http://10.30.10.21:8180/rest/v2/device/activate?udid=70c66d125385ae3&appKey=bvdkiumgmzplrp3sslhv5pp2gqqhttmzg5gh2bq5&timestamp=1493969183&signature=bb0d225098ad1907d5a63750627060a140f0b3ad

udid=test-udid01

1. REST接口参数描述：

 1.1 设备激活接口的REST API（POST请求）：

 URL：

内网测试URL：[http://10.30.10.21:8180/rest/](http://10.30.10.21:8180/rest/v2/device/activate)**[v2](http://10.30.10.21:8180/rest/v2/device/activate)**[/device/activate](http://10.30.10.21:8180/rest/v2/device/activate)

[线上正式URL：](http://10.30.10.21:8180/rest/v1/device/activate)[http://dc.hivoice.cn/rest/**v2**/device/activate](http://dc.hivoice.cn/rest/v2/device/activate)

参数列表（所有参数在构造POST参数的时候，都需要进行**UTF-8编码**）：

udid：设备唯一标识（由deviceSn加密生成）（必填项）；  
deviceSn：客户认可的设备唯一标识（由客户端APP设置）（可选项）；  
appKey：应用KEY（必填项），测试可以用：bvdkiumgmzplrp3sslhv5pp2gqqhttmzg5gh2bq5；  
timestamp: 访问时间戳（必填项）,Unix时间戳；  
appVersion：应用版本号（可选项）；  
pkgName：包名（可选项）；  
imei：设备的IMEI（可选项）；  
macAddress：设备MAC地址（可选项）；  
wifiSsid：WIFI的名称（可选项）；  
telecomOperator：运营商（可选项）；  
bssId：当前的接入点MAC地址（可选项）；  
productName：设备名称（可选项）；  
productModel：设备型号名称（可选项）；  
productMfr：制造商名称（可选项）；  
productOs：操作系统（可选项）；  
productOsVersion：操作系统版本号（可选项）；  
hardwareSn：厂商硬件序列号（可选项）；  
memo：备注（可选项）；  
signature: 数据签名（必填项）；

1.2 设备刷新token接口的REST API（POST请求）：

 URL：

内网测试URL：[http://10.30.10.21:8180/rest/](http://10.30.10.21:8180/rest/v2/token/refresh)**[v2](http://10.30.10.21:8180/rest/v2/token/refresh)**[/token/refresh](http://10.30.10.21:8180/rest/v2/token/refresh)

线上正式URL：[http://dc.hivoice.cn/rest/](http://dc.hivoice.cn/rest/v2/token/refresh)**[v2](http://dc.hivoice.cn/rest/v2/token/refresh)**[/token/refresh](http://dc.hivoice.cn/rest/v2/token/refresh)

参数列表（所有参数在构造POST参数的时候，都需要进行**UTF-8编码**）：

udid：设备唯一标识（由deviceSn加密生成）（必填项）；  
appKey：应用KEY（必填项），测试可以用：bvdkiumgmzplrp3sslhv5pp2gqqhttmzg5gh2bq5；  
token：当前有效的token（必填项）；  
timestamp: 访问时间戳（必填项）,Unix时间戳；  
signature: 数据签名（必填项）；

以上参数中必填项不可以为空，可选项可以为空，但是SDK提供对应的设置接口，如果没有设置就传空字符串。

2. 签名构造规则：

内容签名生成规则：  
A）将所有请求参数的值放入List中，注意：计算sign时所有参数不应进行URLEncode；  
B）将格式化后的参数以字典序升序排列，拼接在一起，注意字典序中大写字母在前，空值（null）使用空字符串代替；  
C）将B形成字符串获取SHA1摘要，形成一个40位的十六进制（字母大写）字符串，即为本次请求signature（签名）的值；

然后再POST请求时，所有参数的值都应进行URLEncode；(参数值是数字或字母的可以不进行URLEncode)

该签名值基本可以保证请求是合法者发送且参数没有被修改，但无法保证不被偷窥。

**构造签名算法的JAVA代码** 折叠原码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87 | public String buildActivateSignature(ActivateReqVO reqVO, String appSecret) {      List<String> params = new ArrayList<String>();      params.add(reqVO.getUdid());      params.add(reqVO.getDeviceSn());      params.add(reqVO.getAppKey());      params.add(String.valueOf(reqVO.getTimestamp()));      params.add(reqVO.getPkgName());      params.add(reqVO.getAppVersion());      params.add(reqVO.getImei());      params.add(reqVO.getMacAddress());      params.add(reqVO.getWifiSsid());      params.add(reqVO.getTelecomOperator());      params.add(reqVO.getBssId());      params.add(reqVO.getProductName());      params.add(reqVO.getProductModel());      params.add(reqVO.getProductMfr());      params.add(reqVO.getProductOs());      params.add(reqVO.getProductOsVersion());      params.add(reqVO.getHardwareSn());      params.add(reqVO.getMemo());      params.add(appSecret);        String signature = buildSignature(params);      logger.debug("generated signature: " + signature);      return signature;  }    /\*\*   \* 对参数列表构造响应签名   \*   \* @param params   \* @return   \*/  public static String buildSignature(List<String> params) {      if (params == null || params.isEmpty()) {          return "";      }        // 升序排序参数值      Collections.sort(params);        StringBuilder sb = new StringBuilder();      for (String param : params) {          sb.append(param == null ? "" : param);      }        return getSHA1Digest(sb.toString());  }    /\*\*   \* 将字符串进行SHA1获取摘要，摘要为十六进制字符串   \*   \* @param data   \* @return   \* @throws Exception   \*/  public static String getSHA1Digest(String data) {      String digest = null;      try {          MessageDigest md = MessageDigest.getInstance("SHA-1");          byte[] bytes = md.digest(data.getBytes("UTF-8"));          digest = byte2hex(bytes);      } catch (Exception e) {          logger.error(e.getMessage(), e);      }        return digest;  }    /\*\*   \* 二进制转十六进制字符串   \*   \* @param bytes   \* @return   \*/  private static String byte2hex(byte[] bytes) {      StringBuilder sign = new StringBuilder();      for (int i = 0; i < bytes.length; i++) {          String hex = Integer.toHexString(bytes[i] & 0xFF);          if (hex.length() == 1) {              sign.append("0");          }          sign.append(hex.toUpperCase());      }        return sign.toString();  } |

3. REST接口返回的结果JSON描述：

返回的结果JSON格式如下：  
{

"returnCode": "dc\_0000",  
"message": "成功",  
"costTime": 259,  
"result": {  
"token": "442a7897-9c1b-4230-ac0e-c1365c5bf1b9",  
"validTime": 1928517  
}

}  
结果JSON中的字段说明：

returnCode: 返回结果码，SDK根据结果码判断当前请求的状态；  
message：提示信息；  
costTime：接口访问花费的时间，单位：毫秒；  
result：结果字段，激活成功时才会有此字段；  
token：当前设备的token；  
validTime：token的有效时间，单位：秒；

其中返回结果码（returnCode）包含如下：

dc\_0000：成功；  
dc\_0001：应用不存在；  
dc\_0002：请求时间戳超出了请求有效期；  
dc\_0003：签名错误；  
dc\_0004：没有激活权限  
dc\_0005：超过了最大激活频率  
dc\_0006：无效的设备token  
dc\_0007：必填参数的参数值不能为空  
dc\_9999：其他错误；