**接口名称**

通知消息发送接口

**方法名称**

sendMessage

**请求方式**

post

**参数格式**

json

**接口类型**

restful

**开发语言**

java

**host**

127.0.0.1 api.speiyou.cn

**线上 url**

http://message.speiyou.cn/message-center/message/sendmessage

**仿真 url**

http://message.fangzhen.speiyou.cn/message-center/message/sendmessage ip:10.200.172.168 公网ip：59.110.147.18

**调用方**

[]

**http 头**

msgType: 1：短信 2：邮件 3：钉钉  
 ------

**输入参数**

platform 平台编码 必传 String  
accessKey 接入方秘钥 必传 String   
msgType 消息类型 请求头部（必传） Integer(短信为1，邮件为2)  
singleMsgId 消息防重id 必传 . String  
templateId 模板编码 必传 String  
timingTime 发送消息时间 可选 Long（Unix时间戳）  
isReport 状态报告回执 可选 Integer  
isReply 消息回复 可选 Integer  
content 消息内容 必传 String  
signature 加密字符串 必传 String  
  
signature加密方式：  
利用AES加密方式，加密秘钥accessKey，如果accessKey不满足16位，则使用默认秘钥“ThisIsASecretKey”,加密字符串位：phoneNumbers/to+templateId+msgType。  
再将生成的2进制数组转成16进制字符串。  
  
发送短信：  
phoneNumbers 电话号码 必传  
  
发送邮件:  
from 发送方邮件账号 必传  
to 接收方邮件账号 必传  
cc 抄送邮件账号 可选  
filePath 附件路径

**旧输出参数**

errMsg 返回消息描述  
code 状态码  
success 是否成功

**备注说明**

accessKey，platform，templateId请找@南晴@张广羚申请  
加密工具类  
public class AesUtil {  
  
public final static String ALGORITHM\_AEPP = "AES/ECB/PKCS5Padding";  
  
private static final byte[] KEY= new byte[] { 'T', 'h', 'i', 's', 'I', 's', 'A', 'S', 'e', 'c', 'r', 'e', 't', 'K', 'e', 'y' };  
  
/\*\*  
\* AES加密  
\*  
\* @param content 内容  
\* @param password 密钥  
\* @return 加密后数据  
\*/  
public static byte[] encrypt(byte[] content, byte[] password) {  
if (content == null || password == null) {  
return null;  
}  
try {  
Key key=null;  
Cipher cipher;  
if (ALGORITHM\_AEPP.endsWith("PKCS7Padding")) {  
cipher = Cipher.getInstance(ALGORITHM\_AEPP, "BC");  
} else {  
cipher = Cipher.getInstance(ALGORITHM\_AEPP);  
}  
if(password.length!= 16){  
key = new SecretKeySpec(KEY, "AES");  
}else{  
key = new SecretKeySpec(password, "AES");  
}  
cipher.init(Cipher.ENCRYPT\_MODE, key);  
return cipher.doFinal(content);  
} catch (Exception e) {  
return null;  
}  
}  
  
/\*\*  
\* 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();  
}  
  
}  
String sourceStr = (phoneNumbers+ templateId + msgType);  
byte[] buf = AES.encrypt(sourceStr.getBytes(), accessKey.getBytes());  
String signature = AES.parseByte2HexStr(buf);

**请求示例**

{"accessKey":"46DF752BF9F7FC80","content":"{'text':'coding'}","phoneNumbers":"18840852084","platform":100030,"signature":"F1423AB0A979382D4E621C16E2C41C5F87A549063869B12F87A0ED4F0FF21601","singleMsgId":"100030","templateId":"100030"}

**状态**

正常

**接口类型**

内部接口

**安全级别**

高