全球短信 API

调用接口：

1）请求地址：http://www.voipbeam.com/globalSMS/sendSMS

2）调用方式：get

3）接口描述：

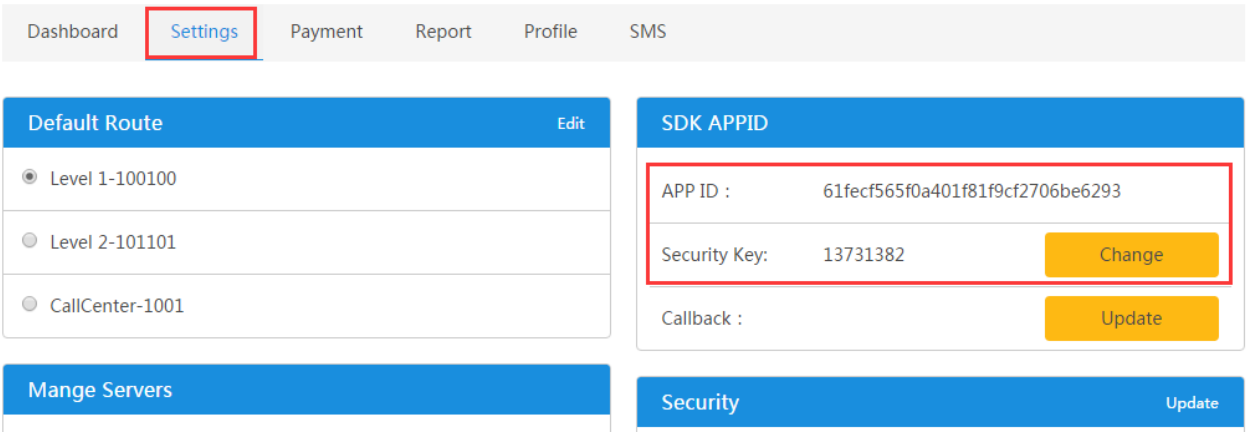
字段名称	字段说明	类型	必填	备注
sip	用户帐号	String	Y	字段名字，字段值
content	短信内容	String	Y	需要用（Security Key）使用 AES128 进行加密。
AppID	用户的应用 ID	String	Y	APP ID
dest	目标号码	String	Y	
sig	发送时间	String	Y	需要用（Security Key）使用 AES128 进行加密。发送后半小时后失效。
prefix	国家或地区的区号	String	Y	

4）返回内容（json 格式）

字段名称	字段说明	类型	必填	备注
code	响应状态	String	Y	0，发送成功； -1，程序异常； -2，缺少参数； -3，用户信息不对应； -4，短信内容为空； -5，内容加密出错； -6，发送时间超时。
msg	信息详情	String	Y	如果错误则返回错误信息详情
msg_id	成功后返回数据	String	Y	如果成功，返回短信 ID；否则没有该值。

5）注意事项：

Security Key 与 APP ID 需要在 voipbeam 的用户后台中获取。



获取短信状态接口：

1）请求地址：http://www.voipbeam.com/globalSMS/getSMSStatus
2）调用方式：get
3）接口描述：

字段名称	字段说明	类型	必填	备注
sip	用户帐号	String	Y	字段名字，字段值
AppID	用户的应用 ID	String	Y	APP ID
msg_id	短信 ID	String	Y	

4）返回内容（json 格式）

字段名称	字段说明	类型	必填	备注
code	响应状态	String	Y	0，获取成功； -1，程序异常； -2，缺少参数； -3，用户信息不对应； -4，非当前用户的短信。
msg	信息详情	String	Y	如果获取成功返回短信的状态，否则返回错误信息详情。

5）注意事项：
国内短信无法获取接收状态。

加密解密方法：

```
import java.security.SecureRandom;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;


/**
 * 加密
 *
 * @param content
 *            需要加密的内容
 * @param password
 *            加密密码
 * @return
```

```
 */
public class AES128 {

	public static byte[] encrypt(String content, String password) {
		try {
			KeyGenerator kgen = KeyGenerator.getInstance("AES");
			SecureRandom secureRandom = SecureRandom.getInstance("SHA1PRNG");
			secureRandom.setSeed(password.getBytes());
			kgen.init(128, secureRandom);
			SecretKey secretKey = kgen.generateKey();
			byte[] enCodeFormat = secretKey.getEncoded();
			SecretKeySpec key = new SecretKeySpec(enCodeFormat, "AES");
			Cipher cipher = Cipher.getInstance("AES");// 创建密码器
			byte[] byteContent = content.getBytes("utf-8");
			cipher.init(Cipher.ENCRYPT_MODE, key);// 初始化
			byte[] result = cipher.doFinal(byteContent);
			return result; // 加密
		} catch (Exception e) {
			e.printStackTrace();
		}
		return null;
	}

	/**
	 * 解密
	 *
	 * @param content
	 *                待解密内容
	 * @param password
	 *                解密密钥
	 * @return
	 */
	public static byte[] decrypt(byte[] content, String password) {
		try {
			/*
			 * KeyGenerator kgen = KeyGenerator.getInstance("AES");
			 * kgen.init(128, new SecureRandom(password.getBytes())); SecretKey
			 * secretKey = kgen.generateKey();
			 */
			KeyGenerator kgen = KeyGenerator.getInstance("AES");
			SecureRandom secureRandom = SecureRandom.getInstance("SHA1PRNG");
			secureRandom.setSeed(password.getBytes());
			kgen.init(128, secureRandom);
			SecretKey secretKey = kgen.generateKey();
```

```
            byte[] enCodeFormat = secretKey.getEncoded();
            SecretKeySpec key = new SecretKeySpec(enCodeFormat, "AES");
            Cipher cipher = Cipher.getInstance("AES");// 创建密码器
            cipher.init(Cipher.DECRYPT_MODE, key);// 初始化
            byte[] result = cipher.doFinal(content);
            return result; // 加密
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
}

public static void main(String[] args) {
        String content = "ddddddddddddddddddddddddddddddd";
        String password = "ddddddddddd";
        // 加密
        System.out.println("加密前：" + content);
        byte[] encryptResult = encrypt(content, password);
        String encryptResultStr = parseByte2HexStr(encryptResult);
        System.out.println("加密后：" + encryptResultStr);
        // 解密
        byte[] decryptFrom = parseHexStr2Byte(encryptResultStr);
        byte[] decryptResult = decrypt(decryptFrom, password);
        System.out.println("解密后：" + new String(decryptResult));

}

/**
 * 将二进制转换成 16 进制
 *
 * @param buf
 * @return
 */
public static String parseByte2HexStr(byte buf[]) {
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < buf.length; i++) {
            String hex = Integer.toHexString(buf[i] & 0xFF);
            if (hex.length() == 1) {
                hex = '0' + hex;
            }
            sb.append(hex.toUpperCase());
        }
        return sb.toString();
        /* return new String(Base64.encodeToByte(buf)).replaceAll("\\+", "%2B"); */
```

```
    }

    /**
     * 将 16 进制转换为二进制
     *
     * @param hexStr
     * @return
     */
    public static byte[] parseHexStr2Byte(String hexStr) {
        if (hexStr.length() < 1)
            return null;
        byte[] result = new byte[hexStr.length() / 2];
        for (int i = 0; i < hexStr.length() / 2; i++) {
            int high = Integer.parseInt(hexStr.substring(i * 2, i * 2 + 1), 16);
            int low = Integer.parseInt(hexStr.substring(i * 2 + 1, i * 2 + 2), 16);
            result[i] = (byte) (high * 16 + low);
        }
        return result;
        /* return Base64.decode(hexStr); */
    }
}
```