内容识别开发文档 V2

1 说明

接口开发文档包含 6 部分：

- **检测 API 请求和应答格式说明**
  检测 API 接口的请求格式描述。
- **检测 API 公共字段**
  接口通用字段，具体接口处不再单独说明。
- **接口鉴权**
  接口鉴权方式。
- **数据回调 API 请求和应答格式说明**
  数据的异步执行结果，通过回调反馈结果。
- **文本类**
  文本识别功能接口说明；数据回调接口说明。
- **图片类**
  图片识别功能接口说明；数据回调接口说明。

2 检测 API 请求和应答格式

接口协议：HTTP。
检测请求方法（Method）为：POST。
检测请求体的类型（Content-Type）为：application/json。
检测结果：返回 JSON 格式数据。
具体参数详见文本类和图片类的检测接口描述。

3 公共字段

公共参数是用于标识产品和接口鉴权目的的参数，每次请求均需要携带这些参数。
注：如非必要，在每个接口单独的接口文档中不再对这些参数进行说明。

3.1 请求公共字段

请求公共参数

参数名称	类型	必选	最大长度	描述
appId	String	Y	32	产品标识，由系统为您分配
timestamp	Number	Y	13	请求的当前 UNIX 时间戳。
nonce	Number	Y	11	随机整数，防重放
signature	String	Y	32	请求签名，用来验证本次请求的合法性，具体算法见 接口鉴权

3.2 响应公共字段

所有接口响应采用 json 格式， 如无特殊说明，每次请求的返回值中，都包含下面字段：

参数名称	类型	描述
code	Number	接口返回值，200:正常；其他值：调用出错
msg	String	结果说明，调用正常，返回 ok,接口出错会返回错误描述。
result	Json Object	接口返回结果，json 格式，详见每个接口的定义说明。

3.3 响应返回码

响应返回码（code）是系统服务执行的状态结果。当返回码不为 200 时， 表示请求未正常执行，返回码描述 (msg) 对该结果进行了细化补充，用户可根据返回码判断 API 的执行情况。

所有接口调用返回值均包含 code 和 msg 字段， code 为返回码值，msg 为返回码描述信息，返回码表如下：

返回码	返回码描述	说明
200	ok	接口调用成功
400	bad request	必要参数没有填写完全，如 appId 或 signature 等
401	signature error	签名验证失败
402	bad parmeters	请求业务参数不合法
403	bad http method	不支持的 http 方法
404	not found	请求地址或业务不存在
410	high frequency	请求频率超过限制
411	len exceed quota	请求长度超过限制
503	service error	服务处理异常

4 接口鉴权

对外提供的 api 接口，使用签名和验签的方式验证请求的合法性。用户每一次调用请求都需要计算签名并放置到请求的 signature 字段中。

4.1 获取安全凭证

在调用 API 之前，首先需要申请安全凭证，即 appId 和 secretKey，appId 是用于标识 API 调用者的身份和业务类别，secretKey 是用于加密签名字符串和服务器端验证签名字符串的密钥，secretKey 需要用户严格保管，避免泄露。

4.2 签名计算方式

如无特殊说明，所有 api 接口的签名均采用如下方式：

1、组合要签名的字符串，appId，timestamp，secretKey 三个字段以半角逗号相连拼接。例如：

appId：201705120001

timestamp：1496202697433

secretKey：5156F2FBA13D87F663FA0128BCD422738

最后格式为：201705120001,1496202697433,5156F2FBA13D87F663FA0128BCD422738

2、把上一步骤拼装好的字符串采用 utf-8 编码，计算 md5 值，得到的结果为 16 字节的二进制数据，转换为十六进制字符串即为 signature。

4.3 代码示例

appid：201705120001

timestamp：1496202697433

secretKey：5156F2FBA13D87F663FA0128BCD422738

计算得到的 signature：a6e934710d1adbe7752068da31dea525

```
/**
 * @param data 要参与签名的数据
 * @return 返回十六进制的字符串
 */
public static String getSignature(String data) throws UnsupportedEncodingException {
    //计算 MD5，然后转换为十六进制字符串。
    return DigestUtils.md5Hex(data.getBytes("UTF-8"));
}
```

5 回调 API 请求和应答格式

接口协议：HTTP。

检测请求方法（Method）为：POST。

检测请求体的类型（Content-Type）为：application/x-www-form-urlencoded。

检测结果：返回 JSON 格式数据。

具体参数详见文本类和图片类的回调接口描述。

6 文本类接口

6.1 文本实时识别

6.1.1 请求数据

请求需包含请求公共字段，其他字段信息如下：

字段名称	类型	必选	最大长度	描述
id	String	Y	128	用户数据唯一标识,能够根据该值定位到该条数据
content	String	Y	10000	用户发表内容，建议对内容中 HTML 标签、JSON 格式、表情符等做过滤，只发送纯文本内容。
extension	String	N	128	扩展信息，用于实时识别没有准确结果，待结果变更时回调使用。系统回调时，会原样返回 id 和 extension 两个字段，作为用户数据处理标识。例如用户修改昵称的数据，修改了多次，id 是一样的，无法区分先后顺序，用户就可以在这个字段设置时间戳等。此字段完全由用户根据业务自行设计，目的就是通过 id+extension 可以准确确定一条数据。如果没有此类需求，此字段可不填写
ip	String	N	64	用户 IP 地址
uid	String	Y	128	用户唯一标识，如游戏中的角色 Id
nickName	String	N	64	用户昵称或角色名称
msgType	Number	N		消息类型（聊天频道），1-世界，2-公会，3-私聊，其他类型可自行定义，值：4-127
toUid	String	N	128	接收者的用户唯一标记
deviceId	String	N	128	用户设备 id
publishTime	Number	N	8	发表时间，UNIX 时间戳(毫秒值)
openId	String	N	128	非必填，用户账户 id
region	String	N	64	非必填，游戏区服等信息
userLevel	Number	N	8	非必填，用户等级，处理用户时有辅助作用

6.1.2 响应数据

响应需包含响应公共字段，result 的 json 字段信息如下：

字段名称	类型	描述
action	Number	识别结果，0：不通过，1：通过，2：可疑
rule	Number	类别，1：角色禁言,0：其他
score	Double	可信度 0-100.0，分值越高，可信度越大
reason	String	参考分类，包含：政治、色情、广告、灌水等。通过时没有原因

格式如：

```
{
    “action”: 0,
    “rule”:1,
    “score”: 100.0,
    “reason”: ”用户策略”
}
```

6.1.3 代码示例

```
/**
 * 实时识别文本的示例
 */
public class TextDetectApi {

    // 平台 api 地址
    private static final String API_URL = "http://api.test-yun.cn:8080/v2/text";

    private static Gson gson = new Gson();

    public static void main(String[] args) throws Exception {
        // appId 需要平台进行分配，在调用时替换成您实际申请的 appId
        String appId = "201612100001";
        // 获取您自己的密钥
        String secretKey = "0000000000000000";
        // 获取当前时间戳
        long timestamp = System.currentTimeMillis();
        // 需要签名的数据，格式如：“201612100001,123333122”
        String signData = appId + "," + timestamp;
        // 用你的私钥对 sign_string 进行签名
        String signature = SignUtil.getSignature( secretKey , signData);
```

```
        TextRequestJson request = new TextRequestJson();
        //--公共参数
        request.setAppId(appId);
        request.setTimestamp(timestamp);
        request.setNonce(new Random().nextInt());
        request.setSignature(signature);
        //--业务参数
        //----必填项
        request.setId(UUID.randomUUID().toString().replace("-", ""));
        request.setContent("当微信公众号被封，我们该看什么？");
        request.setUid("110110");//发布用户的 id 或账户名称
        //----建议选填项
        request.setPublishTime(System.currentTimeMillis());
        //request.setIp("127.0.0.1");//用户真实 ip 时建议填写
        //request.setDeviceId("1234567890");//用户的设备号，建议填写
        //request.setNickName("小二哥");
        //request.setToUid("599912");//接收者的 id
        //request.setExtension("5555");//有需求时可根据业务自行设定格式填写，例如编辑类
的，可填写{"type":"edit"}
        String strResult = HttpClientUtil.postJson(API_URL, gson.toJson(request));
        System.out.println(strResult);
    }
}
```

6.2 文本离线结果回调

离线数据，结果有变更时，系统会及时回调用户。
需要用户开发相应接口，能够接收系统回调的数据，并在运营平台配置回调地址。
鉴于网络原因，有可能会出现超时等情况，如果调用失败系统会尝试 3 次，间隔 20 秒。

6.2.1 回调请求数据

请求由检测平台发起：数据格式为：version=1.x&json_data=xxxxx&sign=xxxxx
签名算法：由于回调数据比较敏感，因此采用 HmacSha1 算法进行签名。

- **version**：回调版本号，目前为 1.1
- **json_data**：回调的数据内容
- **sign**：系统对 json_data 的 HmacSha1 算法签名。使用您的 secretKey 进行验签。

Java 验签示例代码

```
/**
     * 使用 HMAC-SHA1 签名方法对对 encryptText 进行签名
     * @param encryptText 被签名的字符串
```

```
     * @param encryptKey  密钥
     */
    public static byte[] hmacSHA1Encrypt(String encryptText, String encryptKey) throws
Exception {
        byte[] data=encryptKey.getBytes(ENCODING);
        //根据给定的字节数组构造一个密钥,第二参数指定一个密钥算法的名称
        SecretKey secretKey = new SecretKeySpec(data, MAC_NAME);
        //生成一个指定 Mac 算法 的 Mac 对象
        Mac mac = Mac.getInstance(MAC_NAME);
        //用给定密钥初始化 Mac 对象
        mac.init(secretKey);
        byte[] text = encryptText.getBytes(ENCODING);
        byte[] enText = mac.doFinal(text);
        return Base64.encodeBase64(enText);
    }

    /**
      * 验证 hmacSha1 是否匹配
      * @param encryptText
      * @param signature
      * @param encryptKey
     */
public static boolean verify(String encryptText, String signature,String encryptKey){
     try {
               byte[] destSign = hmacSHA1Encrypt(encryptText,encryptKey);
               String destStr = new String(destSign,ENCODING);
               if (destStr.equalsIgnoreCase(signature)) {
                    return true;
               }
          } catch (Exception e) {
                logger.error("verify error:[{}]", encryptKey, e);
          }
     return false;
    }
```

请求的 json 数据格式说明：

字段名称	类型	必选	最大长度	描述
appId	String	Y		产品标识，由系统为您分配的
id	String	Y	128	用户数据唯一标识，能够根据该值定位到该条数据
extension	String	Y	128	用户自定义的数据，原样返回
result	Json Object	Y		数据结果，json 对象

result 对象的格式说明（定义为 json 方便扩展）

字段名称	类型	必选	最大长度	描述
action	Number	Y		结果，0：不通过，1：通过，2：可疑

6.2.2 回调响应数据

客户收到回调数据处理完毕后，需反馈结果，格式为 JSON。code 为非 200 的值或没有正常反馈 JSON 结构，系统会认为调用失败，最多尝试 3 次调用。

字段名称	类型	必选	最大长度	描述
code	Number	Y		200-正常；其余值为存在问题

7 图片类接口

7.1 图片识别

7.1.1 请求数据

请求需包含请求公共字段，其他字段信息如下：

字段名称	类型	必选	最大长度	描述
id	String	Y	128	用户数据唯一标识,能够根据该值定位到该条数据
images	Array	Y	10 张图片	图片对象数组,json 格式，见 image 说明
ip	String	N	32	用户 IP 地址
uid	String	N	128	用户唯一标识，如果无需登录则为空
nickName	String	N	64	用户昵称
toUid	String	N	128	接收者的用户唯一标记
deviceId	String	N	128	用户设备 id
publishTime	Number	N	13	发表时间，UNIX 时间戳(毫秒值)
extension	String	N	128	扩展信息，用于实时识别没有准确结果，待结果变更时回调使用。系统回调时，会原样返回 id 和 extension 两个字段，作为用户数据处理标识。例如用户修改昵称的数据,修改了多次,id 是一样的，无法区分先后顺序，用户就可以在这个字段设置时间戳等。此字段完

				全由用户根据业务自行设计,目的就是通过 id+extension 可以准确确定一条数据。如果没有此类需求,此字段可不填写

image 对象说明

字段名称	类型	必选	最大长度	描述
name	String	Y	512	图片名称，超出 512 会截断保存
data	String	Y	2048	图片 url

7.1.2 响应数据

响应需包含响应公共字段，result 的 json 字段信息如下：

参数名称	类型	描述
action	Number	识别结果，0：不通过，1：通过，2：可疑
score	Double	可信度 0-100，分值越高，可信度越大
reason	Number	1

说明：批量识别时，action 优先级为 不通过、可疑、通过。

格式如：

```
{
    action: 0
    score: 98.0
    reason: ”色情”
}
```

7.1.3 代码示例

```
/**
 * 实时识别图片的示例
 */
public class ImageDetectApi {
    // 平台 api 地址
    private static final String API_URL = "http://api.test-yun.cn:8080/v2/image";
    private static Gson gson = new Gson();

    public static void main(String[] args) throws Exception {
        // appId 需要平台进行分配，在调用时替换成您实际申请的 appId
        String appId = "201612100001";
        // 获取您自己的密钥
```

```
        String secretKey = "0000000000000000";
        // 获取当前时间戳
        long timestamp = System.currentTimeMillis();
        // 需要签名的数据，格式如：“201612100001,123333122”
        String signData = appId + "," + timestamp;
        // 用你的私钥对 sign_string 进行签名
        String signature = SignUtil.getSignature(secretKey, signData);
        ImageRequestJson request = new ImageRequestJson();
        //--公共参数
        request.setAppId(appId);
        request.setTimestamp(timestamp);
        request.setNonce(new Random().nextInt());
        request.setSignature(signature);
        //--业务参数
        //----必填项
        request.setId(UUID.randomUUID().toString().replace("-", ""));
        request.addImage("test1", "http://img1.mm131.com/pic/2966/5.jpg");
        request.addImage("test2", "http://img1.mm131.com/pic/2966/8.jpg");
        //----建议选填项
        request.setPublishTime(System.currentTimeMillis());
        request.setUid("110111");//建议填写发布用户的 id 或账户名称
        //request.setIp("127.0.0.1");//用户真实 ip 时建议填写
        //request.setDeviceId("1234567890");//用户的设备号，建议填写
        //request.setNickName("小二哥");
        //request.setToUid("599912");//接收者的 id
        //request.setExtension("5555");//有需求时可根据业务自行设定格式填写，例如编辑类
的，可填写{"type":"edit"}
        String strResult = HttpClientUtil.postJson(API_URL, gson.toJson(request));
        System.out.println(strResult);
    }
}
```

7.2 图片离线结果回调

离线数据，结果有变更时，系统会及时回调用户。
需要用户开发相应接口，能够接收系统回调的数据，并在系统运营平台配置回调地址。
鉴于网络原因，有可能会出现超时等情况，如果调用失败系统会尝试 3 次，间隔 20 秒。

7.2.1 回调请求数据

同文本回调请求

7.2.2 回调响应数据

同文本响应数据