讯飞 AI 服务市场

授权码激活的软件类服务接入指南

V1.0

科大讯飞股份有限公司
iFLYTEK CO., LTD

变更历史

版本	变更说明	作者	日期
V1	创建文档	Jacky	2018-06-25

目录

1. 简介

对于一些有开发能力的服务商，我们提供了自动获取授权信息的接口，方便服务商在平台成交的服务能够自动发送授权信息给企业客户，提高服务商效率。

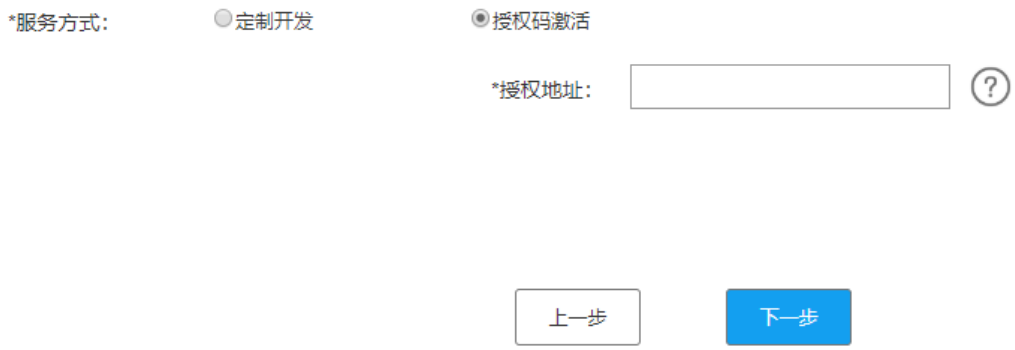
2. 服务商密钥（aesKey=NGH39FbRoKfNu5bM）

由服务市场平台生成，服务市场平台颁发 入驻成功会自动生成，查看路径：服务商中心>基本资料>Key

3. 服务商授权地址(authUrl)

3.1 授权码激活的软件类服务

填写截图如下



4. 接口概述

接口提供方：服务商
接口调用方：服务市场

4.1 接口地址

```
POST ${authUrl} HTTP/1.1
Content-Type:"application/x-www-form-urlencoded; charset=utf-8"
```

4.2 参数说明

参数	类型	必须	说明	示例
reqParam	String	是	服务市场加密后的入参	生成策略参见 5.1

5. 接口参数生成策略

5.1 reqParam 生成策略

第一步:AES 加密(关键参数+过期时间戳,aesKey)
第二步:对以上加密结果进行 Base64 编码

5.1.1 关键参数

参数	类型	说明	示例
from	String	来源	xfyun
isTest	bool	测试（服务商发布测试，服务商自行控制是否销毁此次调用返回的授权信息） 正式（来自线上用户购买请求）	true|false
goodsName	String	服务名称	Ai 语音服务
skuInfo	string	每页显示条数	[{"attr":"使用周期","spec":"1 年 "},{"attr":" 使 用 次 数 ","spec":"1000 次"}]

5.1.2 过期时间 expireTime 秒

安全考虑，服务商验证 如果过期请忽略此次请求 默认 5 分钟

5.2 Demo

5.2.1 平台会按照以下格式构造 reqParam

```
 String json =
{
```

```
    "expireTime": 1526993203,
    "isTest": false,
    "from": "xfyun",
    "goodsName": "语音听写",
    "skuInfo": [
        {
            "attr": "初级包",
            "spec": "72000 元"
        },
        {
            "attr": "服务次数",
            "spec": "2 千万次"
        },
        {
            "attr": "有效期",
            "spec": "1 年"
        }
    ]
}
String reqParam = AesUtil.encrypt(json,aesKey)
```

5.2.2 curl 示例

请求：

```
curl  -d "reqParam=
w4L2i2ONdAQCbLXR2w/0zl0gZY1fMEYuYkkaAB3K9DrXeoNIlmM7/gAymLVnNSGQ4c1Si7iLtUz
6K/dhyfIssvJfC32jMuoKQm5t6D2yTwg0cbqKAln+cH9OPHYH5A1gvgJ1lbqUU1cv8twx1ZWkVt
5vix7+J5oYTNVIV4/rD133xwiWy6X1VTKqJNpF4tZQCwIbZ8nin4/0eyXmGXBxQGq8VUGoLZvps
dnjJ5LNhnoVQiKplBsKS80NT+CBSvL3c0u3pY2X41sH37VSUi9AX5yB+S70Lm2hPBrAfvY7Z78=
" "${authUrl}"
```

响应体：

```
   成功
{
        "code": 0,
        "data": " 提供给企业客户的经过 AES 加密过的授权信息"
}
   失败
{
        "code":自定义错误码（数字型）,
        "desc": " 错误信息"
}
```

6 服务商

6.1 解密处理

String result = AesUtil.decrypt(reqParam,aesKey);
result 是个 json 格式字符串，请自行处理
aesKey:参考服务商密钥

6.2 加密处理

String authInfo = "token=123456&appId=123";//需要展示给用户的授权信息
String data= AesUtil.encrypt(authInfo,aesKey);
// iK9Ge1Zo8sbR4WoBu/32tEgAnPYz7FmJa4fvq67po+A=

Demo:

返回

```
{
        "code": 0,
        "data" : "iK9Ge1Zo8sbR4WoBu/32tEgAnPYz7FmJa4fvq67po+A="
}
```

7 JAVA 版加解密 AesUtil.java 参考

```
import javax.crypto.*;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

public class AesUtil {

   /**
     * 加密
     * @param content 需要加密的内容
     * @param password  加密密码
     * @return
     */
    public static String encrypt(String content, String password) {
        try {
            KeyGenerator kgen = KeyGenerator.getInstance("AES");
            kgen.init(128, new SecureRandom(password.getBytes()));
            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 new sun.misc.BASE64Encoder().encode(result);
            //return result; // 加密
        } catch (NoSuchAlgorithmException | NoSuchPaddingException |
InvalidKeyException | IllegalBlockSizeException |
UnsupportedEncodingException | BadPaddingException e) {
            e.printStackTrace();
        }
        return null;
    }

    /**解密
     * @param content  待解密内容
     * @param password 解密密钥
     * @return
```

```
     */
    public static String decrypt(String content, String password) {
        try {
            byte[] encrypted = new
sun.misc.BASE64Decoder().decodeBuffer(content);
            KeyGenerator kgen = KeyGenerator.getInstance("AES");
            kgen.init(128, new SecureRandom(password.getBytes()));
            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(encrypted);
            return new String(result); // 加密
        } catch (NoSuchAlgorithmException | NoSuchPaddingException |
InvalidKeyException | BadPaddingException | IllegalBlockSizeException |
IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```