腾讯慧眼1v1说明文档——H5端

流程说明

1. 通过`postcompareinfo`接口传入1v1相关数据
2. 调用`startcompare`接口开始识别(此时会跳转到腾讯侧页面等待)
3. 1v1成功后，将跳转回业务页面(业务页面url需要在`startcompare`接口设置)
4. 通过`getdetectinfo`结果拉取1v1比对结果

接口说明

预提交1v1数据接口

- 接口
  https://iauth-sandbox.wecity.qq.com/new/cgi-bin/postcompareinfo.php
- 描述
  预提交1v1数据，需要和2.2接口配合使用
- 参数格式
  body为`json`格式数据，请求方式为`POST`
- 头部信息

要求	参数名	类型	参数说明
必选	signature	String	接口签名，具体见签名算法

- 请求包体

要求	参数名称	类型	说明
必选	appid	String	分配的appid
必选	image2	String	用户自拍照片内容，base64编码(<5M)
必选	compare_type	int	compare_type=3时，使用预存参数拉取腾讯证件库进行图像比对； compare_type=4时，使用预存参数直接进行图像比对
必选	uid	String	一般传参用户uid
可选	ID	String	身份证号码
可选	name	String	姓名
可选	image_type	int	image_type=0时为带网纹证件照片；image_type=1时为高清证件照
可选	image1	String	证件照片内容，base64编码(<5M)；compare_type=4时提供
可选	phone	String	手机号

- 回参格式

```
{
  "errorcode": 0,
  "errormsg": "success",
  "data": {
    "token": "{xxx-xxx-xxx-xxx}"
  }
}
```

开始1v1比对接口

- 接口
  https://iauth-sandbox.wecity.qq.com/new/cgi-bin/startcompare.php
- 描述
  开始1v1比对，此接口正确调用后将重定向到腾讯侧loading页面，待1v1比对成功/失败后将跳转至业务侧页面
- 参数格式
  body为`json`格式数据，请求方式为`POST`

注意，此接口成功时将自动重定向，故需要使用**form**表单方式提交数据

要求	参数名称	类型	说明
必选	token	String	业务流水号，2.1接口返回的token
必选	appid	String	分配的appid
必选	signature	String	接口签名
必选	redirect	String	回调地址

1v1结果回调

- 接口
  无
- 描述
  1v1比对成功后，将跳转到2.2接口内的`redirect`地址上，相关参数将拼接在redirect后方
  如：
  `redirect?token=TOKEN&;uid=UID`
  拿到token后使用2.4接口拉取识别信息结果
- 参数格式
  无

注意 重定向参数可能为空或被恶意篡改，请注意代码健壮性

拉取识别信息

- 接口
  https://iauth-sandbox.wecity.qq.com/new/cgi-bin/getdetectinfo.php
- 描述
  拉取信息接口。回包内容已加密，详细算法请参看加解密算法
- 参数格式
  body为`json`格式数据，请求方式为`POST`
- 头部信息

要求	参数名	类型	参数说明
必选	signature	String	接口签名，具体见签名算法

- 请求包体

要求	参数名	类型	参数说明
必选	token	string	上一个接口返回的token序列号
必选	appid	string	分配的appid

- 请求结果格式

要求	参数名	类型	参数说明
必选	errorcode	int	取数据是否成功，返回状态码,0表示成功，非0值为出错
必选	errormsg	String	返回错误描述
必选	data	String	BASE64数据（加密数据）

签名算法

Step 1 获取应用appid、密钥secretkey和过期时间expired

应用appid: 用来标识唯一的应用；

密钥secretkey: 使用加密算法时使用的秘钥；

expired: 当次请求的时间戳的有效时间；

Step 2 拼接有效签名串

a=xxxxx&m=xxxxxxx&t=1427786065&e=600

a为appid

m为调用的apiName，

t为当前时间戳，是一个符合UNIX Epoch时间戳规范的数值，单位为秒

e为此签名的凭证有效期，是一个符合UNIX Epoch时间戳规范的数值，单位为秒,

同appid和secretkey一样，由API提供方给定。

拼接有效签名串的结果,下文称之为orignal

Step 3 生成签名串

(1)API提供方 使用 HMAC-SHA1 算法对请求进行签名。

(2)签名串需要使用 Base64 编码。

根据签名方法signature= Base64(HMAC-SHA1(secretkey, orignal) + original)，其中secretkey为Step1获取，orignal为Step2中拼接好的签名串，对orignal使用HMAC-SHA1算法进行签名，然后将orignal附加到签名结果的末尾，再进行Base64编码，得到最终的sign。

注：此处使用的是标准的Base64编码，不是urlsafe的Base64编码，请注意。 以 JAVA 语言为例,其他语言均有类似算法. JAVA语言的示例代码见附件。

Step 4 使用签名串

将Step 3生成的signature，填充到http请求的head头部的signature字段中即可。

加解密算法

AES 256算法

使用项目提供的AES解密秘钥解密

NodeJS 代码参考

AES-256-ECB + PKCS7

```
function encryptAes256ECBPKCS7(data, resultKey) {

    try {

        let iv = ;

        var clearEncoding = 'utf8';

        var cipherEncoding = 'base64';

        var cipherChunks = [];

        var cipher = crypto.createCipheriv('aes-256-ecb', resultKey, iv);

        cipher.setAutoPadding(true);

        cipherChunks.push(cipher.update(data, clearEncoding, cipherEncoding));

        cipherChunks.push(cipher.final(cipherEncoding));

        return cipherChunks.join('');

    } catch (e) {

        console.error(e);

        return "";

    }

}

function decryptAes256ECBPKCS7(data, resultKey) {

    try {

        if (!data) {

            return ;

        }

        let iv = ;

        var clearEncoding = 'utf8';

        var cipherEncoding = 'base64';

        var cipherChunks = [];

        var decipher = crypto.createDecipheriv('aes-256-ecb', resultKey, iv);


        decipher.setAutoPadding(true);
```

```
        let buff = data.replace('\r', '').replace('\n', '');
        cipherChunks.push(decipher.update(buff, cipherEncoding, clearEncoding));
        cipherChunks.push(decipher.final(clearEncoding));
        return cipherChunks.join('');
    } catch (e) {
        console.error(e);
        return ;
    }
}
```

PHP 代码参考

AES-256-ECB +PKCS7（由于 PHP 底层的 256 和 Nodejs、Java 的不一样。所以 PHP 使用的是128长度）

```php
/**
 * 利用mcrypt做AES加密解密
 */
class AES{
    /**
     * 算法,另外还有192和256两种长度
     */
    const CIPHER = MCRYPT_RIJNDAEL_128;
    /**
     * 模式
     * 1. MCRYPT_MODE_ECB(electronic codebook)
     适合对小数量随机数据的加密，比如加密用户的登录密码之类的。
     * 2. MCRYPT_MODE_CBC (cipher block chaining)
     适合加密安全等级较高的重要文件类型。
     * 3. MCRYPT_MODE_CFB (cipher feedback)
     适合于需要对数据流的每一个字节进行加密的场合。
     * 4. MCRYPT_MODE_OFB (output feedback, in 8bit) 和CFB模式兼容，但比C
     FB模式更安全。CFB模式会引起加密的错误扩散，如果一个byte出错，则其后
     续的所有byte都会出错。OFB模式则不会有此问题。但该模式的安全度不是很
     高，不建议使用。
     * 5. MCRYPT_MODE_NOFB (output feedback, in nbit)
     和OFB兼容，由于采用了块操作算法，安全度更高。
     * 6. MCRYPT_MODE_STREAM
     是为了WAKE或者RC4等流加密算法提供的额外模型。
     */
    const MODE = MCRYPT_MODE_ECB;

    /**
     * pkcs7补码
     * @param string $string  明文
     * @param int $blocksize Blocksize , 以 byte 为单位
     * @return String
     */
    private function addPkcs7Padding($string, $blocksize = 16) {
        $len = strlen($string); //取得字符串长度
        $pad = $blocksize - ($len % $blocksize); //取得补码的长度
        $string .= str_repeat(chr($pad), $pad); //用ASCII码为补码长度的字符， 补足最后一段
        return $string;
    }

    /**
     * 加密然后base64转码
     * @param $str
     * @param $key
     * @return string
     */
    function aes256cbcEncrypt($str,$key ) {
        $iv = mcrypt_create_iv(mcrypt_get_iv_size(self::CIPHER,self::MODE),MCRYPT_ENCRYPT);
        return base64_encode(mcrypt_encrypt(self::CIPHER, $key, $this->addPkcs7Padding($str) ,
self::MODE, $iv));
    }
```

```
    /**
     * 除去pkcs7 padding
     *
     * @param String 解密后的结果
     *
     * @return String
     */
    private function stripPkcs7Padding($string){
        $slast = ord(substr($string, -1));
        $slastc = chr($slast);
        $pcheck = substr($string, -$slast);

        if(preg_match("/$slastc{".$slast."}/", $string)){
            $string = substr($string, 0, strlen($string)-$slast);
            return $string;
        } else {
            return false;
        }
    }
    /**
     * 解密
     * @param String $encryptedText 二进制的密文
     * @param String $key 密钥
     * @return String
     */
    function aes256cbcDecrypt($encryptedText, $key) {
        $encryptedText =base64_decode($encryptedText);
        $iv = mcrypt_create_iv(mcrypt_get_iv_size(self::CIPHER,self::MODE),MCRYPT_ENCRYPT);
        return $this->stripPkcs7Padding(mcrypt_decrypt(self::CIPHER, $key, $encryptedText,
self::MODE, $iv));
    }
}
```