xFace 说明文档

xFace 是针对在局域网内部署的设备，设备随版本迭代完善接口功能，开发人员可通过同局域网内的客户端发送 HTTP 请求与设备直接进行通信。本文档对人脸设备进行简单介绍，并对设备提供的接口加以解释说明，以便开发人员能够更好得使用和理解各个接口。

Demo 采用 C#语言，下载地址：https://github.com/zhpengbj/vsFaceTest

作者：张洪鹏
Email:81339155@qq.com

设备版本号	发布日期	说明
V1.0.5.151	2018-07-29	初版
V1.0.5.186	2018-08-20	增加 2.3.2-后台验证。 增加 2.1.2 中增加设备心跳包，type=5。 增加 2.2.3-时段增加或修改。 增加 2.2.4-时段删除。 增加 2.2.5-人员设置时间。 增加 2.2.6-数据刷新。 修改 2.3.1-输出照片 ftp 路径，时间、设备 IP 非 SDK 功能： 增加了看守服务（每 10 秒检查 APP 是否运行）。如果不是专有设备（比如手机测试用），需要特殊说明，给没有看守服务的 APP。 增加 FTP 服务。 增加显示“时段管理”，不能在界面上修改，如果需要修改可以调用 SDK。
V1.0.5.202	2018-08-27	增加 2.2.7-2.2.11 的接口 增加 2.3.4-历史识别记录回调方法说明 在 2.1.2 中增加 type=6（历史识别记录回调 URL） 修改设备生成识别记录的目录结构，增加了当时的日期（yyyy-MM-dd）。 增加历史识别记录回调的服务 增加了定时清理识别记录和抓拍照片的服务 修正了回调对象中 type 的属性 增加了对设备类型的判断，新增对立桶设备（F_LT_001）的文件控制，并对其硬件（指示灯、继电器、风扇）的控制
V1.0.5.223	2018-11-28	2.2.7（人员增加、更新）去掉返回结果-3 的类型。原有必须是字符或数字，现在不做任何判断 增加 http://x.x.x.x:8090/ 页面展示，’x.x.x.x’为设备 IP，可以来判断设备的接口服务是否启动。 解决接口服务异常问题 2.3.3(心跳包)增加 APP 启动时间（runtime）
V1.0.6.364	2019-3-26	**DEMO 版本：V2019.3.26.1** **App 优化：** ● 界面优化，修改参数的默认值 ● 优化识别流程，"保存认证图片"勾选时，进行异步处理 ● 界面提示：把人脸矩形框改成动态圆形；信息弹出 ● 主界面，标题字体变大，增加时间显示，增加连接状态 ● 设备授权后,不会因为重装 APP 而丢失授权 ● 判断 APP 使用内存，如果占用率过大，则重启 APP。重启耗时 2 秒

		● 修改 试用版本限制条件 ● 优化活检，首次时必须上外网激活 ● 修改 按人脸矩形框保存抓拍照片 **App 新增、修改内容：** ● 增加 "关于"里增加“存储”的信息 ● 设置参数增加二级菜单；取消“设置”按键，长按预览界面进入；进入设置界面前，需要输入密码验证 ● 摄像头采用支持 1280*720 ● 增加“变焦比例”设置，从而增加认别距离 ● 根据进出方向，提示、播放语音不同 ● 增加“停留时间”设置，未识别会再次尝试识别 ● 增加 版本日志，点击“版本号”显示 ● 增加 对特定设备的支持 ● 增加 对人脸角度判断：水平转角<15，俯仰角<15 **接口修改：** ● 增加 读取和设置 App 的运行参数。2.1.6 和 2.1.7 ● 增加 读取和设置 推送的运行参数。2.1.8 和 2.1.9 ● 增加 读取 URL 接口。2.1.5 ● 增加 读取和设置 所有 URL 的接口。2.1.1 和 2.1.5 ● 增加 还原出厂设置的接口，不会丢失授权。2.1.10 ● 增加 重启 APP 的接口。2.1.11 ● 增加 在 2.3.1 实时回调和历史回调数据项：抓拍照片 base64 的数据 ● 增加 在 2.3.1 历史回调数据项：批次、GUID、识别分数，可以通过 GUID 滤重 ● 增加 得到当天的识别记录情况。2.2.12 ● 增加 广播查找设备功能。2.1.12 ● 增加 在广播和心跳包的数据项：内存、硬盘、机器名、设备类型、是否初始化，是否已授权 等数据 ● 修改 对心跳包 2.31.3 回应，APP 会有连接状态
1		

目录

1.概述

1.1 简述

xFace 设备为局域网版本，无需公网，局域网内就可通过设备对外提供的接口即可对设备进行操作。接口围绕人员管理、照片管理、人脸认别回调等诸应用相关的核心业务，供统一的对外服务接口，供第三方平台客户进行调用。

1.2 接口规范

接口根地址：http://设备 ip 地址:8090/

接口形式：通过 HTTP 请求的方式对外提供服务。

接口安全：初次调用接口需要先对接口(2.1.1)进行初始化密码设置，后续调用任何接口都需要传入 pass 作为接口安全校验秘钥。

接口返回:所有接口返回的数据,都包含三个字段,即 result、success、msgtype、msg,称为基本字段;个别接口返回含 data 字段,携带接口响应数据处理结果,称为业务字段。

基本字段	描述	类型	附加说明
result	接口响应	Int	1:成功,0:失败. 通常只要能响应，该值均为 1
success	操作状态	Boolean	True:成功,False:失败
msgtype	返回类型	Int	接口返回的类型。提示的错误信息可能会变，但错误类型不会变，可用此字段来处理相应的功能逻辑
msg	返回信息	String	接口返回的信息

业务字段	描述	类型	附加说明
data	返回数据	Int/String/Object/List 等	返回的业务数据，类型可为数值、字符串或集合等

1.3 接口返回

接口通用返回说明:

```
public class ResultInfo<T> {
    private Int result;//仅表示接口调用状态，1 成功，0 失败，通常只要人脸设备开户、服务能响应，该
```

值均为 1

private Boolean success;//操作状态，成功为 true，以该字段为准标识操作状态

private T data;//接口返回数据封装类或集合

private Int msgtype;//异常信息类型

private String msg;//异常信息提示

}

文档中所涉及到的接口返回示例，个别接口的返回数据会有略微调整，须以真实的返回结果为准。

1.4 设备识别人员接口调用流程

主要流程：设置密码→照片同步→识别人员→识别回调

- **设置密码：**设备初始无密码，调用所有接口都需要传入密码参数；在密码设置及修改接口，newPass 和 oldPass 传入一样的值，即成为设备密码
- **照片同步：**人脸识别成功后，会显示注册人员的姓名；人员、照片创建成功后，该注册照片会保存在设备内；设备需连局域网在线
- **识别人员：**人员刷脸后，如果在设备内保存的人脸底库里，则会报相应的信息（显示、语音等）
- **识别回调 ：** 若设置了回调地址，人员识别成功后会向回调地址 POST 字段 ip、personId、time（识别时间）、deviceKey，识别分数，识别照片的 base64，识别类型，通过方向等；

其它：检查更新→ APK 下载

- 检查更新：可以调协检查更新 URL，App 启动时会先访问此 Url,得到 APK 更新包的新版本号
- APK 下载：可设置 APK 更新包的下载的 URL，配合上面的接口使用

2.接口说明

文档中会有图片配合说明，环境为：
工具：PostMan
电脑 IP：192.168.8.100
人脸识别 IP：192.168.8.101

通用返回结果说明：

msgtype	msg	结果或错误原因
-100	不固定，会报出系统异常	返回系统报错信息
-101	请使用 POST 请求	需要用post方请求
-102	接口服务未设置密码，请先设置密码	如果是初始系统，未设置密码的情况下调用API接口
-103	密码错误，请检查密码正确性	传入的密码(pass)错误

2.1 设备管理接口

2.1.1 设备通讯密码

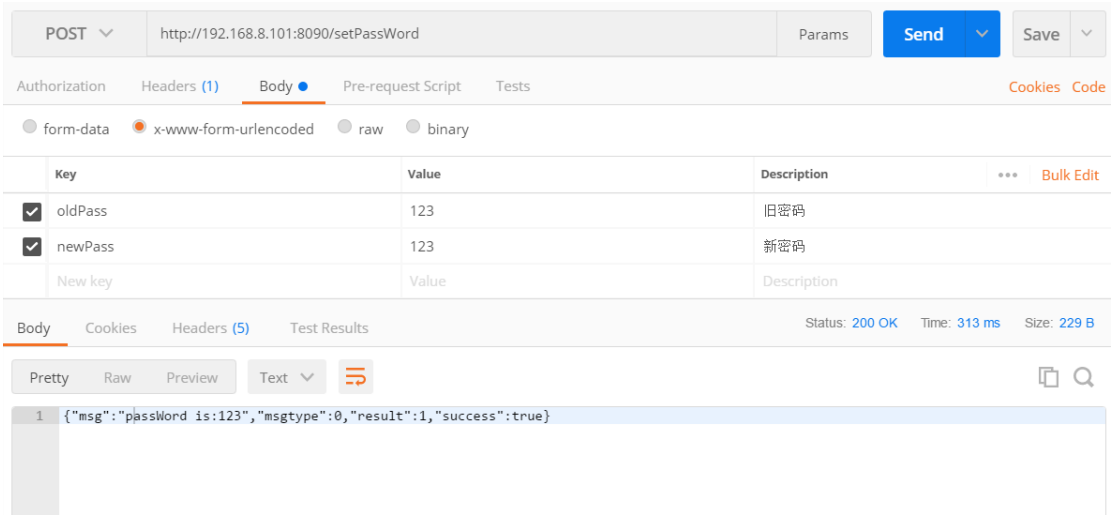
URL：/setPassWord
method：POST

参数名	描述	类型	必传	说明
oldpass	旧密码	String	Y	
newpass	新密码	String	Y	

说明：

- 新设备或还原出厂后的设备，调用其他接口前，需要先进行初始密码设置，oldPass 和 newPass 传入一样的值即可。
- 修改密码时，分别传入新旧密码
- 此接口调用不需要传入 pass
- 密码不可为空或空格
- 请妥善管理此密码，如果忘记了通讯密码，只能在设备上重装 APP!!!

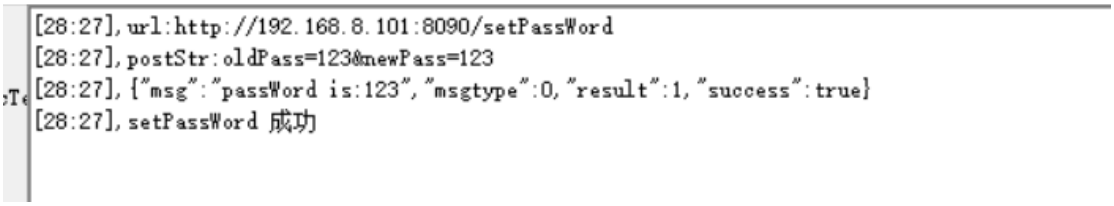
PostMan 示例：



返回示例：

{"msg":"passWord is:123","msgtype":0,"result":1,"success":true}

Demo示例：



返回结果说明：

msgtype	msg	结果或错误原因
0	passWord is:***	成功
-1	请检查密码是否为空	新密码为空
-2	初次设置密码，请确保 oldPass，newPass 相同	如果新设备，设置密码时，需要新、旧密码一样
-3	旧密码错误，要修改密码，请确保旧密码和原来设置的密码相同	修改密码时，传入错误码的旧密码

2.1.2 设置管理 URL

URL：/setUrl

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
typeId	类型	Int	Y	1:识别回调地址 2:查询新版本号地址

				3:Apk 更新包下载地址 4:后台验证回调地址 5:心跳包地址 6:历史识别回调地址 0:设置全部
callbackUrl	url 地址	String	Y	可以为空

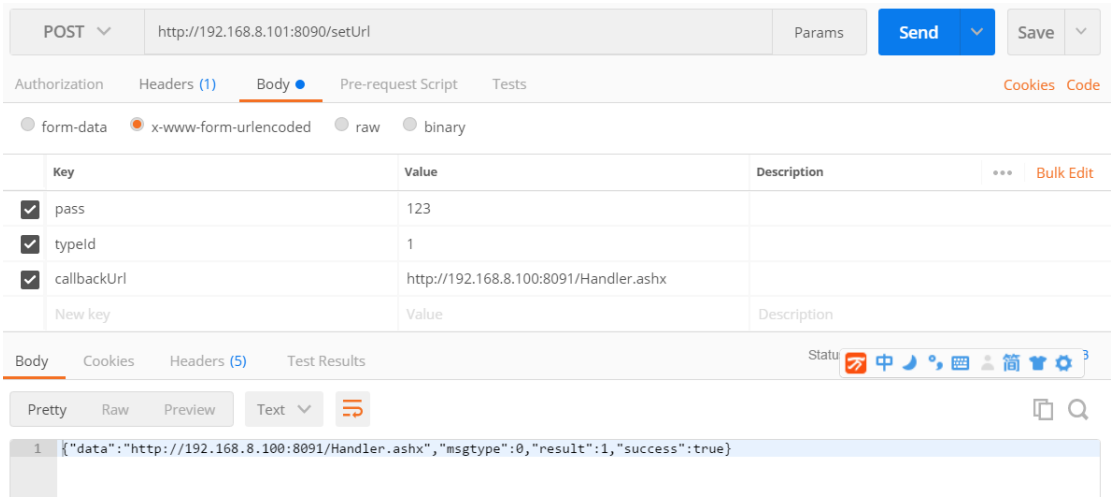
说明：

- 给设备设置一个识别回调、查询新版本号、APk 更新包下载地址
- 设置成功后当人员识别后，会向识别回调地址
- 传入内容为空可以清空地址，清空某个 url 地址后，相应的功能将失效
- 如果传入 typeid=4 时，设置为空时，则不需要后台验证。如果不为空，并且 APK 里设置了需要“平台验证”则会发起后台验证
- 如果传入 typeid=0 时，需要按如下 JsonClass 设置：

```
/// <summary>
/// 管理URL类
/// </summary>
public class UrlPar
{
    public string downNewApkUrl { get; set; }
    public string getNewApkVersionUrl { get; set; }
    public string heartBeatUrl { get; set; }
    public string identifyCallBack { get; set; }
    public string identifyCallBack_His { get; set; }
    public string verifyCallBack { get; set; }
}
```

-

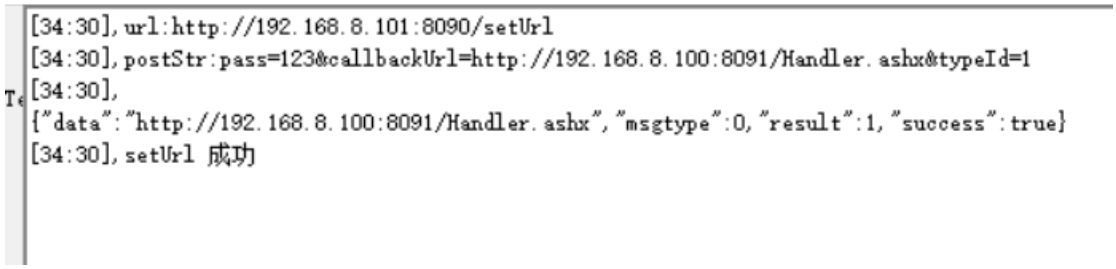
PostMan 示例：

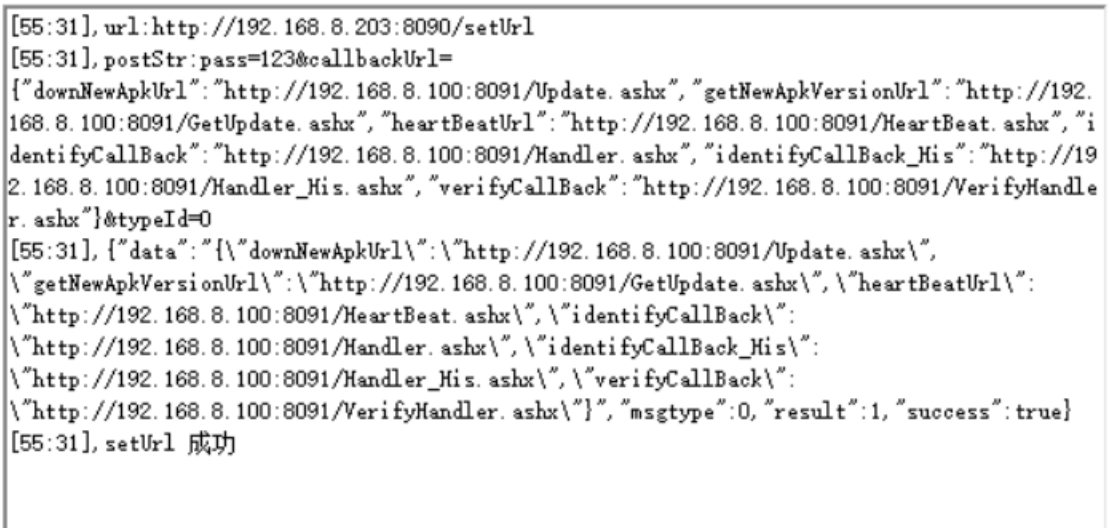


返回示例：

{"data":"http://192.168.8.100:8091/Handler.ashx","msgtype":0,"result":1,"success":true}

Demo示例：





返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功

2.1.3 得到设备机器码

URL：/getMachineCode

method：POST

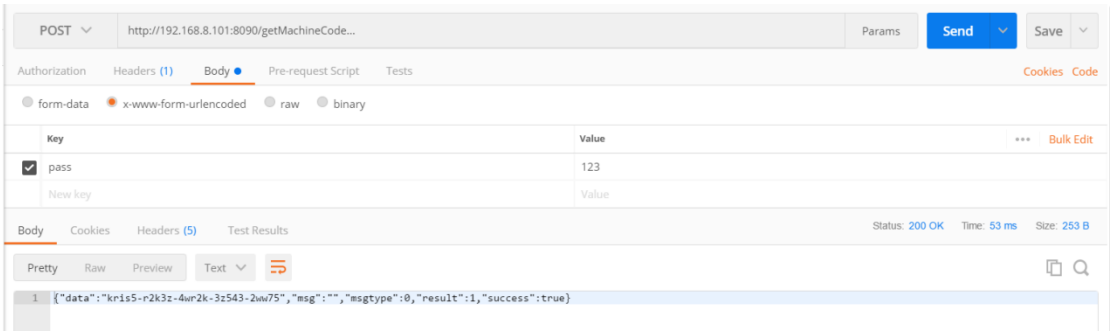
参数名	描述	类型	必传	说明
pass	访问密码	String	Y	

业务字段	描述	类型	附加说明
data	设备的机器码	String	

说明：

- 如果需要设备注册，需要先得到设备的机器码。然后调用设备注册接口（2.1.4）完成设备注册。
- 设备不注册的情况下，只能使用一段时间，一段时间后，则无法进行人脸识别。

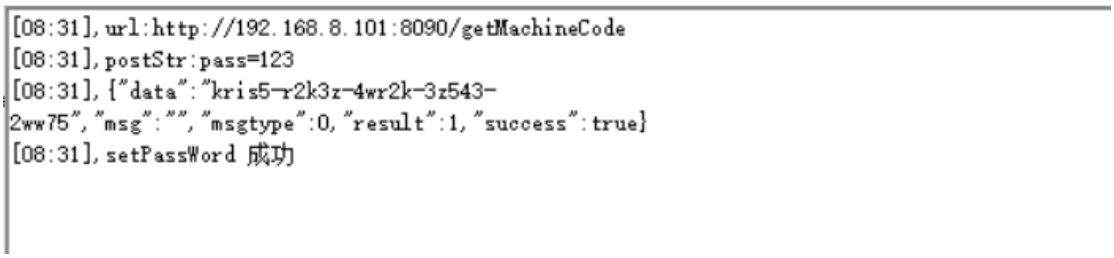
PostMan 示例：



返回示例：

{"data":"kris5-r2k3z-4wr2k-3z543-2ww75","msg":"","msgtype":0,"result":1,"success":true}

Demo示例：



返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功

2.1.4 设备授权

URL：/setAuthorize

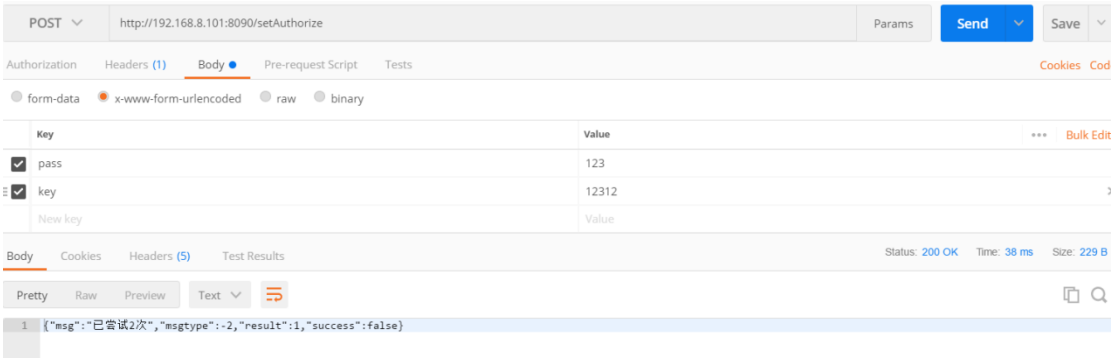
method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
key	授权码	String	Y	

说明：

- 授权码由 xFace 方提供，每个机器码会对应一个授权码
- 授权可以尝试 3 次，3 次后则不能授权

PostMan 示例：



返回示例：

{"msg":"已尝试1次","msgtype":-2,"result":1,"success":false}

Demo示例：

```
[12:40],url:http://192.168.8.101:8090/setAuthorize
[12:40],postStr:pass=123&key=
[12:40],{"msg":"参数异常","msgtype":-1,"result":1,"success":false}
[12:40],有返回，但出错了：参数异常
[12:48],url:http://192.168.8.101:8090/setAuthorize
[12:48],postStr:pass=123&key=123
[12:48],{"msg":"已尝试1次","msgtype":-2,"result":1,"success":false}
[12:48],有返回，但出错了：已尝试1次
```

返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功
-1	参数异常	key为空
-2	已尝试N次	授权码无效

2.1.5 读取管理 URL

URL：/getUrl

method：POST

参数名	描述	类型	必传	说明

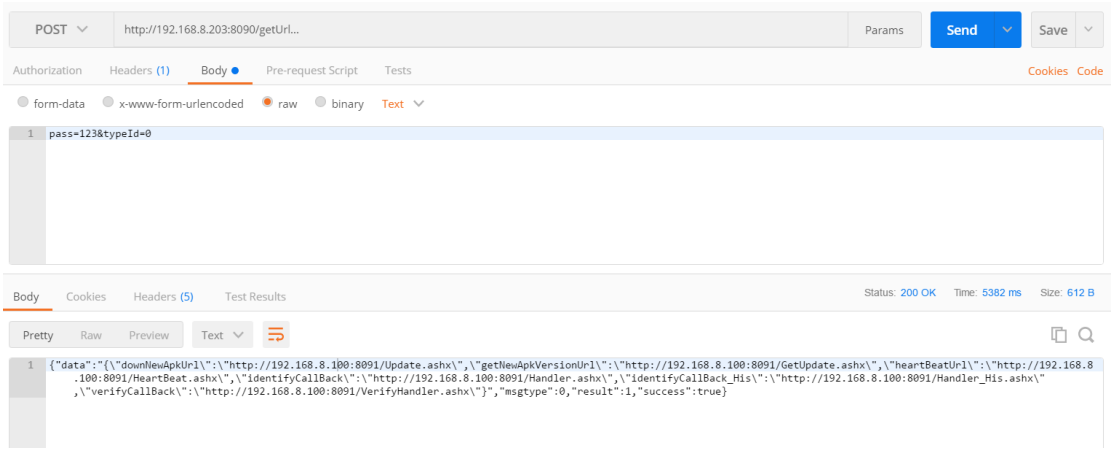
pass	访问密码	String	Y	
typeId	类型	Int	Y	1:识别回调地址 2:查询新版本号地址 3:Apk 更新包下载地址 4:后台验证回调地址 5:心跳包地址 6:历史识别回调地址 0:读取全部
callbackUrl	url 地址	String	Y	可以为空

说明：

- 读取设备上的管理 URL，可配合 2.1.1 使用
- 如果传入 typeid=0 时，则返回 JsonClass，如下：

```
/// <summary>
/// 管理URL类
/// </summary>
public class UrlPar
{
    public string downNewApkUrl { get; set; }
    public string getNewApkVersionUrl { get; set; }
    public string heartBeatUrl { get; set; }
    public string identifyCallBack { get; set; }
    public string identifyCallBack_His { get; set; }
    public string verifyCallBack { get; set; }
}
```

PostMan 示例：

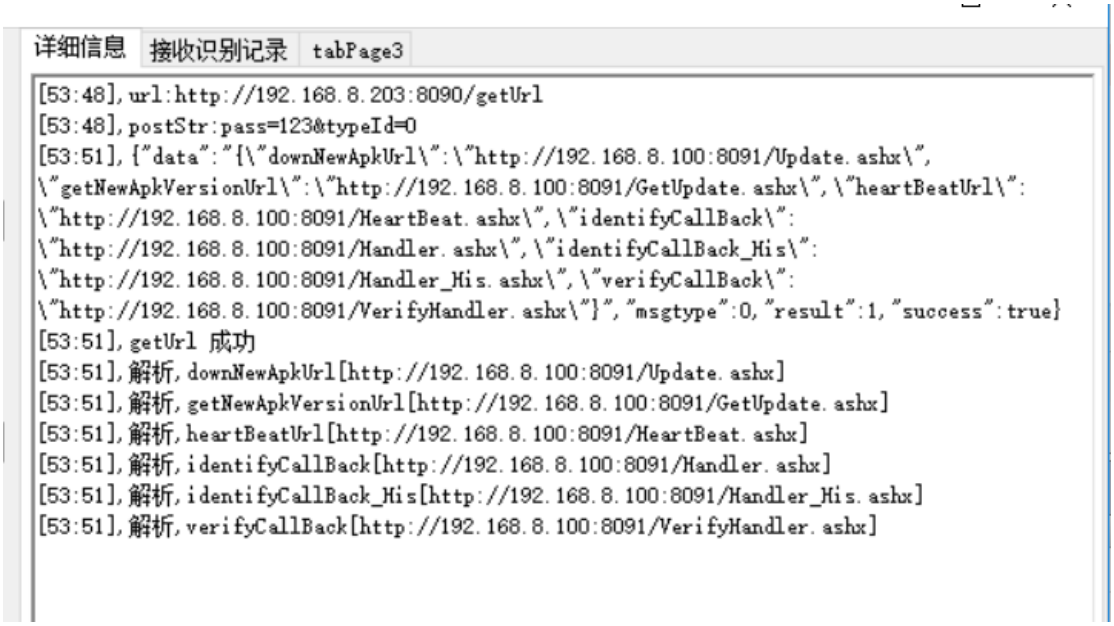


返回示例：

{"data":"{\"downNewApkUrl\":\"http://192.168.8.100:8091/Update.ashx\",\"getNewApkVersionUrl\":\"http://192.168.8.100:8091/GetUpdate.ashx\",\"heartBeatUrl\":\"http://192.168.8.100:8091/

HeartBeat.ashx\",\"identifyCallBack\":\"http://192.168.8.100:8091/Handler.ashx\",\"identifyCallBack_His\":\"http://192.168.8.100:8091/Handler_His.ashx\",\"verifyCallBack\":\"http://192.168.8.100:8091/VerifyHandler.ashx\"}","msgtype":0,"result":1,"success":true}

Demo示例：



返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功

2.1.6 读取运行参数

URL：/getAppConfig

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	

说明：

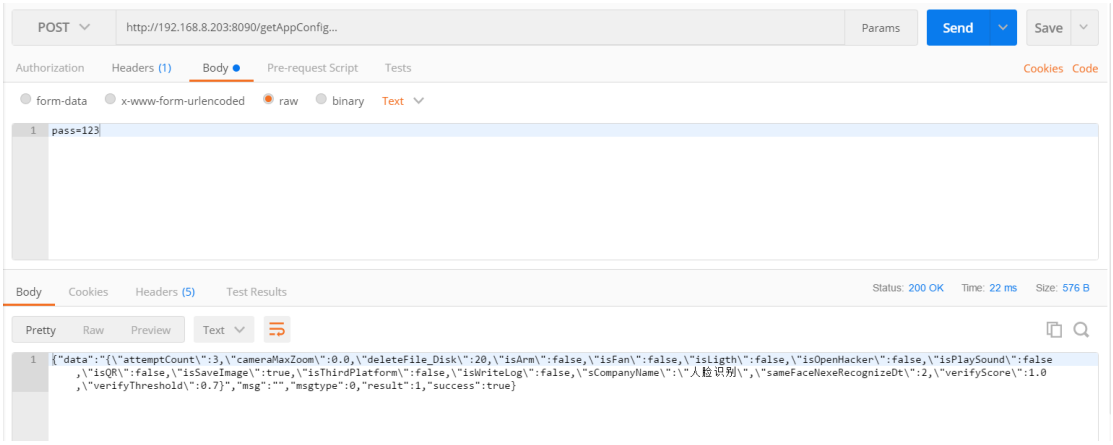
- 得到 APP 的运行参数，是 ClassJsonString 方式返回

```
/// <summary>
/// 运行参数
/// </summary>
public class AppRunConfig
{
```

```
/// <summary>
/// 尝试次数
/// 可选：1,2,3,4,5
/// </summary>
public int attemptCount { get; set; }
/// <summary>
/// 变焦的比例
/// 可选：0，0.5，1,1.5，2
/// </summary>
public float cameraMaxZoom { get; set; }
/// <summary>
/// 保留空间的比例
/// </summary>
public int deleteFile_Disk { get; set; }
/// <summary>
/// 是arm系统
/// </summary>
public bool isArm { get; set; }
/// <summary>
/// 启用灯光
/// </summary>
public bool isLigth { get; set; }
/// <summary>
/// 启用活检
/// </summary>
public bool isOpenHacker { get; set; }
/// <summary>
/// 启用播放语音
/// </summary>
public bool isPlaySound { get; set; }
/// <summary>
/// 启用识别二维码
/// </summary>
public bool isQR { get; set; }
/// <summary>
/// 保存识别照片
/// </summary>
public bool isSaveImage { get; set; }
/// <summary>
/// 启用平台验证
/// </summary>
public bool isThirdPlatform { get; set; }
/// <summary>
/// 启用写log
```

```
        /// 调试问题时使用，工作状态正是不要启用。因为log文件会很大
        /// </summary>
        public bool isWriteLog { get; set; }
        /// <summary>
        /// 公司名称
        /// </summary>
        public string sCompanyName { get; set; }
        /// <summary>
        /// 停留时间
        /// 单位：秒
        /// </summary>
        public int sameFaceNexeRecognizeDt { get; set; }
        /// <summary>
        /// 识别图片时的质量
        /// 可选：0，0.5，1,1.5，2
        /// </summary>
        public float verifyScore { get; set; }
        /// <summary>
        /// 识别阀值
        /// </summary>
        public float verifyThreshold { get; set; }
    }
```

PostMan 示例：



返回示例：

{"data":"{\"attemptCount\":3,\"cameraMaxZoom\":0.0,\"deleteFile_Disk\":20,\"isArm\":false,\"isFan\":false,\"isLigth\":false,\"isOpenHacker\":false,\"isPlaySound\":false,\"isQR\":false,\"isSaveImage\":true,\"isThirdPlatform\":false,\"isWriteLog\":false,\"sCompanyName\":\"人脸识别\",\"sameFaceNexeRecognizeDt\":2,\"verifyScore\":1.0,\"verifyThreshold\":0.7}","msg":"","msgtype":0,"result":1,"success":true}

Demo示例：

2.1.7 设置运行参数

URL：/setAppConfig

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
appconfig	运行参数的jsonString	String	Y	

说明：

- Appconfig 的 josn 对象请参看 2.1.6 说明。
- 设置参数时，请认真确定该参数的可选则的范围。
- 不用重启 APP，参数就能生效

PostMan 示例：

返回示例：

{"msg":"","msgtype":0,"result":1,"success":true}

Demo示例：

返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功

2.1.8 读取推送参数

URL：/getSendConfig

method：POST

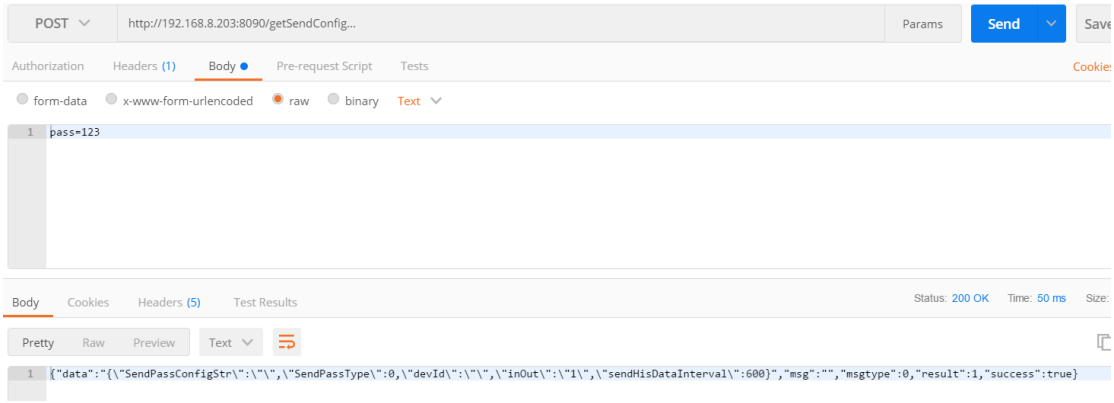
参数名	描述	类型	必传	说明
pass	访问密码	String	Y	

说明：

- 得到 APP 的推送参数，是 ClassJsonString 方式返回

```
/// <summary>
/// 推送参数
/// </summary>
public class AppSendConfig
{
    public string SendPassConfigStr { get; set; }
    public int SendPassType { get; set; }
    /// <summary>
    /// 设备ID
    /// 用户自定义
    /// </summary>
    public string devId { get; set; }
    /// <summary>
    /// 设备方向
    /// 1：进，2：出
    /// </summary>
    public string inOut { get; set; }
    /// <summary>
    /// 历史记录推送时间间隙
    /// 单位：秒
    /// </summary>
    public int sendHisDataInterval { get; set; }
}
```

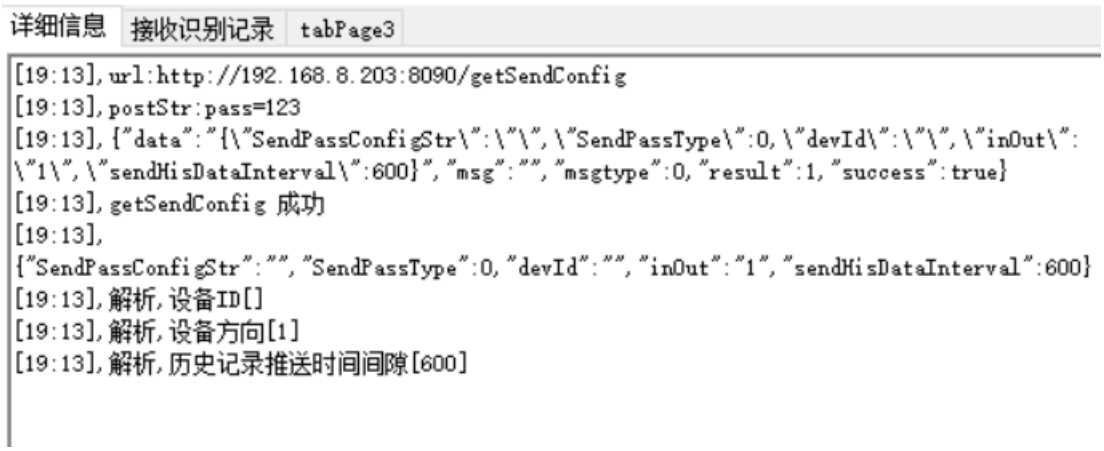
PostMan示例：



返回示例：

{"data":"{\"SendPassConfigStr\":\"\",\"SendPassType\":0,\"devId\":\"\",\"inOut\":\"1\",\"sendHisDataInterval\":600}","msg":"","msgtype":0,"result":1,"success":true}

Demo示例：



2.1.9 设置推送参数

URL：/setSendConfig

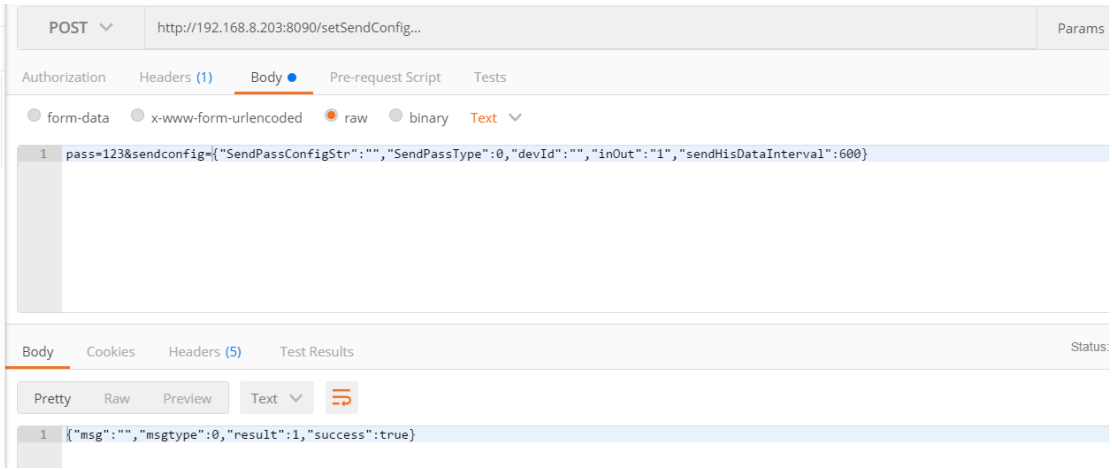
method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
sendconfig	推送参数的jsonString	String	Y	

说明：

- SendConfig 的 josn 对象请参看 2.1.8 说明。
- 设置参数时，请认真确定该参数的可选则的范围。
- 不用重启 APP，参数就能生效

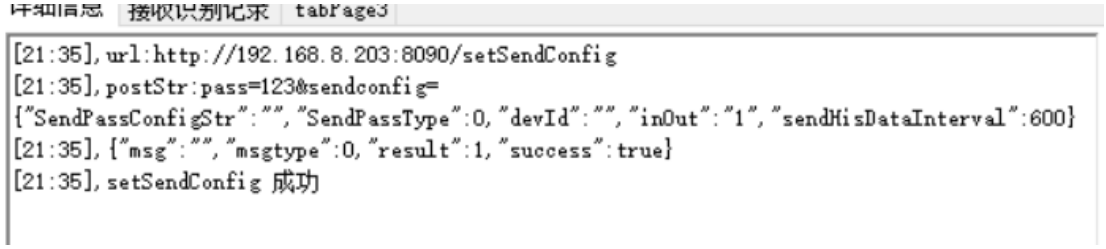
PostMan 示例：



返回示例：

{"msg":"","msgtype":0,"result":1,"success":true}

Demo示例：



返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功

2.1.9 设置推送参数

URL：/setSendConfig

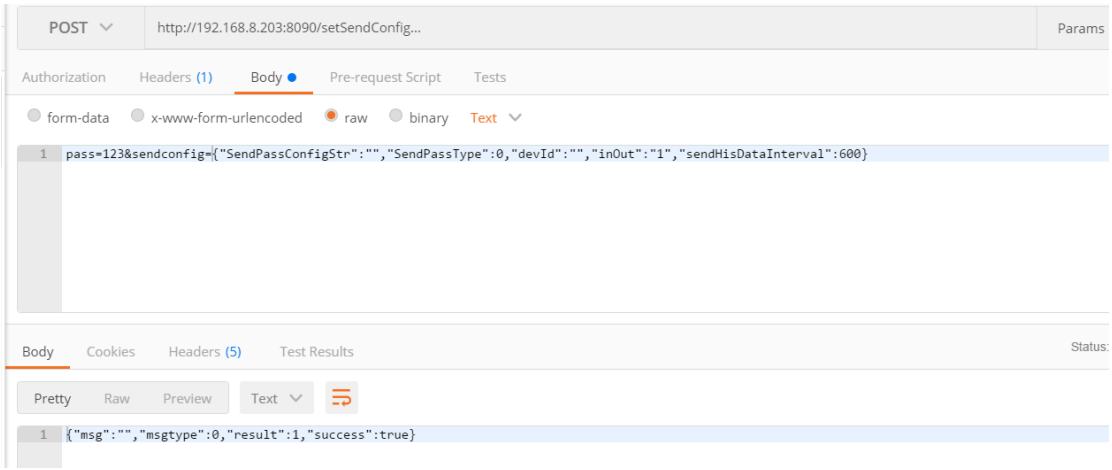
method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
sendconfig	推送参数的jsonString	String	Y	

说明：

- SendConfig 的 josn 对象请参看 2.1.8 说明。
- 设置参数时，请认真确定该参数的可选则的范围。
- 不用重启 APP，参数就能生效

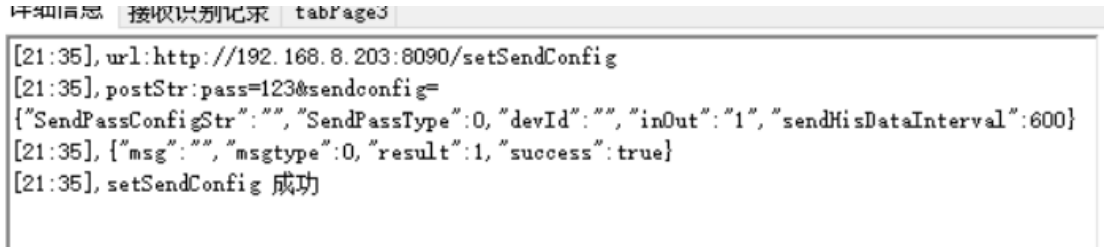
PostMan 示例：



返回示例：

{"msg":"","msgtype":0,"result":1,"success":true}

Demo示例：



返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功

2.1.10 还原出厂设置

URL：/device/reset

method：POST

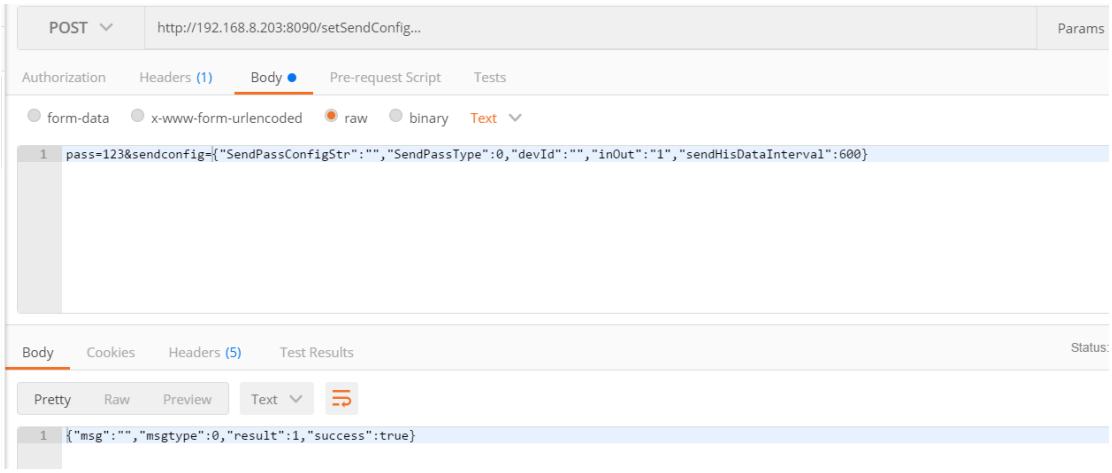
参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
delete	清空参数	String	Y	

说明：

- 还原出厂设置，会把所有的注册人员和识别记录清空

- Delete=true 时，则清空所有参数，参数恢复到默认值。
- 还原成功后，APP 会自动重启
- 如果数据过多，会持续一段时间，直到数据全部清空

PostMan 示例：



返回示例：

Demo示例：

因为还原出厂后会马上重启APP，所以没有返回

2.1.11 重启 APP

URL：/device/restart

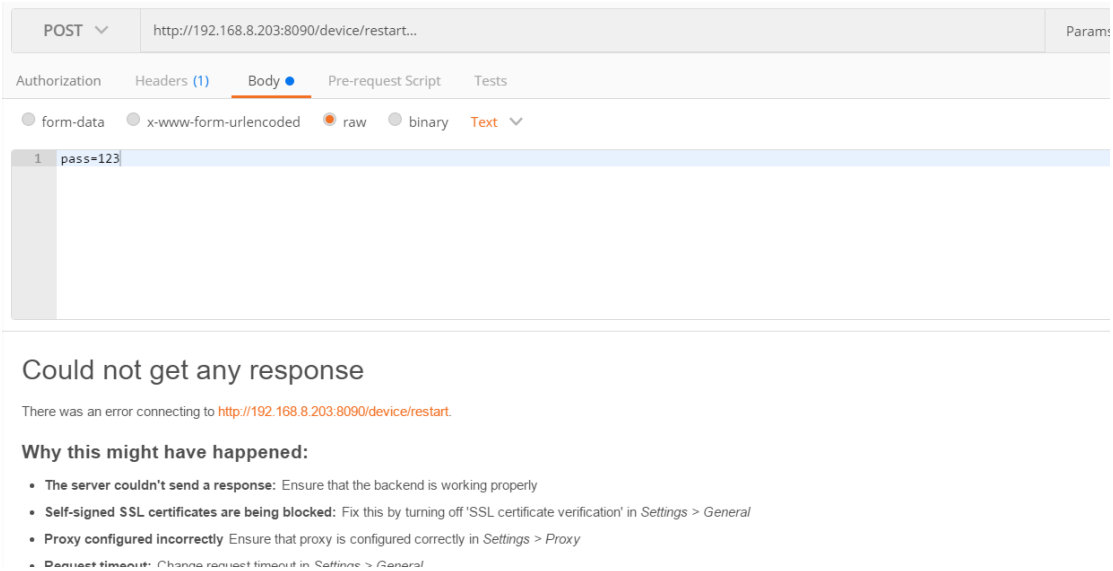
method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	

说明：

- 会重新启动 APP

PostMan 示例：



返回示例：

Demo示例：

因为重启APP，所以没有返回

2.1.12 广播查找设备

以广播形式得到网络上所有运行的设备数据。

请求搜索	**0x10**		
广播端口	**9000**		
请求返回	**0x11**		

- 返回的 json 对象，请参看 2.3.3
- 如果 APP 没有启动，广播不生效
- APP 收到广播后，会在主界面上有提示

Demo示例：

详细信息 接收识别记录 tabPage3

[01:27],设备[192.168.8.151]:[{"availMem":"981.27","buildModel":"MI MAX 2","deviceKey":"35t4w-z9454-twz94-5432v-x92z5","deviceMachineCode":"35t4w-z9454-twz94-5432v-x92z5","deviceType":0,"disk":0.6,"diskInfo":"1.16%[可用0.6G,共51.6G]","faceCount":8,"ip":"192.168.8.151","isAuth":0,"isInited":1,"memory":"59.43","personCount":4,"runtime":"1d0:53","sendCount":2726,"starttime":"2019-03-25 22:08:17","time":"2019-03-26 23:01:46","totalDisk":51.6,"totalMem":3596.5078,"version":"V1.0.6.363"}]
[01:27],解析,机器码[35t4w-z9454-twz94-5432v-x92z5]
[01:27],解析,机器编号[35t4w-z9454-twz94-5432v-x92z5]
[01:27],解析,Ip[192.168.8.151]
[01:27],解析,人员数[4]
[01:27],解析,照片数[8]
[01:27],解析,运行时间[1d0:53]
[01:27],解析,版本号[V1.0.6.363]
[01:27],解析,App占用内存[59.43mb]-[1.65%]
[01:27],解析,系统可用内存[981.27mb]-[27.28%]
[01:27],解析,系统总内存[3596.508mb]

2.2 人员照片同步接口

2.2.1 用户查询接口

URL：/user/findDifference

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
isDelete	直接删除设备上没有的人脸	String	Y	true:删除 false:不删除
imageKeys	人脸照片的特征值	String 用“,”分隔	Y	

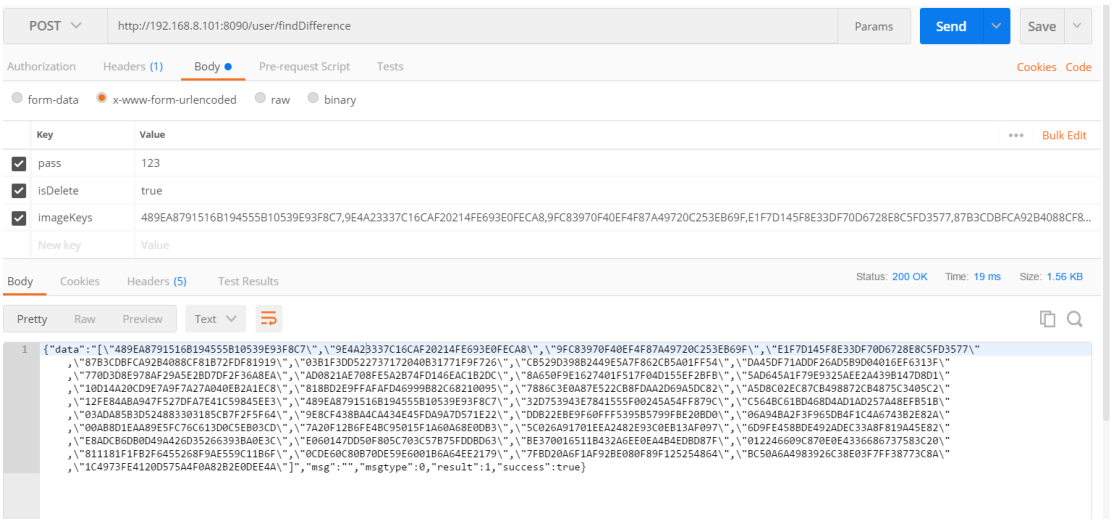
业务字段	描述	类型	附加说明
data	需要增加的照片特征值 list	String 用“,”分隔	提供差异化更新（新增）：可以通过这个字段得到设备中没有，需要增加的照片特征值 list。根据需要调用增加人脸接口。
msg	需要删除的照片特征值 list	String 用“,”分隔	提供差异化更新（新增）：可以通过这个字段得到设备多余，需要删除的照片特征值 list。

			如果 **isDelete=true**,设备直接把多余的照片删除，并返回已经删除的照片特征值 list。 如果 **isDelete=fase**,则不删除，只是返回多余的照片特征值 list.

说明：

- 把人脸照片生成一个照片特征值，建议用 md5。当照片有变化时，则特征值也会随之发生变化。Demo（C#）里的照片特征值生成为 31 位的字符串。
- 建议第三方平台把有效（需要同步的照片）的人脸照片特征值列表加载到内存里，定时进行同步。当有人脸数据发生变化时，更新内存就可以了。
- 此接口对比单个的增加、删除、修改接口有一个很大的好处。就是可以和现有设备进行对比，达到数据准确的同步的效果。设备上多余的数据，可以根据（isDelete=true）直接删除；而没有的照片，可以调用新增接口进行新增。**以往同步经常会出现，需要同步的人脸和设备上的人脸数差 1 个或多个的时候，没办法知道这差异是哪里，所以才特别增加了此接口。**

示例 A：PostMan 示例，当前设备初始化后，没有人脸数据：



Date 有返回，说明需要增加的照片
Msg 为空，说明没有需要删除的照片

此例 Demo 的截图为：从下图可以看出来，需要增加 37 张照片，然后顺序调用增加照片的 API

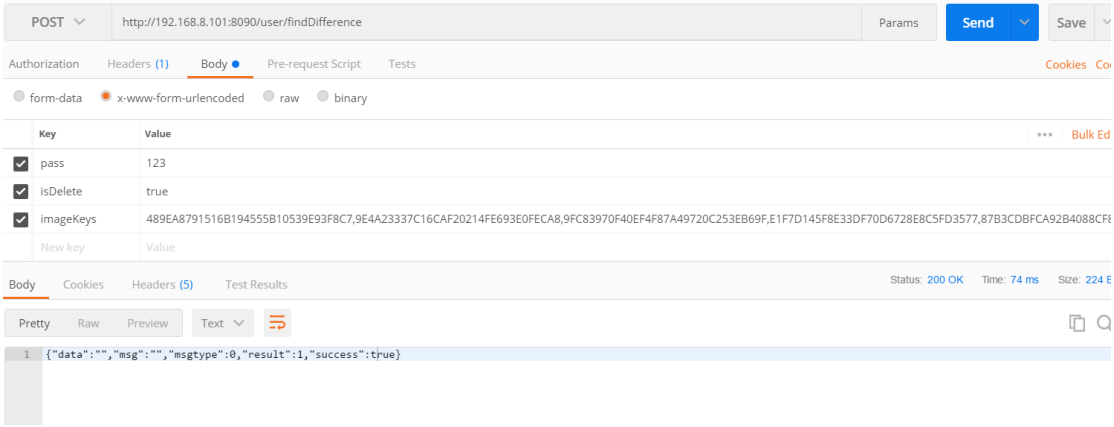
[47:51], {"data":"[\"489EA8791516B194555B10539E93F8C7\", \"9E4A23337C16CAF20214FE693E0FECA8
\", \"9FC83970F40EF4F87A49720C253EB69F\", \"E1F7D145F8E33DF70D6728E8C5FD3577\",
\"87B3CDBFCA92B4088CF81B72FDF81919\", \"03B1F3DD522737172040B31771F9F726\",
\"CB529D398B2449E5A7F862CB5A01FF54\", \"DA45DF71ADDF26AD5B9D04016EF6313F\",
\"770D3D8E978AF29A5E2BD7DF2F36A8EA\", \"AD0821AE708FE5A2B74FD146EAC1B2DC\",
\"8A650F9E1627401F517F04D155EF2BFB\", \"5AD645A1F79E9325AEE2A439B147D8D1\",
\"10D14A20CD9E7A9F7A27A040EB2A1EC8\", \"818BD2E9FFAFAFD46999B82C68210095\",
\"7886C3E0A87E522CB8FDAA2D69A5DC82\", \"A5D8C02EC87CB498872CB4875C3405C2\",
\"12FE84ABA947F527DFA7E41C59845EE3\", \"489EA8791516B194555B10539E93F8C7\",
\"32D753943E7841555F00245A54FF879C\", \"C564BC61BD468D4AD1AD257A48EFB51B\",
\"03ADA85B3D524883303185CB7F2F5F64\", \"9E8CF438BA4CA434E45FDA9A7D571E22\",
\"DDB22EBE9F60FFF5395B5799FBE20BD0\", \"06A94BA2F3F965DB4F1C4A6743B2E82A\",
\"00AB8D1EAA89E5FC76C613D0C5EB03CD\", \"7A20F12B6FE4BC95015F1A60A68E0DB3\",
\"5C026A91701EEA2482E93C0EB13AF097\", \"6D9FE458BDE492ADEC33A8F819A45E82\",
\"E8ADCB6DB0D49A426D35266393BA0E3C\", \"E060147DD50F805C703C57B75FDDBD63\",
\"BE370016511B432A6EE0EA4B4EDBD87F\", \"012246609C870E0E4336686737583C20\",
\"811181F1FB2F6455268F9AE559C11B6F\", \"0CDE60C80B70DE59E6001B6A64EE2179\",
\"7FBD20A6F1AF92BE080F89F125254864\", \"BC50A6A4983926C38E03F7FF38773C8A\",
\"1C4973FE4120D575A4F0A82B2E0DEF4A\"]", "msg":"", "msgtype":0, "result":1, "success":true}
[47:51], 需要增加的记录数: 37
[47:51], 需要删除的记录数: 0
[47:51], 增加照片[1/37], FileName[张洪鹏], 特征值[489EA8791516B194555B10539E93F8C7],
ReturnStr[{"msgtype":0, "result":1, "success":true}]

[47:53], 增加照片[2/37], FileName[张洪鹏], 特征值[9E4A23337C16CAF20214FE693E0FECA8],
ReturnStr[{"msgtype":0, "result":1, "success":true}]

[47:54], 增加照片[3/37], FileName[于明明], 特征值[9FC83970F40EF4F87A49720C253EB69F],
ReturnStr[{"msgtype":0, "result":1, "success":true}]

[47:54], 增加照片[4/37], FileName[仇之怡], 特征值[E1F7D145F8E33DF70D6728E8C5FD3577],

示例 B：PostMan 示例，设备同步照片（新增）后：



输入参数没有变化的情况，可以看到返回的 date,msg 都为空，说明没有需要增加和删除的照片。

Demo 的截图如下：说明用 Demo 完成**示例 A** 后，也就是把照片都同步到设备后。在 Demo

上再点同步则会得到没有任何数据变化的结果，所以可以不做任何操作。


```
[53:43],url:http://192.168.8.101:8090/user/findDifference
[53:43],postStr:pass=123&isDelete=true&imageKeys=
489EA8791516B194555B10539E93F8C7,9E4A23337C16CAF20214FE693E0FECA8,9FC83970F40EF4F87A49720
C253EB69F,E1F7D145F8E33DF70D6728E8C5FD3577,87B3CDBFCA92B4088CF81B72FDF81919,03B1F3DD52273
7172040B31771F9F726,CB529D398B2449E5A7F862CB5A01FF54,DA45DF71ADDF26AD5B9D04016EF6313F,770
03D8E978AF29A5E2BD7DF2F36A8EA,AD0821AE708FE5A2B74FD146EAC1B2DC,8A650F9E1627401F517F04D155
EF2BFB,5AD645A1F79E9325AEE2A439B147D8D1,10D14A20CD9E7A9F7A27A040EB2A1EC8,818BD2E9FFAFAFD4
5999B82C68210095,7886C3E0A87E522CB8FDAA2D69A5DC82,A5D8C02EC87CB498872CB4875C3405C2,12FE84
ABA947F527DFA7E41C59845EE3,489EA8791516B194555B10539E93F8C7,32D753943E7841555F00245A54FF8
79C,C564BC61BD468D4AD1AD257A48EFB51B,03ADA85B3D524883303185CB7F2F5F64,9E8CF438BA4CA434E45
7DA9A7D571E22,DDB22EBE9F60FFF5395B5799FBE20BD0,06A94BA2F3F965DB4F1C4A6743B2E82A,00AB8D1EA
A89E5FC76C613D0C5EB03CD,7A20F12B6FE4BC95015F1A60A68E0DB3,5C026A91701EEA2482E93C0EB13AF097
6D9FE458BDE492ADEC33A8F819A45E82,E8ADCB6DB0D49A426D35266393BA0E3C,E060147DD50F805C703C57
375FDDBD63,BE370016511B432A6EE0EA4B4EDBD87F,012246609C870E0E4336686737583C20,811181F1FB2F
5455268F9AE559C11B6F,0CDE60C80B70DE59E6001B6A64EE2179,7FBD20A6F1AF92BE080F89F125254864,BC
50A6A4983926C38E03F7FF38773C8A,1C4973FE4120D575A4F0A82B2E0DEE4A
[53:43],{"data":"","msg":"","msgtype":0,"result":1,"success":true}
[53:43],需要增加的记录数：0
[53:43],需要删除的记录数：0
```


返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功
-1	参数异常	imageKeys参数为''

2.2.2 用户增加、更新

URL：/user/createOrUpdate

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
user	用户对象	JsonString	Y	用户对象转成的 Json 字符串

以下是 user 类的定义

```
/// <summary>
/// 人员，照片类
/// </summary>
public class User
{
    /// <summary>
    /// 用户ID，唯一标识
```

```
        /// </summary>
        public string userId { get; set; }
        /// <summary>
        /// 用户名字
        /// </summary>
        public string userName { get; set; }
        /// <summary>
        /// 用户特征值
        /// </summary>
        public string userKey { get; set; }
        /// <summary>
        /// 照片的ID
        /// </summary>
        public string imageId { get; set; }
        /// <summary>
        /// 照片的特征值
        /// </summary>
        public string imageKey { get; set; }
        /// <summary>
        /// 照片的Base64数据
        /// </summary>
        public string imageBase64 { get; set; }
        /// <summary>
        /// 照片的序号，默认为0，可输入0或1
        /// </summary>
        public int direct { get; set; }
    }
```

说明：

- 设备会按 userId 进行更新，如果没有此 ID 的数据，则新增。如果存在，则更新
- Direct 代表照片位置，一个人可以有 1-2 个照片。

返回示例：

{"msgtype":0,"result":1,"success":true}

Demo示例：

[47:51],需要增加的记录数：37
[47:51],需要删除的记录数：0
[47:51],增加照片[1/37],FileName[张洪鹏],特征值[489EA8791516B194555B10539E93F8C7],
ReturnStr[{"msgtype":0,"result":1,"success":true}]

[47:53],增加照片[2/37],FileName[张洪鹏],特征值[9E4A23337C16CAF20214FE693E0FECA8],
ReturnStr[{"msgtype":0,"result":1,"success":true}]

[47:54],增加照片[3/37],FileName[于明明],特征值[9FC83970F40EF4F87A49720C253EB69F],
ReturnStr[{"msgtype":0,"result":1,"success":true}]

[47:54],增加照片[4/37],FileName[仇之怡],特征值[E1F7D145F8E33DF70D6728E8C5FD3577],
ReturnStr[{"msgtype":0,"result":1,"success":true}]

[47:54],增加照片[5/37],FileName[何永森],特征值[87B3CDBFCA92B4088CF81B72FDF81919],
ReturnStr[{"msgtype":0,"result":1,"success":true}]

[47:55],增加照片[6/37],FileName[刘小夯],特征值[03B1F3DD522737172040B31771F9F726],
ReturnStr[{"msgtype":0,"result":1,"success":true}]

[47:55],增加照片[7/37],FileName[刘桂影],特征值[CB529D398B2449E5A7F862CB5A01FF54],
ReturnStr[{"msgtype":0,"result":1,"success":true}]

[47:56],增加照片[8/37],FileName[刘海兵],特征值[DA45DF71ADDF26AD5B9D04016EF6313F],
ReturnStr[{"msgtype":0,"result":1,"success":true}]

返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功
-1	参数异常	user为空字符
-2	参数异常	user字符串转化成User对象异常
-3	参数异常	Base64数据无法转化成照片格式
-4	参数异常	其它，系统异常

2.2.3 用户时段权限-新增或删除

URL：/pastime/createOrUpdate

method：POST

人脸界面：可以在“设置”→“时段管理” 里查看时段信息

参数名	描述	类型	必传	说明

pass	访问密码	String	Y	
passtimes	时段对象	JsonObject	Y	

passtime 详细说明：

- 可以调协多个时段
- 每个段可以定义多个星期[1,7]和多个时间段
- 时段段的格式为：HH:mm:ss

对象类定义

```
public class PassTimes
{
    /// <summary>
    /// 时段名称
    /// </summary>
    public string Name { get; set; }
    /// <summary>
    /// 时段列表
    /// </summary>
    public List<PassTime> passTimeList { get; set; }
}
/// <summary>
/// 时段，有星期列表和时间段列表
/// </summary>
public class PassTime
{

    /// <summary>
    /// 星期列表 值在[1, 7]
    /// </summary>
    public List<String> WeekList;
    /// <summary>
    /// 时间段列表，可以多个
    /// </summary>
    public List<PassTimeOne> PassTimeByWeekList;
}
/// <summary>
/// 时段段对象
/// </summary>
public class PassTimeOne
{
    /// <summary>
    /// 开始时间 hh:mi:ss
    /// </summary>
    public String Dt1;
```

```
        /// <summary>
        /// 结果时间 hh:mi:ss
        /// </summary>
        public String Dt2;
        }
```

以下是 passtim 类的定义

```
    /// <summary>
    /// 人员通过时段类
    /// </summary>
    public class pastime
    {
        /// <summary>
        /// 用户ID，唯一标识
        /// </summary>
        public string userId { get; set; }
        /// <summary>
        /// 通过的时间段
        /// </summary>
        public string passtime{ get; set; }

    }
```

给对象赋值的方法，在 Demo Code 里可以看到：

```
         /// <summary>
        /// 营业的时段数据
        /// 周1-7
        /// 时段:08:00:00-23:00:00
        /// </summary>
        /// <returns></returns>
        private PassTimes GetNewPassTimes2()
        {
            PassTimes res = new PassTimes();

            res.Name = "正常营业";
            res.passTimeList = new List<PassTime>();
            PassTime _PassTime = new PassTime();
            _PassTime.WeekList = new List<string>();
            _PassTime.PassTimeByWeekList = new List<PassTimeOne>();
            _PassTime.WeekList.Add("1");
            _PassTime.WeekList.Add("2");
            _PassTime.WeekList.Add("3");
            _PassTime.WeekList.Add("4");
            _PassTime.WeekList.Add("5");
```

```
            _PassTime.WeekList.Add("6");
            _PassTime.WeekList.Add("7");

            //时段
            PassTimeOne _PassTimeOne = new PassTimeOne();
            _PassTimeOne.Dt1 = "08:00:00";
            _PassTimeOne.Dt2 = "23:00:00";
            _PassTime.PassTimeByWeekList.Add(_PassTimeOne);
            //加入
            res.passTimeList.Add(_PassTime);
            return res;
        }
```

返回示例：


```
{"msg":"","msgtype":0,"result":1,"success":true}
```


Demo示例：


```
[36:13],url:http://192.168.8.101:8090/passtime/createOrUpdate
[36:13],postStr:pass=123&passtimes={"Name":"住宿生","passTimeList":[{"WeekList":
["5"],"PassTimeByWeekList":[{"Dt1":"12:00:00","Dt2":"13:00:00"},
{"Dt1":"17:00:00","Dt2":"18:00:00"}]},{"WeekList":["7"],"PassTimeByWeekList":
[{"Dt1":"17:00:00","Dt2":"18:00:00"}]}]}
[36:13],{"msgtype":0,"result":1,"success":true}
[36:13],Set passtime[住宿生] 成功
[36:13],url:http://192.168.8.101:8090/passtime/createOrUpdate
[36:13],postStr:pass=123&passtimes={"Name":"正常营业","passTimeList":[{"WeekList":
["1","2","3","4","5","6","7"],"PassTimeByWeekList":
[{"Dt1":"08:00:00","Dt2":"23:00:00"}]}]}
[36:13],{"msgtype":0,"result":1,"success":true}
[36:13],Set passtime[正常营业] 成功
[36:13],url:http://192.168.8.101:8090/passtime/createOrUpdate
[36:13],postStr:pass=123&passtimes={"Name":"上下班(学)","passTimeList":[{"WeekList":
["1","2","3","4","5"],"PassTimeByWeekList":[{"Dt1":"07:00:00","Dt2":"08:00:00"},
{"Dt1":"11:00:00","Dt2":"13:00:00"},{"Dt1":"16:00:00","Dt2":"18:00:00"}]}]}
[36:13],{"msgtype":0,"result":1,"success":true}
[36:13],Set passtime[上下班(学)] 成功
```


返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功
-1	参数异常	Passtimes 为空
-2	参数异常	passTimeList 转化异常
-3	参数异常	系统异常

2.2.4 用户时段权限-删除

URL：/pastime/delete

method：POST

人脸界面：

- **可以在“设置”→“时段管理” 里查看时段信息；**
- **如果有人员此时段，然后调用此接口删除时段，则人员对应的时段会变成“未知时段”将可以 7*24 通过。**

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
passtimename	时段名称	String	Y	

返回示例：

{"msg":"passtime 添加成功","msgtype":0,"result":1,"success":true}

Demo示例：

```
[45:11],url:http://192.168.8.101:8090/passtime/delete
[45:11],postStr:pass=123&passtimename=住宿生
[45:11],{"msgtype":0,"result":1,"success":true}
[45:11],passtime delete[住宿生] 成功
```

返回结果说明：

msgtype	**msg**	**结果或错误原因**
0	无	成功
-1	参数异常	passtimename 为空

2.2.5 人员设置时间

URL：/user/setpasstime

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
usersetpasstime	用户对象	JsonObjectString	Y	

以下是 usersetpasstime 类的定义

```
public class UserSetPassTime
{
    /// <summary>
    /// 用户ID
    /// </summary>
    public string userId { get; set; }
    /// <summary>
    /// 时段的名称
    /// </summary>
    public string passTimeName { get; set; }
}
```

说明：

- 人员时段信息，可以在 APP 的“设置”-“用户管理”里查看
- 如果人员设置时段后，此时段被删除，则此人员会变归属“未知时段”，7*24 可以通过
- 时段名可以传””，此人就变成“未知时段”，7*24 通过
- 如果人员不在设置的时段内，则会有在设置上有异常提示

返回示例：

```
{"msgtype":0,"result":1,"success":true}
```

Demo示例：

```
[51:13],url:http://192.168.8.101:8090/user/setpasstime
[51:13],postStr:pass=123&usersetpasstime={"userId":"zhp","passTimeName":"住宿生"}
[51:13],{"msgtype":0,"result":1,"success":true}
[51:13],set user passtime 成功
[51:13],url:http://192.168.8.101:8090/refresh
[51:13],postStr:pass=123
[51:13],{"msg":"36,37,2","msgtype":0,"result":1,"success":true}
[51:13],refresh 成功:36,37,2
```

返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功
-1	参数异常	usersetpasstime 为空
-2	参数异常	对 usersetpasstime 做 JSON 转化异常
-3	参数异常	系统异常

2.2.6 数据刷新

URL：/ refresh

method：POST

接口说明：在调用人员接口时，设备里的数据不会马上刷新。需要在处理完人员和照片后，调用此接口。

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	

Msg **说明：**

- Msg 会返回刷新后的数据列表的数量
- 用”,”分隔
- 先后为用户数量，人脸照片数量，时段数量

返回示例：

{"msg":"36,37,3","msgtype":0,"result":1,"success":true}

Demo示例：

```
[51:13],url:http://192.168.8.101:8090/user/setpasstime
[51:13],postStr:pass=123&usersetpasstime={"userId":"zhp","passTimeName":"住宿生"}
[51:13],{"msgtype":0,"result":1,"success":true}
[51:13],set user passtime 成功
[51:13],url:http://192.168.8.101:8090/refresh
[51:13],postStr:pass=123
[51:13],{"msg":"36,37,2","msgtype":0,"result":1,"success":true}
[51:13],refresh 成功:36,37,2
```

返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功

2.2.7 人员增加、更新

URL：/person/createOrUpdate

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
person	人员对象	JsonString	Y	人员对象转成的 Json 字符串

以下是 person 类的定义

```
/// <summary>
    /// 人员对象
    /// </summary>
    public class Person
    {
        /// <summary>
        /// ID
        /// 如果传入，格式：[A-Za-z0-9]{0,32}
        /// 可以为空，系统会自动生成ID，并在返回数据中体现
        /// </summary>
        public string id { get; set; }
        /// <summary>
        /// 姓名
        /// </summary>
        public string name { get; set; }

        public override string ToString()
        {
            return string.Format("id:[{0}],name:[{1}]",id,name);
        }
    }
```

说明：

- 设备会按 id 进行更新，如果没有此 ID 的数据，则新增。如果存在，则更新
- 如果传入 id 为空，则系统会自动生成一个 id,并在返回的 data 里体现
- 如果传入的 id 非空，必须符合长度为 32，由大小写字母和数据组合。

返回示例：

传入 id:”1”,name:”1”的返回结果：

{"data":"{\"id\":\"1\",\"name\":\"1\"}","msg":"","msgtype":0,"result":1,"success":true}

传入 id:””,name:”1”的返回结果：

{"data":"{\"id\":\"b00e37068f35401baadf09e615e0d38d\",\"name\":\"1\"}","msg":"","msgtype":0,"result":1,"success":true}

Demo示例：

传入 id:”1”,name:”1”的返回结果：

```
[59:54],url:http://192.168.8.101:8090/person/createOrUpdate
[59:54],postStr:pass=123&person={"id":"1","name":"1"}
[59:54],{"data":"{\"id\":\"1\",\"name\":\"1\"}","msg":"","msgtype":0,"result":1,"success":true}
[59:54],person createOrUpdate 成功
[59:54],url:http://192.168.8.101:8090/refresh
[59:54],postStr:pass=123
[59:54],{"msg":"0,0,3","msgtype":0,"result":1,"success":true}
[59:54],refresh 成功:0,0,3
```

传入 id:””,name:”1”的返回结果：

```
[00:17],url:http://192.168.8.101:8090/person/createOrUpdate
[00:17],postStr:pass=123&person={"id":"","name":"1"}
[00:17],{"data":"{\"id\":\"b00e37068f35401baadf09e615e0d38d\",\"name\":\"1
\"}","msg":"","msgtype":0,"result":1,"success":true}
[00:17],person createOrUpdate 成功
[00:17],url:http://192.168.8.101:8090/refresh
[00:17],postStr:pass=123
[00:17],{"msg":"0,0,3","msgtype":0,"result":1,"success":true}
[00:17],refresh 成功:0,0,3
```

返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功
-1	参数异常	person为空字符
-2	参数异常	person字符串转化成User对象异常
~~-3~~	~~参数异常~~	~~id的格式不正确~~
-4	参数异常	其它，系统异常

2.2.8 人员查询

URL：/person/find

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
id	人员 id	String	Y	

说明：

- 人员 id=”-1”时，则查询所有人员
- 返回的 data 里数据列表

返回示例：

```
{"data":"[{\"id\":\"1\",\"name\":\"1\"}]","msg":"","msgtype":0,"result":1,"success":true}
```

Demo示例：

```
[00:36],url:http://192.168.8.101:8090/person/find
[00:36],postStr:pass=123&id=1
[00:36],{"data":"[{\"id\":\"1\",\"name\":\"1
\"}]","msg":"","msgtype":0,"result":1,"success":true}
[00:36],person find 成功
[00:36],person[1]:id:[1],name:[1]

[01:24],url:http://192.168.8.101:8090/person/find
[01:24],postStr:pass=123&id=-1
[01:24],{"data":"[{\"id\":\"1\",\"name\":\"1\"},{\"id\":\"b00e37068f35401baadf09e615e0d38d\",
\"name\":\"1\"}]","msg":"","msgtype":0,"result":1,"success":true}
[01:24],person find 成功
[01:24],person[1]:id:[1],name:[1]
[01:24],person[2]:id:[b00e37068f35401baadf09e615e0d38d],name:[1]

[01:36],url:http://192.168.8.101:8090/person/find
[01:36],postStr:pass=123&id=22
[01:36],{"data":"","msg":"","msgtype":0,"result":1,"success":true}
[01:36],person find 成功
[01:36],无数据
```

返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功
-1	参数异常	id为空

2.2.9 人删除

URL：/person/delete

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
id	人员 id	String	Y	

说明：

- 人员 id=”-1”时，则删除所有人员和所有照片
- 人员 id 可以传入多个，用”,”分隔
- 返回数据中 data 会有删除成功的的 id 列表，如果传入的要删除 id 在设备里没有，则不会在返回 data 数据项里体现

返回示例：

{"data":"[1]","msg":"delete count:1","msgtype":0,"result":1,"success":true}

Demo示例：

```
[07:26],url:http://192.168.8.101:8090/person/delete
[07:26],postStr:pass=123&id=1,2,3
[07:26],{"data":"[1]","msg":"delete count:1","msgtype":0,"result":1,"success":true}
[07:26],person delete 成功
[07:26],url:http://192.168.8.101:8090/refresh
[07:26],postStr:pass=123
[07:26],{"msg":"0,0,3","msgtype":0,"result":1,"success":true}
[07:26],refresh 成功:0,0,3
```

返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功
-1	参数异常	id为空
-2	参数异常	Id用","分隔转化成数组出错

2.2.10 照片增加、更新

URL：/face/createOrUpdate

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
face	照片对象	JsonString	Y	照片对象转成的 Json 字符串

以下是 Face 类的定义

```
public class Face
    {
        /// <summary>
        /// 人员id
        /// </summary>
        public string userId { get; set; }
        /// <summary>
        /// 人员姓名，不更新人员数据
        /// 只是根据此数据生成设备注册照片名
        /// </summary>
        public string userName { get; set; }
```

```
        //public string faceId { get; set; }
        /// <summary>
        /// 人员照片的序号
        /// </summary>
        public int direct { get; set; }
        /// <summary>
        /// 照片ID，唯一
        /// </summary>
        public string imageId { get; set; }
        /// <summary>
        /// 照片key,可用照片的md5值
        /// </summary>
        //public string faceImageFileName { get; set; }
        public string imageKey { get; set; }
        /// <summary>
        /// 照片base64
        /// </summary>
        public string imageBase64 { get; set; }
    }
```

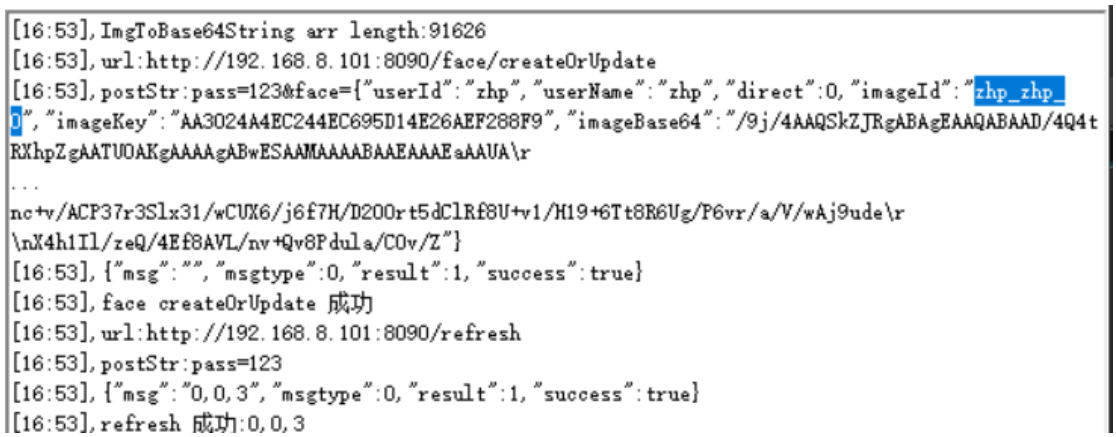
说明：

- 设备会按 imageId 进行更新，如果没有此 ID 的数据，则新增。如果存在，则更新

返回示例：

{"msg":"","msgtype":0,"result":1,"success":true}

Demo示例：



```
[16:53],ImgToBase64String arr length:91626
[16:53],url:http://192.168.8.101:8090/face/createOrUpdate
[16:53],postStr:pass=123&face={"userId":"zhp","userName":"zhp","direct":0,"imageId":"zhp_zhp_
0","imageKey":"AA3024A4EC244EC695D14E26AEF288F9","imageBase64":"/9j/4AAQSkZJRgABAgEAAQABAAD/4Q4t
RXhpZgAATUOAKgAAAAgABwESAAMAAAABAAEAAAEaAAUA\r
...
nc+v/ACP37r3Slx31/wCUX6/j6f7H/D2OOrt5dClRf8U+v1/H19+6Tt8R6Ug/P6vn/Aj9ude\r
\nX4h1Il/zeQ/4Ef8AVL/nv+Qv8Pdula/COv/Z"}
[16:53],{"msg":"","msgtype":0,"result":1,"success":true}
[16:53],face createOrUpdate 成功
[16:53],url:http://192.168.8.101:8090/refresh
[16:53],postStr:pass=123
[16:53],{"msg":"0,0,3","msgtype":0,"result":1,"success":true}
[16:53],refresh 成功:0,0,3
```

返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功

-1	参数异常	face为空字符
-2	参数异常	face字符串转化成Face对象异常
-3	参数异常	Face. imageBase64无法转化成image文件
-4	参数异常	其它，系统异常

2.2.10 照片查询

URL：/face/find

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
id	照片 id	String	Y	

说明：

- 照片 id=”-1”时，则查询所有人员
- 返回的 data 里数据列表

返回示例：

{"data":"[{\"id\":\"1\",\"name\":\"1\"}]","msg":"","msgtype":0,"result":1,"success":true}

Demo示例：

```
[20:43],url:http://192.168.8.101:8090/face/find
[20:43],postStr:pass=123&id=zhp_zhp_0
[20:43],{"data":"[{\"direct\":0,\"faceId\":\"zhp_zhp_0\",\"faceImageKey\":
\"AA3024A4EC244EC695D14E26AEF288F9\",\"personId\":\"zhp
\"}]","msg":"","msgtype":0,"result":1,"success":true}
[20:43],face find 成功
[20:43],face[1]:faceId:[zhp_zhp_0],personId:[zhp],direct:[0],faceImageFaileName:[],faceImageKey:
[AA3024A4EC244EC695D14E26AEF288F9]
```

返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功
-1	参数异常	id为空

2.2.11 照片删除

URL：/face/delete

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	
id	照片 id	String	Y	

说明：

- 照片 id=”-1”时，则删除所有照片
- 照片 id 只能传一个，不可以传多个

返回示例：

{"msg":"","msgtype":0,"result":1,"success":true}

Demo示例：

```
[22:48],url:http://192.168.8.101:8090/face/delete
[22:48],postStr:pass=123&faceId=zhp_zhp_0
[22:48],{"msg":"","msgtype":0,"result":1,"success":true}
[22:48],face delete 成功
[22:48],url:http://192.168.8.101:8090/refresh
[22:48],postStr:pass=123
[22:48],{"msg":"0,0,3","msgtype":0,"result":1,"success":true}
[22:48],refresh 成功:0,0,3

[22:55],url:http://192.168.8.101:8090/face/find
[22:55],postStr:pass=123&id=zhp_zhp_0
[22:55],{"data":"","msg":"","msgtype":0,"result":1,"success":true}
[22:55],face find 成功
[22:55],无数据
```

返回结果说明：

msgtype	msg	结果或错误原因
0	无	成功
-1	参数异常	id为空

2.2.2 得到当天识别记录情况

URL：/device/getTodayRecord

method：POST

参数名	描述	类型	必传	说明
pass	访问密码	String	Y	

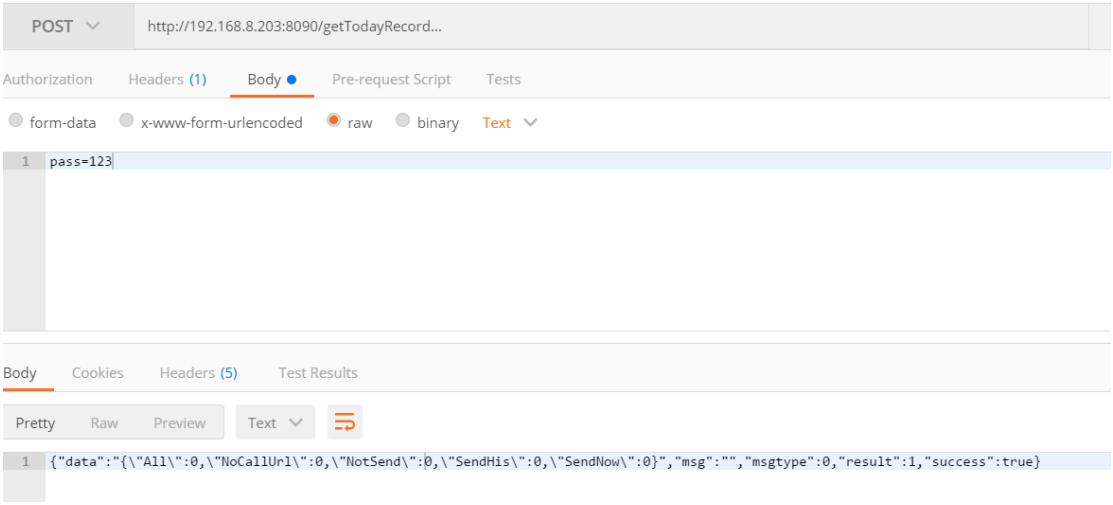
说明：

- 会得到相应的当天的识别记录的情况
- 以 ClassJosnString 返回，类对象如下：

```
/// <summary>
    /// 当天识别的记录情况
    /// </summary>
    public class TodayRecord
    {
        /// <summary>
        /// 当天所有识别记录数
        /// </summary>
        public int All;
        /// <summary>
        /// 当天实时推送记录数
        /// </summary>
        public int SendNow;
        /// <summary>
        /// 当天历史推送记录数
        /// </summary>
        public int SendHis;
        /// <summary>
        /// 当天未推送记录数
        /// </summary>
        public int NotSend;
        /// <summary>
        /// 当天识别记录，但因为未设置回调URL，则不用回调
        /// </summary>
        public int NoCallUrl;


    }
```

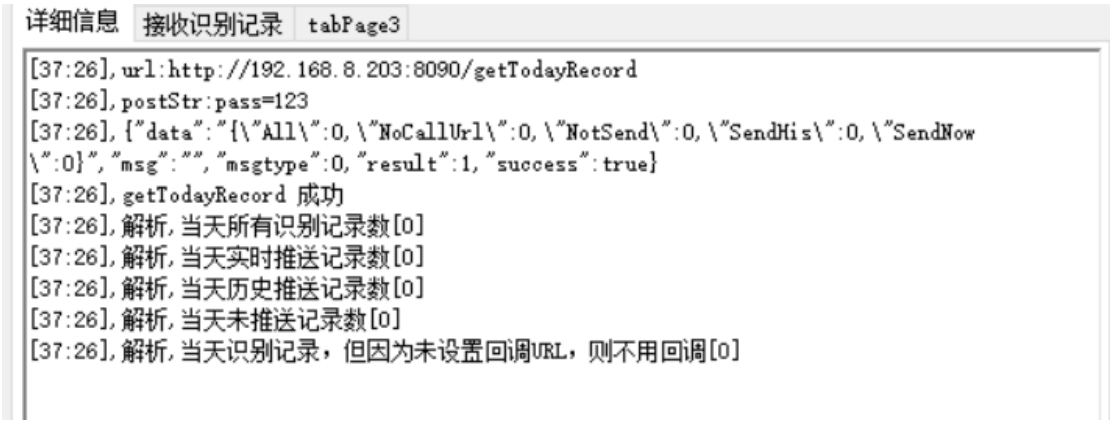
PostMan 示例：



返回示例：

{"data":"{\"All\":0,\"NoCallUrl\":0,\"NotSend\":0,\"SendHis\":0,\"SendNow\":0}","msg":"","msgtype":0,"result":1,"success":true}

Demo示例：



2.3 回调接口

2.3.1 回调方法说明

当有人脸识别后（不管成功或失败），会调用回调方法。

URL：/???，在 2.1.2 里设置 type=1

method：POST

参数名	描述	类型	必传	说明
verify	验证 Json 对象	JsonString	Y	

```
/// <summary>
        /// 返回对象
        /// </summary>
        public class VerifyReturn
        {
            public int result { get; set; }
            public bool success { get; set; }

            public int msgtype { get; set; }
            public string msg { get; set; }

            public object data { get; set; }
        }
        /// <summary>
        /// 识别结果数据对象
        /// </summary>
        public class Verify
        {
            /// <summary>
            /// 数据的标识，可以用来滤重
            /// 也可用id滤重，得考虑设备会被还原出厂，ID有可以会从1重新开始，所以增加了
此数据项
            /// </summary>
            public string guid { get; set; }
            /// <summary>
            /// 本地记录的ID
            /// </summary>
            public int id { get; set; }
            /// <summary>
            /// 设备编号，用户可以自定义
            /// </summary>
            public string deviceKey { get; set; }
            /// <summary>
            /// 机器码
            /// </summary>
            public string machineCode { get; set; }
            /// <summary>
            /// 识别人员编号
            /// 特别说明
            ///     type=2（活检未通过）时，是用来被作弊人员的ID，也就是照片中的人
            ///     type=3(未注册)时，在系统中最像的人，但未达到阈值
            /// </summary>
            public string userId { get; set; }
            /// <summary>
```

```
        /// 识别人员姓名
        /// </summary>
        public string userName { get; set; }
        /// <summary>
        /// 识别结果
        /// 1，识别正常，mes=姓名,矩形框为绿色
        /// 2，活检未通过，mes=提示信息，矩形框为黄
        /// 3，未注册，mes=提示信息，矩形框为红色
        /// 4，找到人员，后台验证无反应 蓝色
        /// 5，找到人员，后台验证失败 蓝色
        /// 6，找到人员，未授权不能通过（通过时段判断） 蓝色
        /// </summary>
        public int type { get; set; }
        /// <summary>
        /// 识别抓拍时照片的Ftp
        /// </summary>
        public string path { get; set; }
        /// <summary>
        /// 识别抓拍时照片的base64
        /// </summary>
        public string base64 { get; set; }

        /// <summary>
        /// 记录类型
        /// 0：实时
        /// 1：历史
        /// </summary>
        public int SendPassType { get; set; }
        public override string ToString()
        {
            return String.Format("id[{0}],type[{1}],
userId[{2}],userName[{3}],sendPassType[{5}]" + (!string.IsNullOrEmpty(path) ?
Environment.NewLine : "") + " path[{4}]",
                id, type, userId, userName, path, SendPassType == 0 ? "实时" : "历史
");
        }
        /// <summary>
        /// 方向
        /// 1：进
        /// 2：出
        /// </summary>
        public int direction { get; set; }
        /// <summary>
        /// 识别时的时间
```

```
        /// </summary>
        public string time { get; set; }
        /// <summary>
        /// 推送批次
        /// 历史退送时，会有此字段
        /// 格式：yyyy-mmm-dd_推送次数_本批次需要上传的记录数
        /// </summary>
        public string banch { get; set; }

        /// <summary>
        /// 识别人脸分数
        /// </summary>
        public float score { get; set; }

    }
```

说明：

- 只要有人脸进入设备识别区域后，都会产生一条认别记录，并调用回调方法。
- 可以根据 type 来区分识别结果:
  1. 识别正常，矩形框、提示灯为绿色
  2. 活检未通过，矩形框、提示灯为黄
  3. 未注册，矩形框、提示灯为红色
  4. 找到人员，后台验证无反应，矩形框、提示灯为蓝色
  5. 找到人员，后台验证失败，矩形框、提示灯为蓝色
  6. 找到人员，未授权不能通过（通过时段判断），矩形框、提示灯为蓝色
- time 识别时间的格式 ： yyyy-MM-dd HH:mm:ss
- path 识别时的抓拍照片需要在设备的**“设置”**里，开启**“保存认证图片”**。如果未开启，则不会保存抓拍照片，返回是“”;
- 抓拍照片的文件名的格式：姓名_ yyyy.MM.dd.HH.mm.ss_识别分数.jpg

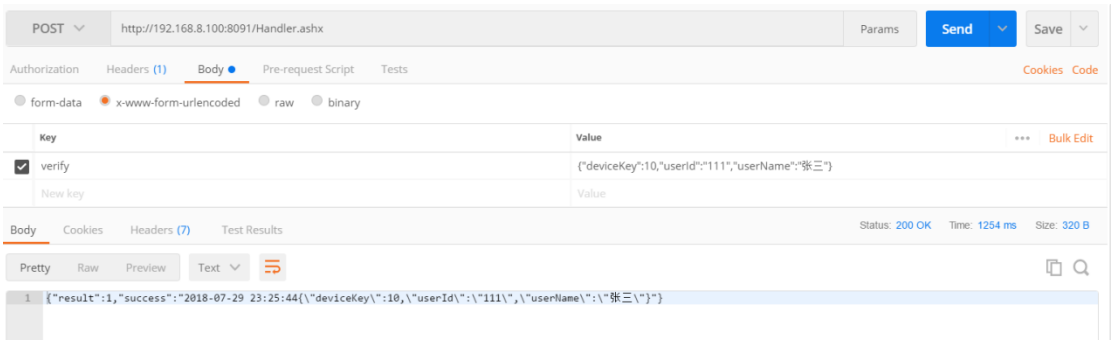
返回

参数名	描述	类型	说明
result	结果	String	
success	验证结果	Boolean	True:成功,False:失败
msgtype	返回类型	Int	接口返回的类型。提示的错误信息可能会变，但错误类型不会变，可用此字段来处理相应的功能逻辑
msg	返回信息	String	接口返回的信息

说明：

- 需要返回 result 字段，=1 代表已经接收到识别数据
- 如果未返回或 result 不等于 1 的时候，会在一定时间（现在是 10 分钟）后，由 2.3.4（历史识别记录回调）继续回调上传
-

PostMan 示例：



返回示例：

{"result":1,"success":true}

Demo示例：

[52:30],{"banch":"","base64":"/9j/...deviceKey":"5874v-v43t9-8wv43-t98zv-54785","direction":1,"guid":"79a1d7ea-8cc0-46d5-90b6-b4909902532c","id":13317,"ip":"192.168.8.203","machineCode":"5874v-v43t9-8wv43-t98zv-54785","path":"ftp://192.168.8.203:8010/FacePic/2019_03_26/verify/success/18/张洪鹏_2019_03_26_18_52_47_0.91203207.jpg","score":0.91203207,"sendPassType":0,"time":"2019-03-26 18:52:47","type":1,"userId":"001","userName":"张洪鹏"}

从上图可以看出来，通过回调方法接收到人脸识别数据

2.3.2 验证回调方法说明

URL：/???，在 2.1.2 里设置 type=4

method：POST

当人脸识别识别到人员后，调用后台验证回调方法来确定结果。可用于人脸消费，计次消费等。

设置 URL 后，如果需要开启，可以到 APP 里**“设置”**—**“平台验证”**选中，才能启用

参数名	描述	类型	必传	说明
deviceKey	机器码	String	Y	
guid	认别记录的 ID	String	Y	

userId	人员编码	String	Y	人员唯一标识
usrName	人员名字	String	Y	
cost	消费金额	Int		如果是人脸多少，则会传此次人脸识用户需要消费的金额，单位：分 暂时不提供
type	识别类型	Int		识别类型，参看 2.3.1 中的 type

返回

参数名	描述	类型	说明
result	结果	String	
success	验证结果	Boolean	True:成功,False:失败 成功才会有后续操作，比如开闸
msgtype	返回类型	Int	要显示的内容
msg	返回信息	String	接口返回的信息

说明：

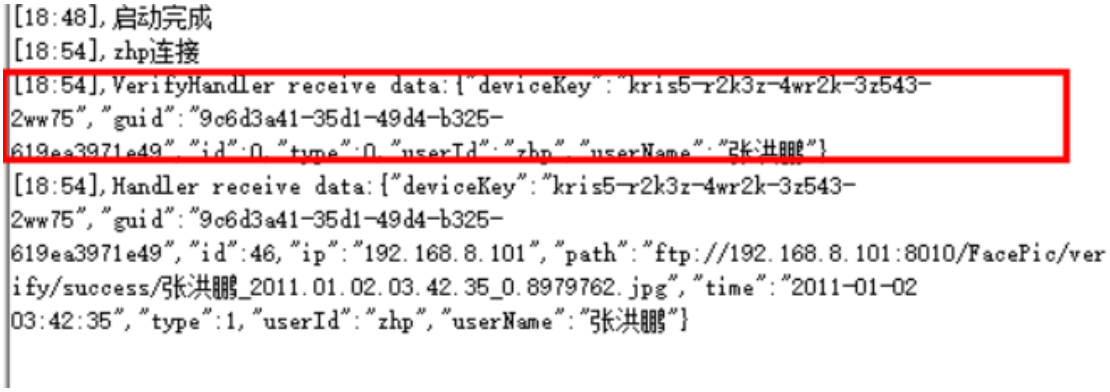
- 需要返回 result 字段，=1 代表已经接收到识别数据

PostMan 示例：

返回示例：

{"result":1,"success":true,"msgtype":0,"msg":"你好,张洪鹏\r\n总次数[100],剩余[97]"}

Demo示例：



从上图可以看出来，人员识别后需要到后台进行验证，并返回结果和相应的提示信息

2.3.3 心跳包方法说明

当人脸识别识别到人员后，调用后台验证回调方法来确定结果。可用于人脸消费，计次消费等

URL：/???，在 2.1.2 里设置 type=5

method：POST

参数名	描述	类型	必传	说明
Info	心跳包内容	JsonObject	Y	见下面：DevicesHeartBeat

心跳包对象：

```
/// <summary>
    /// 设备返回时的对象
    /// </summary>
    public class DevicesHeartBeat
    {
        /// <summary>
        /// 设备编号，可用户自定义，如果没有自定义，则返回机器码。
        /// </summary>
        public String deviceKey { get; set; }
        public int deviceType { get; set; }
        /// <summary>
        /// 设备的机器码
        /// </summary>
        public String deviceMachineCode { get; set; }
        /// <summary>
        /// 运行时间
        /// 格式：天d小时h分钟m
        /// </summary>
        public String runtime { get; set; }
        /// <summary>
        /// APP启动的时间
        /// </summary>
```

```
    public String starttime { get; set; }
    /// <summary>
    /// 当前系统时间
    /// </summary>
    public String time { get; set; }
    public String ip { get; set; }
    public int personCount { get; set; }
    public int faceCount { get; set; }
    public String version { get; set; }
    /// <summary>
    /// APP占用内存
    /// </summary>
    public float memory { get; set; }
    /// <summary>
    /// 系统可用内存
    /// </summary>
    public float availMem { get; set; }
    /// <summary>
    /// 系统总内存
    /// </summary>
    public float totalMem { get; set; }
    /// <summary>
    /// 硬盘大小
    /// </summary>
    public float totalDisk { get; set; }
    /// <summary>
    /// 设备的机器名
    /// </summary>
    public string buildModel { get; set; }
    /// <summary>
    /// 是否初始化
    /// 如果通信密码非空，则认为是初始化完成。
    /// 如果通信密码为空，则认为未初始化，不能操作设备
    /// </summary>
    public int isInited { get; set; }
    /// <summary>
    /// 是否授权
    /// </summary>
    public int isAuth { get; set; }
    /// <summary>
    /// 已使用的硬盘
    /// </summary>
    public float disk { get; set; }
}
```

对象类说明：

参数名	描述	类型	必传	说明
deviceKey	设备编号	String	Y	用户自定义，如果没有自定义，则返回机器码
deviceType	设备类型	Int	Y	咨询厂家
buildModel	设备名称	String		
deviceMachineCode	机器码	String	Y	
Runtime	App 运行时间	String	Y	格式：天 d 小时 h 分钟 m
starttime	App 启动时间	String		
time	机器时间	String	Y	yyyy-MM-dd HH:mm:ss
ip	机器 Ip	String	Y	x.x.x.x
personCount	人员数量	Int	Y	可以人脸 App 界面里:“设置”-“用户管理”看到对应的值
faceCount	人脸照片数量	Int	Y	可以人脸 App 界面里:“设置”-“用户管理”看到对应的值
version	软件版本号	String	Y	x.x.x.x
sendCount	发送次数	Int	Y	已经发送心跳包的次数 每次间隔 1 分钟
memory	APP 占用内存	float	Y	
availMem	系统可用内存	float	Y	
totalMem	系统总内存	float	Y	
disk	已用的硬盘大小	float	Y	
totalDisk	硬盘大小	float	Y	
isInited	是否初始化	float	Y	1:已初始化 如果通信密码非空，则认为是初始化完成。如果通信密码为空，则认为未初始化，不能操作设备
isAuth	是否授权	Int		1：已授权

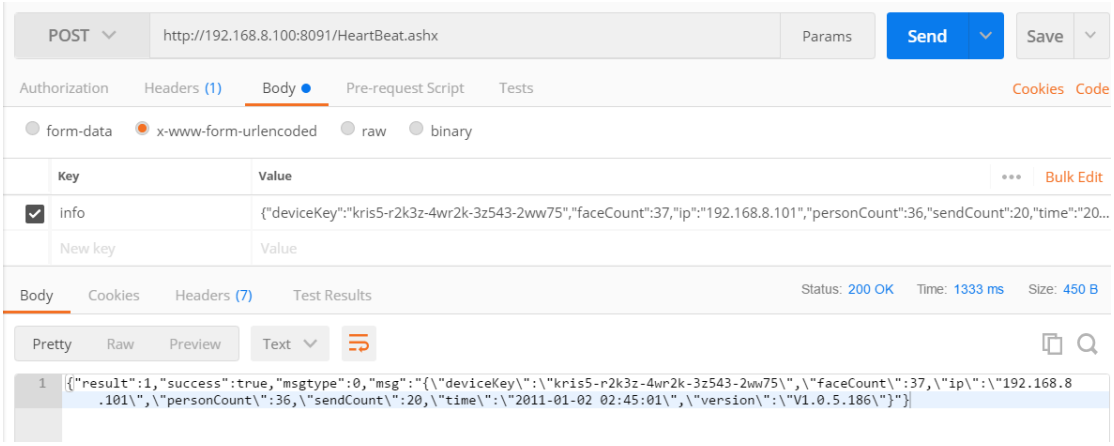
返回

参数名	描述	类型	说明
result	结果	String	
success	验证结果	Boolean	True:成功,False:失败
msgtype	返回类型	Int	
msg	返回信息	String	接口返回的信息

说明：

- 接收后，返回后。APP 主界面就有连接状态。如果 APP 未收到回应，则会提示“未连接”状态。

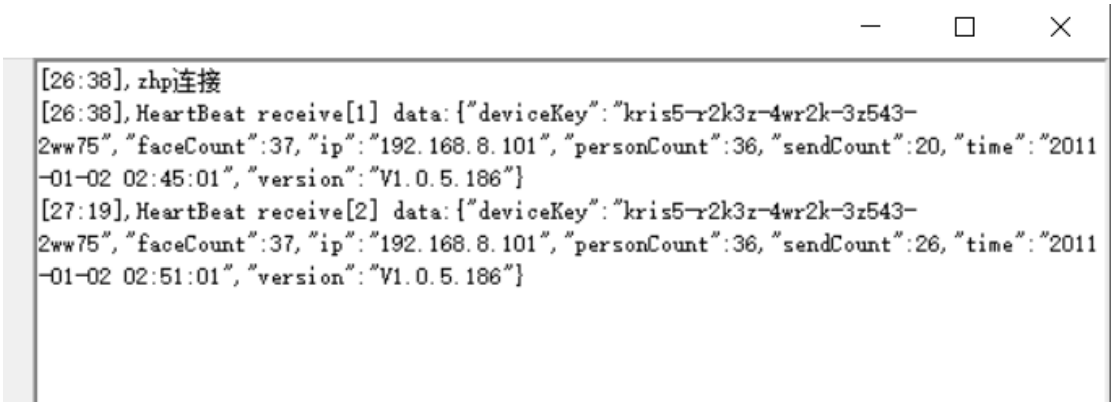
PostMan 示例：



返回示例：

{"result":1,"success":true}

Demo示例：



从上图可以看出来，通过心跳包可以查询到设备的一些基本数据

2.3.4 历史识别记录回调方法说明

当回调方法没的得到正确返回时，每隔 10 分钟会调用此方法，进行历史识别记录回调。

URL：/???，在 2.1.2 里设置 type=6

method：POST

参数名	描述	类型	必传	说明
verify	验证 Json 对象	JsonString	Y	

public class Verify {

```
private String deviceKey;//机器设备码
private int id;//记录 id,数据库中记录唯一标识
private String guid;//记录的 guid
private String userId;//用户 Id
private String userName;//用户姓名
```

```
    private String ip;
    private int type;//识别结果
    private String path;//识别人脸提供的 ftp 路径
    private String time;//识别时间, yyyy-MM-dd HH:mm:ss
}
```

说明：此方法和 2.3.1 主要的区别是时效性，在识别人脸的当时如果没有调用 2.3.1 成功，那么在以后的时间里，会定时用此方法进行回调上传记录。

- 无

返回

参数名	描述	类型	说明
result	结果	String	
success	验证结果	Boolean	True:成功,False:失败
msgtype	返回类型	Int	接口返回的类型。提示的错误信息可能会变，但错误类型不会变，可用此字段来处理相应的功能逻辑
msg	返回信息	String	接口返回的信息

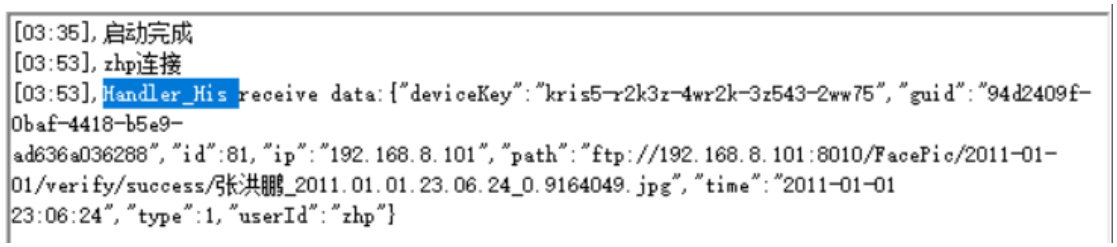
说明：

- 需要返回 result 字段，=1 代表已经接收到识别数据
- 如果未返回或 result 不等于 1 的时候，会在一定时间（现在是 10 分钟）后，继续回调上传

返回示例：

```
{"result":1,"success":true}
```

Demo示例：



从上图可以看出来，通过回调方法接收到人脸识别的历史数据