xFace说明文档

UfaceOffline 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。

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。 |

目录

[1.概述 4](#_Toc522523918)

[1.1 简述 4](#_Toc522523919)

[1.2 接口规范 4](#_Toc522523920)

[1.3接口返回 4](#_Toc522523921)

[1.4 设备识别人员接口调用流程 5](#_Toc522523922)

[2.接口说明 6](#_Toc522523923)

[2.1设备管理接口 6](#_Toc522523924)

[2.1.1设备API接口调用密码 6](#_Toc522523925)

[2.1.2设置管理URL 7](#_Toc522523926)

[2.1.3得到设备机器码 9](#_Toc522523927)

[2.1.4设备授权 10](#_Toc522523928)

[2.1.4设备心跳包 11](#_Toc522523929)

[2.2人员照片同步接口 12](#_Toc522523930)

[2.2.1用户查询接口 12](#_Toc522523931)

[2.2.2用户增加、更新 15](#_Toc522523932)

[2.2.3用户时段权限-新增或删除 17](#_Toc522523933)

[2.2.4用户时段权限-删除 21](#_Toc522523934)

[2.2.5人员设置时间 21](#_Toc522523935)

[2.2.6数据刷新 23](#_Toc522523936)

[2.3人员照片同步接口 24](#_Toc522523937)

[2.3.1回调方法说明 24](#_Toc522523938)

[2.3.2验证回调方法说明 25](#_Toc522523939)

[2.3.3心跳包方法说明 27](#_Toc522523940)

# 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 和 type：face/card\_0/1(现在只默认传0)；

**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API接口调用密码

**URL：/setPassWord**

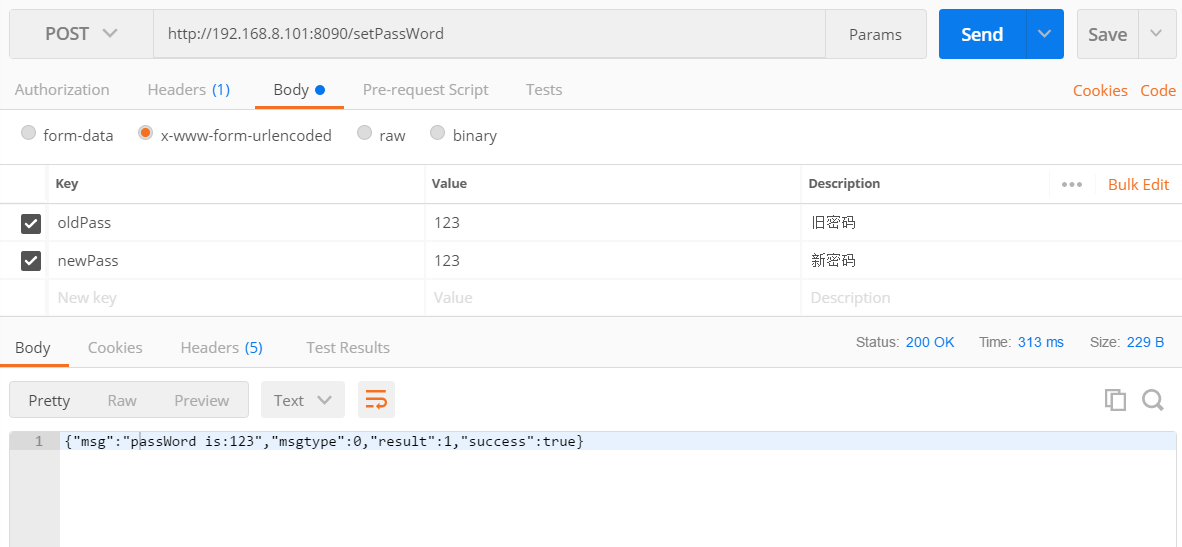
**method：POST**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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：**

* 新设备或重置后（恢复初始化）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

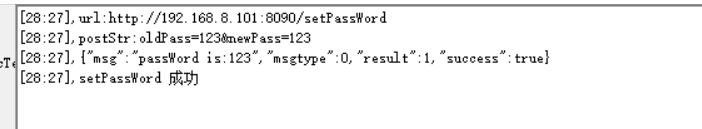
**PostMan示例：**



**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**

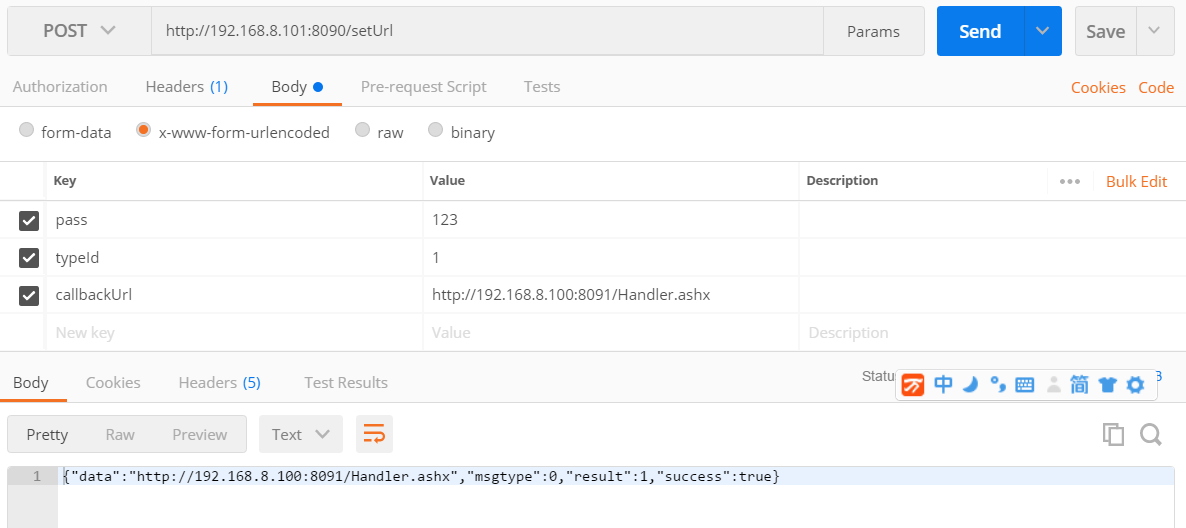
**method：POST**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |
| 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
* 设备成功识别人员后，会向识别回调地址 POST 字段 ip 、 personId 、 time （ 识 别 时 间 ）、deviceKey、type：face/card\_0/1/2 和 path
* 外部接口的 URL 需要符合正则表达式：String check = "((http|ftp|https)://)(([a-zA-Z0-9\\.\_-]+\\.[a-zA-Z]{2,6})|([0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}))(:[0-9]{1,4})\*(/[a-zA-Z0-9\\&%\_\\./-~-]\*)?";
* 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
* 如果类型=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

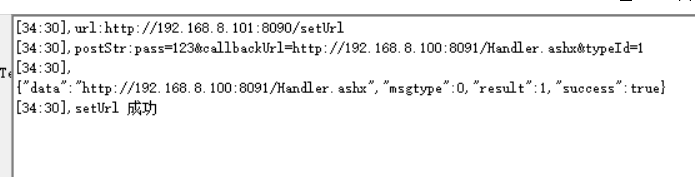
**PostMan示例：**



**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**

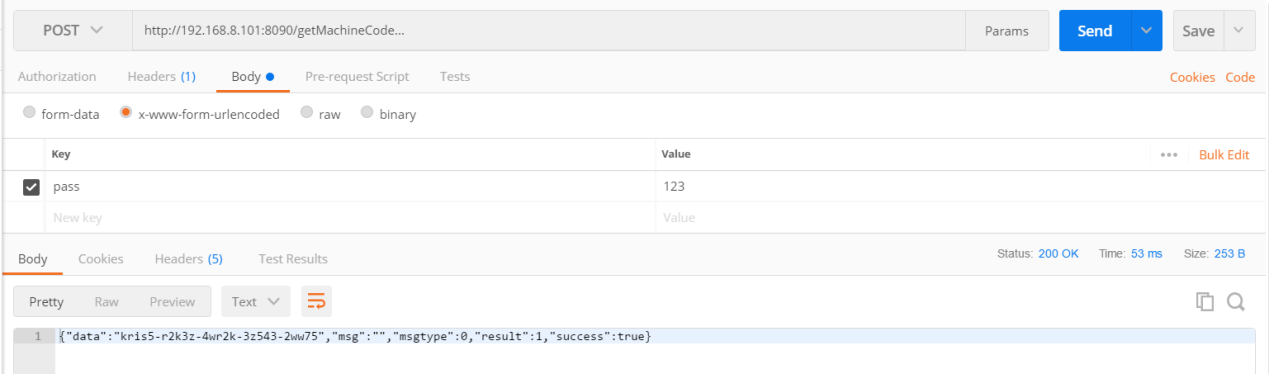
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 描述 | 类型 | 必传 | 说明 |
| pass | 访问密码 | String | Y |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 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。

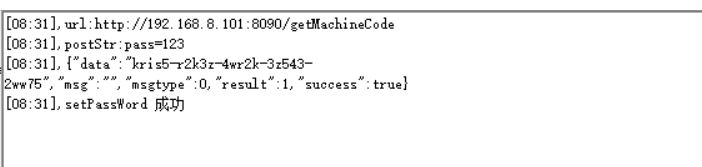
**PostMan示例：**



**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**

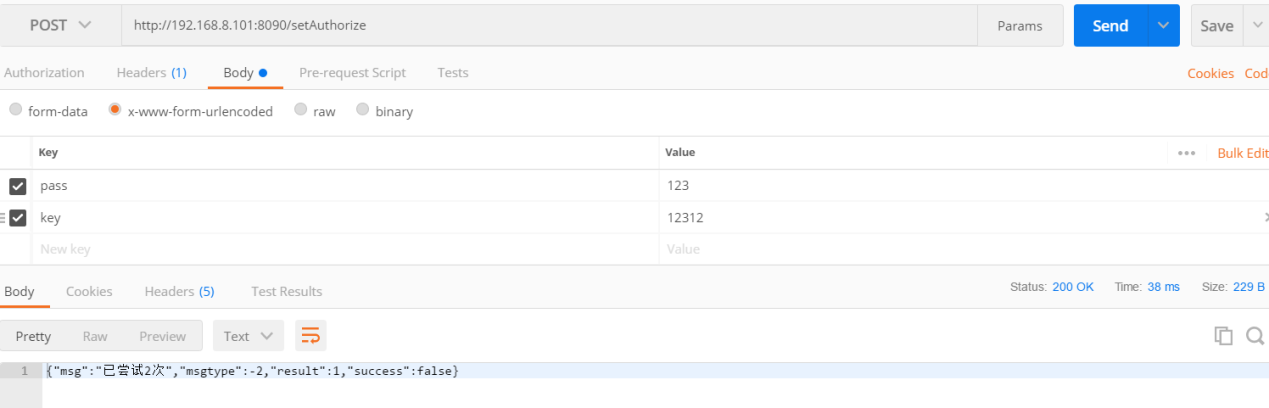
**method：POST**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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

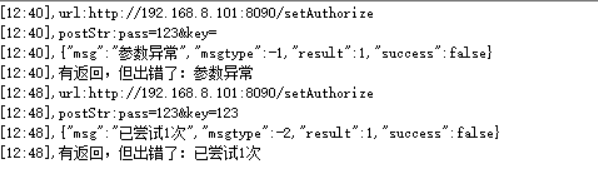
**PostMan示例：**



**返回示例：**

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

**Demo示例：**



**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 |

### 2.1.4设备心跳包

**URL：/setDeviceHeartBeat**

**method：POST**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 描述 | 类型 | 必传 | 说明 |
| pass | 访问密码 | String | Y |  |
| url | 外部接收设备心跳监  听的服务器接口地址 | String | Y |  |

**url说明：**

* 给设备设置一个外部回调地址
* 设备每隔一分钟会向该接口 POST 字段deviceKey、time、ip、personCount、faceCount和 version
* 外部接口的 URL 需要符合正则表达式：String check = "((http|ftp|https)://)(([a-zA-Z0-9\\.\_-]+\\.[a-zA-Z]{2,6})|([0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}))(:[0-9]{1,4})\*(/[a-zA-Z0-9\\&%\_\\./-~-]\*)?";
* 传入内容为空可以清空回调地址，清空后将不再进行回调

**返回示例：**

{"data":"http://www.baidu.com","msg":"","msgtype":0,"result":1,"success":true}

**Post字段内容包含：**

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| deviceKey | String | 备唯一标识码 |
| time | String | 设备当前时间戳 |
| ip | String | 设备当前 IP 地址 |
| personCount | String | 设备当前注册人员数量 |
| faceCount | String | 设备当前注册的照片数量 |
| version | String | 设备版本号 |

**返回结果说明：**

|  |  |  |
| --- | --- | --- |
| **msgtype** | **msg** | **结果或错误原因** |
| 0 | 无 | 成功 |
|  |  |  |
|  |  |  |

## 2.2人员照片同步接口

### 2.2.1用户查询接口

**URL：/user/findDifference**

**method：POST**

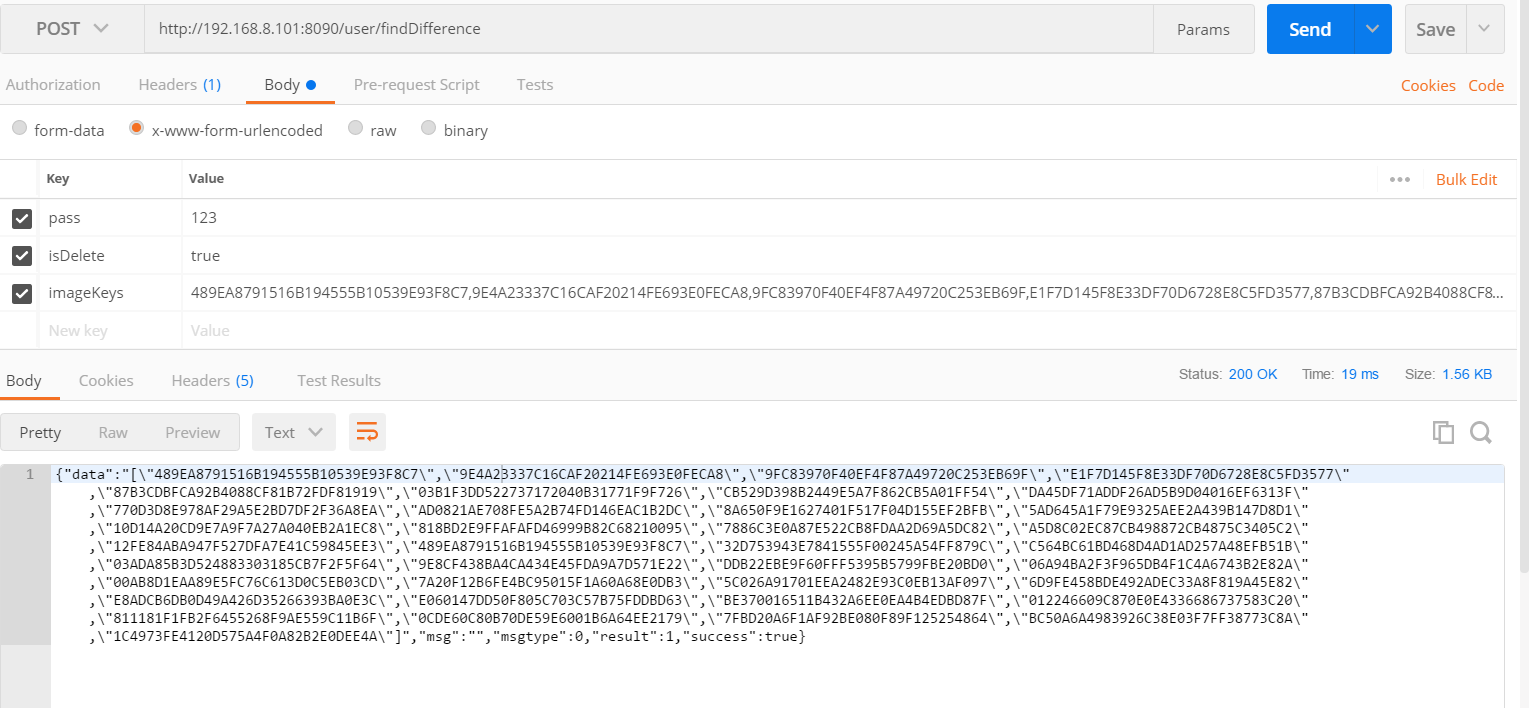
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 描述 | 类型 | 必传 | 说明 |
| pass | 访问密码 | String | Y |  |
| isDelete | 直接删除设备上没有的人脸 | String | Y | true:删除  false:不删除 |
| imageKeys | 人脸照片的特征值 | String  用“,”分隔 | Y |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 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。**以住同步经常会出现，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：**

****

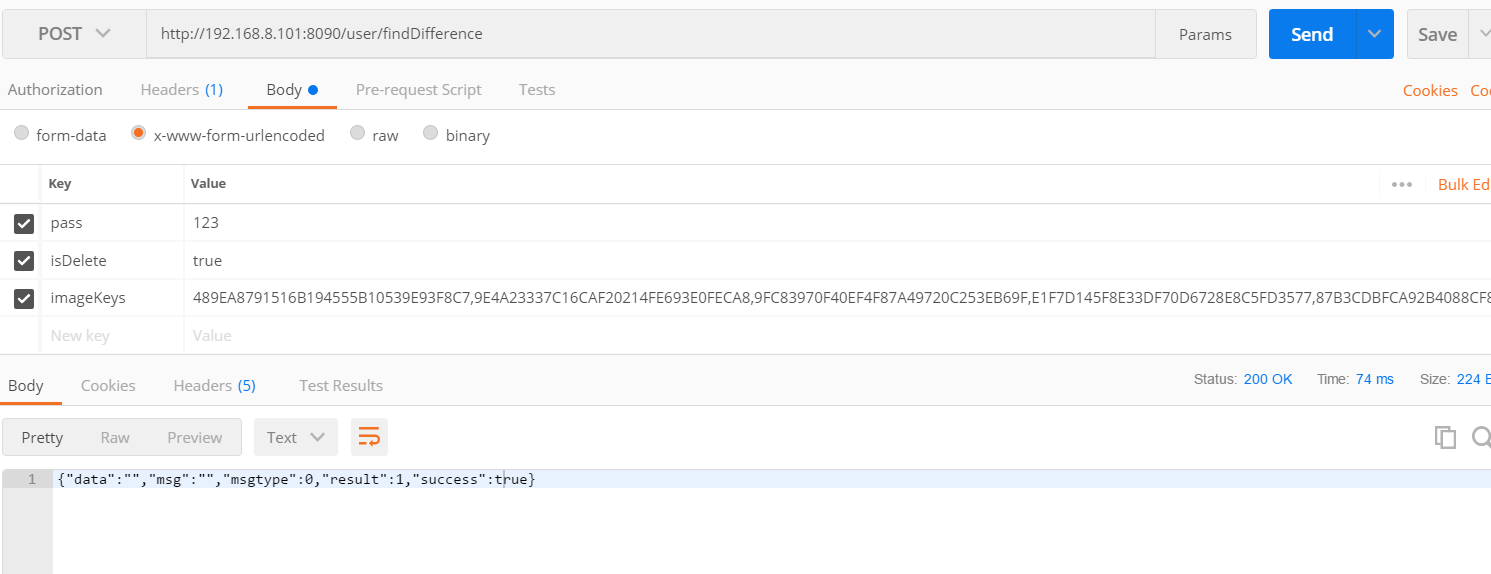
Date有返回，说明需要增加的照片

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



**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。



**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; }

}

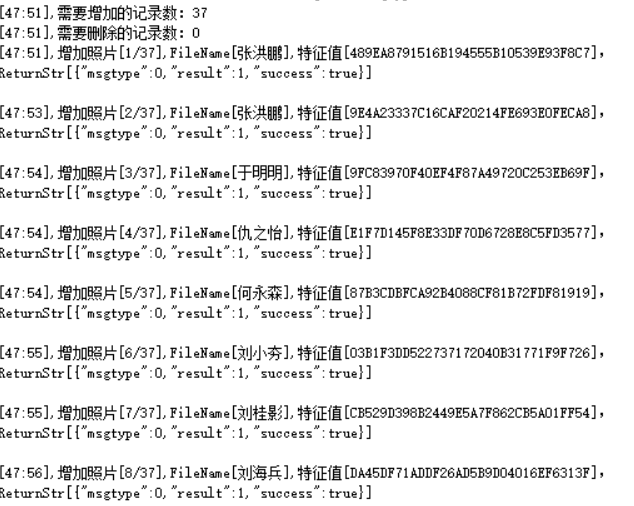
**说明：**

* 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：**



**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);

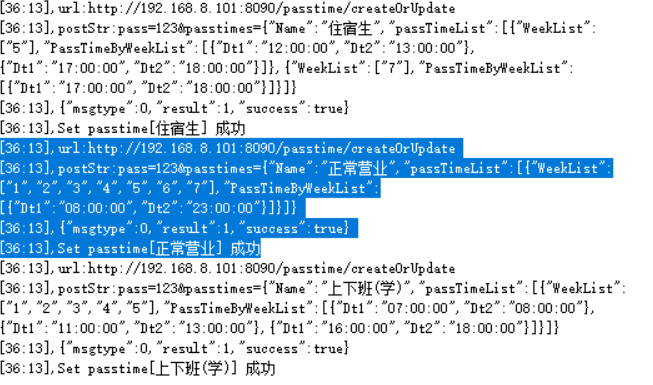
return res;

}

**返回示例：**

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

**Demo示例：**



**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：**

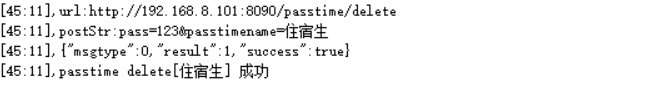
* **可以在“设置”**🡪**“时段管理” 里查看时段信息；**
* **如果有人员此时段，然后调用此接口删除时段，则人员对应的时段会变成“未知时段”将可以7\*24通过。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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：**



**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; }

}

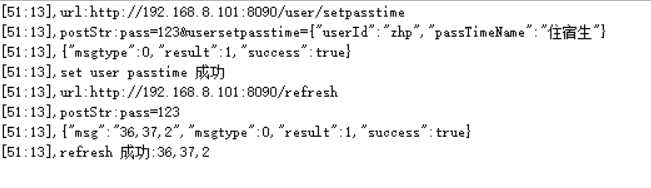
**说明：**

* 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：**



**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 |  |
|  |  |  |  |  |
|  |  |  |  |  |

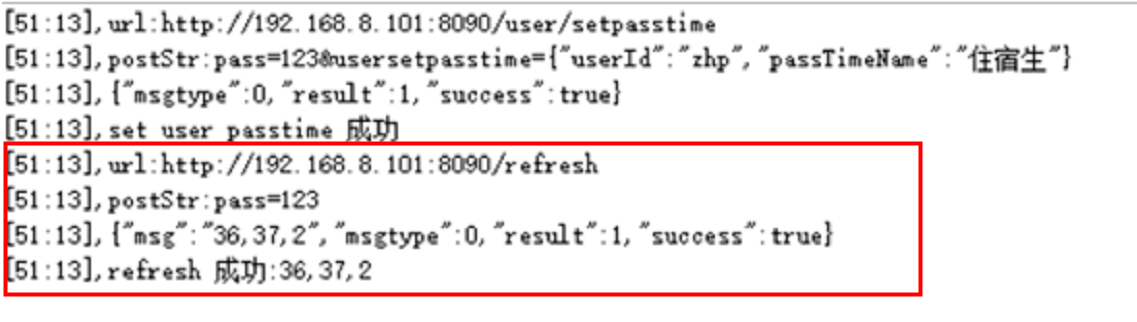
Msg**说明：**

* 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：**



**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：

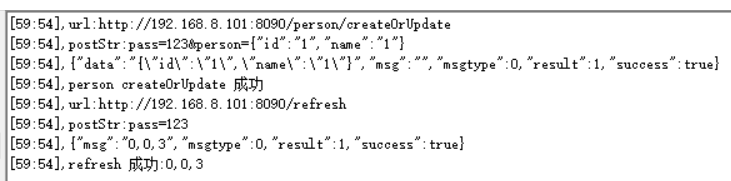
{"data":"{\"id\":\"1\",\"name\":\"1\"}","msg":"","msgtype":0,"result":1,"success":true}

传入id:””,name:”1”的返回结果：

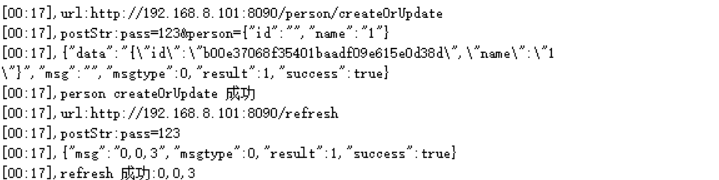
{"data":"{\"id\":\"b00e37068f35401baadf09e615e0d38d\",\"name\":\"1\"}","msg":"","msgtype":0,"result":1,"success":true}

**Demo示例：**

传入id:”1”,name:”1”的返回结果：



传入id:””,name:”1”的返回结果：



**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：**



**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 |  |
|  |  |  |  |  |

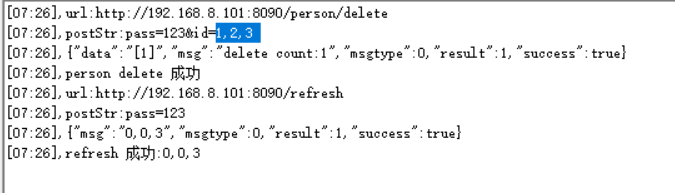
**说明：**

* 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：**



**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; }

}

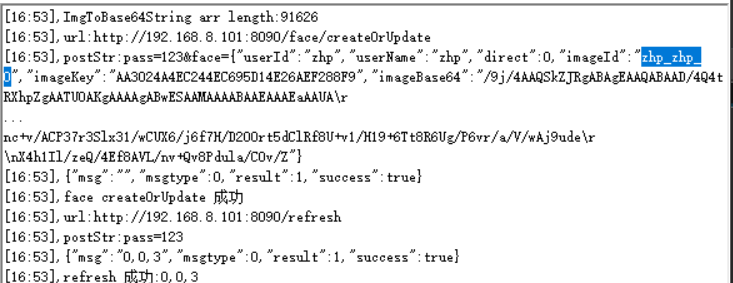
**说明：**

* 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：**



**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 |  |
|  |  |  |  |  |

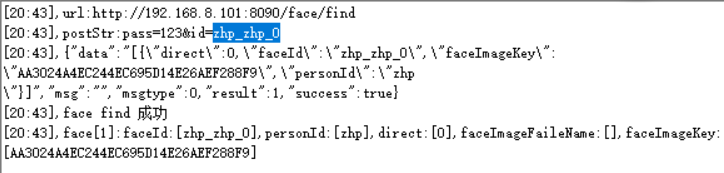
**说明：**

* 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：**



**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 |  |
|  |  |  |  |  |

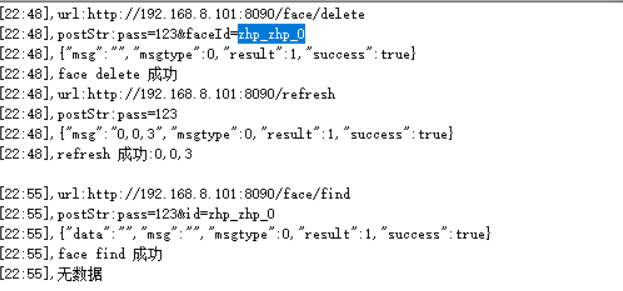
**说明：**

* 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：**



**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 |
|  |  |  |

## 2.3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 |  |

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：**

* 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

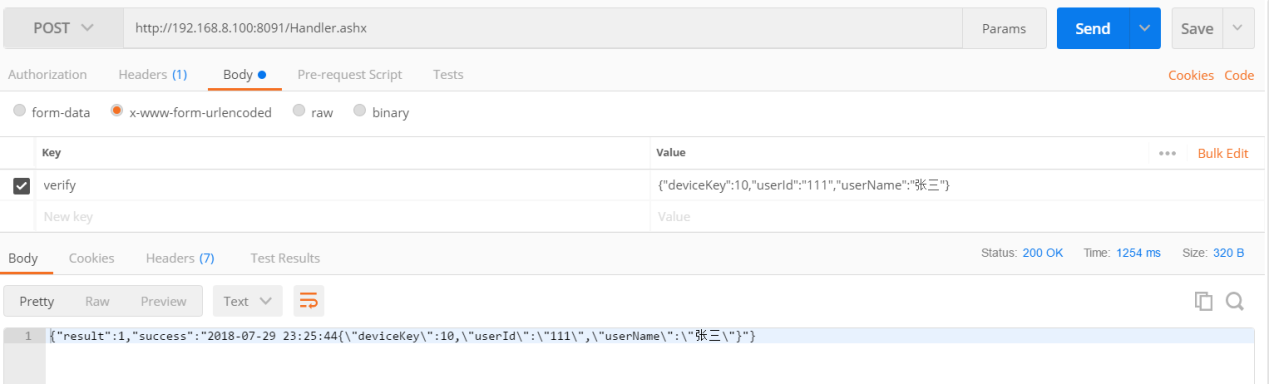
**返回**

|  |  |  |  |
| --- | --- | --- | --- |
| 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

**PostMan示例：**



**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

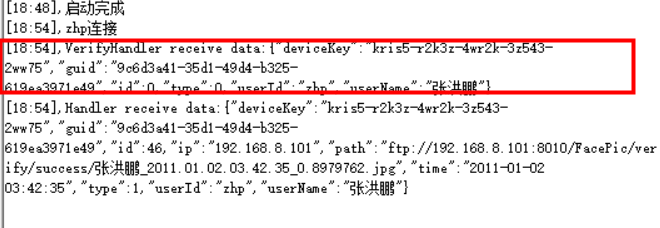
**PostMan示例：**



**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：

public class DevicesHeartBeat

{

public String deviceKey { get; set; }

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; }

}

对象类说明：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名 | 描述 | 类型 | 必传 | 说明 |
| deviceKey | 机器码 | String | Y |  |
| 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 |
|  |  |  |  |  |

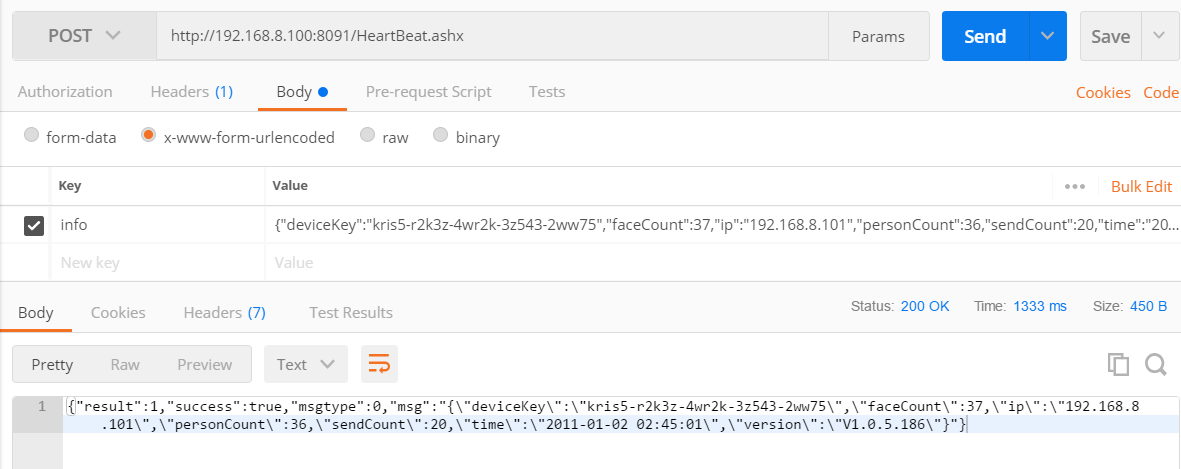
返回

|  |  |  |  |
| --- | --- | --- | --- |
| 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，可以返回，也可以不返回，暂时没有对回应做相应的功能

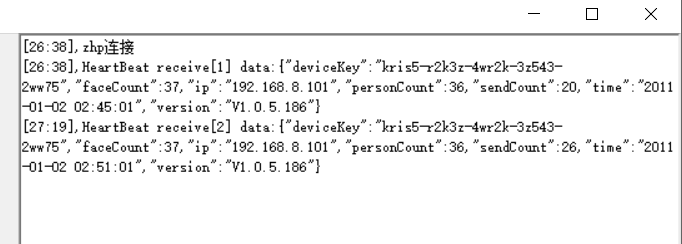
**PostMan示例：**



**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