        k_opad[i] = (byte) (keyb[i] ^ 0x5c);
    }

    MessageDigest md;
    try {
        md = MessageDigest.getInstance("MD5");
    } catch (NoSuchAlgorithmException e) {
        return null;
    }
    md.update(k_ipad);
    md.update(value);
    byte[] dg = md.digest();
    md.reset();
    md.update(k_opad);
```

```
    md.update(dg, 0, 16);
    dg = md.digest();
    return toHex(dg);
}
```

转成 16 进制

```
public static String toHex (byte[] input) {
    if (input == null) {
        return null;
    }
    StringBuffer output = new StringBuffer(input.length * 2);
    for (byte b : input) {
        int current = b & 0xff;
        if (current < 16) {
            output.append("0");
        }
        output.append(Integer.toString(current, 16));
    }
    return output.toString().toUpperCase();
}
```

加解密、签名、token 调用

加密数据：encryptData=encrypt(decryptData, DataSecretIv, DataSecret);

解密数据：decryptData=decrypt(encryptData, DataSecretIv, DataSecret)

签名数据 sig = hmacSign(ret + msg + data, SigSecret);

token 生成：token=encrypt(OperatorID+uuid, DataSecretIV, OperatorSecret)