ALIBABA CLOUD

阿里云

物联网应用服务
开放能力

文档版本：20220606

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.智能通行服务

1.1. 使用必读

更新记录

2021-11-16

根据最新的网络安全要求，人脸识别通行记录里的人脸抓拍图片是否上报到云端，可以进行配置化管理，详见人脸抓拍图存储控制。

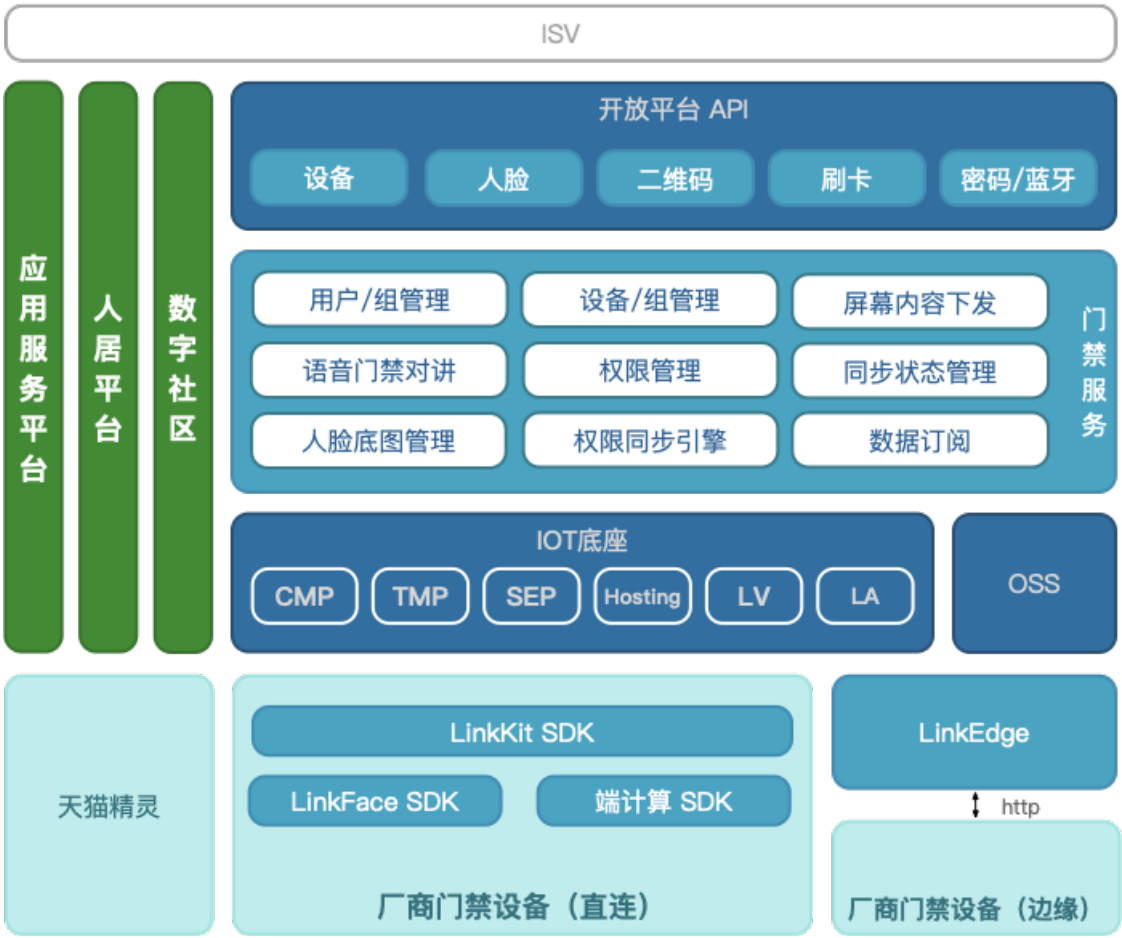
1.2. 功能介绍

1.2.1. 整体介绍

门禁服务是IOT领域服务中的一种，主要面向各类门禁设备提供云、边、端的完整业务闭环能力。门禁服务向厂商提供南向的设备上云和SDK接入，向ISV提供北向的业务服务。

1. 服务架构

门禁服务基于IOT底座，支持直连门禁设备（SDK）和边缘对接（DRIVER）设备的接入能力。服务于数字社区平台、用户服务平台、地产平台、园区平台，向ISV提供服务能力和数据能力。



2. 门禁设备

支持的门禁类型有：

门禁类型	说明
人脸门禁	可选人脸特征下发[1]、健康码校验[2]、
二维码门禁	支持离线校验，可选加密二维码能力
卡/蓝牙门禁	
密码门禁	

注：

[1]人脸特征的支持需要厂商对接算法上云；

[2]健康码校验需要用户联系本地大数据局提供接口服务，具体请联系商务提需求。

对接时参考《设备接入》的《业务流程》进行对接。

3. 服务能力（北向）

北向能力API包括下面几类：

服务能力	说明
用户、设备关系服务	提供用户、用户组、设备组、设备的关系管理能力。 不再需要从用户到设备点对点下发权限，ISV只需要维护关系即可。用户添加的权限，云端会根据用户与设备的关系自动部署到设备上。
人脸权限服务	人脸图上传、修改服务； 同步维护服务
二维码权限服务	二维码获取、配置
卡/蓝牙权限服务	卡/蓝牙获取、配置
密码权限服务	密码获取、配置
功能服务	远程开门、常开门配置

4. 数据能力（北向）

设备事件上报以及被动数据推送有：

数据能力	说明
门禁通行	支持二维码、卡/蓝牙、密码
刷脸通行	支持人脸1.0和2.0
陌生人脸通行	支持人脸1.0 OnDetect事件
业务数据推送	人脸部署状态变更、设备全量权限、设备同步错误
设备报警事件	设备上下线，以及其它设备报警事件推送
健康码验证记录	健康码门禁的校验结果推送

安全声明

> **说明**
>
> 本服务收集的人脸信息仅用于通行场景下的人脸识别。人脸数据的存储和传输都在加密的安全环境中进行，确保人脸信息不会泄露。

1.2.2. 人脸权限服务

概述

人脸门禁服务是一套为企业提供的支持人脸图片管理与人脸权限管理的服务。**人脸门禁服务的云端API同时兼容边缘方案与端侧方案。**

人脸图片管理

接口名称	接口路径	当前版本
保存人脸图片	/face/paas/image/save	1.1.1
删除人脸图片	/face/paas/image/delete	1.1.1
获取人脸图片	/face/paas/image/get	1.0.2

人脸权限管理

接口名称	接口路径	当前版本
增加人脸权限	/face/paas/permission/add	1.0.0
删除人脸权限	/face/paas/permission/delete	1.0.0
查询用户的权限状态	/face/paas/permission/querybyuser	1.0.4
查询设备的权限状态	/face/paas/permission/querybydevice	1.0.4

测温配置

接口名称	接口路径	当前版本
设置测温配置	/entrance/paas/face/temperature/config/set	1.0.0
获取测温配置	/entrance/paas/face/temperature/config/get	1.0.0

保存人脸图片

保存单个用户的人脸图片，支持不同用户账号体系，图片以URL或base64编码的格式传入，宽、高不超过800像素，文件大小不超过1MB。

path	版本
/face/paas/image/save	1.1.1

参数	类型	是否必填	备注
userType	String	是	用户类型，与userId共同作为人脸图片的唯一标识，目前支持的取值有 IDENTITY：用identityId作为userId OPEN：OA三方账号体系
userId	String	是	用户ID

参数	类型	是否必填	备注
imageUrl	String	否	图片URL（图片URL和图片base64数据，两者必填其一，优先使用imageUrl）
imageBase64	String	否	图片base64数据（图片URL和图片base64数据，两者必填其一，优先使用imageUrl）
faceExtInfo	String	否	设备端业务扩展字段，用于同步到设备端实现业务扩展逻辑，建议使用JSON对象格式，利于标准化扩展；不超过1024字符
userName	String	否	（标准化扩展字段）用户姓名，当faceExtInfo为JSON对象格式时，以userName为key填充到faceExtInfo中；不超过64字符
expiredTime	String	否	（标准化扩展字段）过期时间，当faceExtInfo为JSON对象格式时，以expire为key填充到faceExtInfo中，时间格式yyyy-MM-dd HH:mm:ss
userExtInfo	String	否	云端业务扩展字段，用于设备上报数据时推送给ISV，实现云端业务扩展逻辑；不超过1024字符
policy	String	否	人脸匹配后人员通行策略。PERMISSION（默认） - 允许； DENY - 禁止通行

返回结果使用通用结果类型，不使用data域。

入参示例

```
{
  "userType":"OPEN",
  "userId":"xxx",
  "imageUrl":"http://yyy/1.jpg"
}
```

出参示例

```
{
    "code": 200,
    "message": "success"
}
```

常见错误码

code	message	说明
2006	图片文件大小或分辨率不符合要求	宽、高不超过800像素，文件大小不超过1MB
2007	获取图片数据失败	URL下载失败或者BASE64数据解析失败
2012	用户信息未知	identityId或openId不正确
2017	图片内容有误，请确认图片中包含人脸	图片中未检测到人脸
2018	请确认图片中仅包含一个人	图片中检测到多张人脸
2019	照片中人脸区域太小，请对照片进行适度缩放与裁剪	人脸区域在整张图片中占比需要超过总面积的10%
2027	请确保使用正面人脸图片	图片中的人脸偏角过大，可能有以下几种情况：（1）抬头或低头的角度过大；（2）侧脸角度过大；（3）左右歪头的角度过大
2460	request parameter error.	缺少必要参数

删除人脸图片

删除单个用户的人脸图片，支持不同用户账号体系。

path	版本
/face/paas/image/delete	1.1.1

参数	类型	是否必填	备注
userType	String	是	用户类型，与userId共同作为人脸图片的唯一标识
userId	String	是	用户ID

返回结果使用通用结果类型，不使用data域。

入参示例

```
{
  "userType":"OPEN",
  "userId":"xxx"
}
```

出参示例

```
{
    "code": 200,
    "message": "success"
}
```

获取人脸图片

根据用户id和用户类型查询人脸图片，返回的图片格式支持URL和base64编码。

path	版本
/face/paas/image/get	1.0.2

参数	类型	是否必填	备注
userType	String	是	用户类型，与userId共同作为人脸图片的唯一标识
userId	String	是	用户ID
imageFormat	String	否	图片数据的返回格式，目前支持URL和BASE64，默认为URL

返回结果使用通用结果类型，data域是对象，见下表的详细说明：

字段	类型	备注
imageUrl	String	人脸图片URL，当入参imageFormat为空或URL时有值
imageBase64	String	人脸图片base64数据，当入参imageFormat为BASE64时有值

入参示例

```
{
  "userType":"OPEN",
  "userId":"xxx",
  "imageFormat":"URL"
}
```

出参示例

```
{
    "code": 200,
    "data": {
    "imageUrl":"http://yyy/1.jpg"
    },
    "message": "success"
}
```

增加人脸权限

将已保存的用户人脸图片下发到设备端，使设备有权限识别对应的人脸用户。

path	版本
/face/paas/permission/add	1.0.0

参数	类型	是否必填	备注
userType	String	是	用户类型，与userId共同作为人脸图片的唯一标识
userIdList	JSONArray	是	用户ID列表
scopeType	String	是	人脸信息下发的目标范围类型： IOT_ID：将人脸信息下发到指定的设备
scopeIdList	JSONArray	是	人脸信息下发的目标范围列表

返回结果使用通用结果类型，不使用data域。

入参示例

```
{
  "userType":"OPEN",
  "userIdList":
  [
    "xxx"
  ],
  "scopeType":"IOT_ID",
  "scopeIdList":
  [
    "zzz"
  ]
}
```

出参示例

```
{
    "code": 200,
    "message": "success"
}
```

删除人脸权限

将已保存的用户人脸图片从设备端删除，使设备无权限识别对应的人脸用户。

path	版本
/face/paas/permission/delete	1.0.0

参数	类型	是否必填	备注
userType	String	是	用户类型，与userId共同作为人脸图片的唯一标识
userIdList	JSONArray	是	用户ID列表
scopeType	String	是	人脸信息下发的目标范围类型： IOT_ID：将人脸信息下发到指定的设备
scopeIdList	JSONArray	是	人脸信息下发的目标范围列表

返回结果使用通用结果类型，不使用data域。

入参示例

```
{
  "userType":"OPEN",
  "userIdList":
  [
    "xxx"
  ],
  "scopeType":"IOT_ID",
  "scopeIdList":
  [
    "zzz"
  ]
}
```

出参示例

```
{
    "code": 200,
    "message": "success"
}
```

查询用户的权限状态

根据用户id和用户类型查询人脸图片信息及其下发的设备列表（含下发状态）。

path	版本
/face/paas/permission/querybyuser	1.0.4

参数	类型	是否必填	备注
userType	String	是	用户类型，与userId共同作为人脸图片的唯一标识
userId	String	是	用户ID
deviceListPageNo	Integer	否	分页查询的请求页码
deviceListPageSize	Integer	否	分页查询的请求页大小
statusList	List	否	目标状态列表

返回结果使用通用结果类型，data域是对象，见下表的详细说明：

参数	类型	备注
userType	String	用户类型，与userId共同作为人脸图片的唯一标识
userId	String	用户ID
userName	String	用户姓名，不超过64字符
expiredTime	String	人脸图片有效期，时间格式yyyy-MM-dd HH:mm:ss
extInfo	String	业务扩展字段，不超过1024字符
deviceListTotal	Integer	该用户人脸图片执行过下发操作的设备总数
deviceListPageNo	Integer	请求页码
deviceListPageSize	Integer	请求页大小
deviceList	JSONArray	设备列表，包含设备iotId、下发时间、下发状态

入参示例

```
{
  "userType": "OPEN",
  "userId": "xxx",
  "statusList": [
    "transferred",
    "transferDeleted"
  ]
}
```

出参示例

```
{
    "code": 200,
    "data": {
    "userType":"OPEN",
    "userId":"xxx",
    deviceListTotal:1,
    deviceListPageNo:1,
    deviceListPageSize:20,
    deviceList:
    [
      "iotId":"zzz",
      "syncTime":"2019-02-28 19:00:00",
      "syncStatus":"transferred"
    ]
    },
    "message": "success"
}
```

查询设备的下发状态

查询下发到某个设备的人脸图片列表（含下发状态）。

path	版本
/face/paas/permission/querybydevice	1.0.4

参数	类型	是否必填	备注
iotId	String	是	设备ID
pageNo	Integer	否	分页查询的请求页码
pageSize	Integer	否	分页查询的请求页大小
statusList	List	否	目标状态列表

返回结果使用通用结果类型，data域是对象，见下表的详细说明：

参数	类型	备注
total	Integer	下发到该设备的用户人脸图片总数

参数	类型	备注
pageNo	Integer	请求页码
pageSize	Integer	请求页大小
userList	JSONArray	用户人脸列表，包含userId、userType、userName、expiredTime、extInfo、下发时间、下发状态

入参示例

```
{
  "iotId":"zzz"
}
```

出参示例

```
{
  "code": 200,
  "data": {
    "total": 1,
    "pageNo": 1,
    "pageSize": 20,
    "userList": [
      {
        "userType": "OPEN",
        "userId": "xxx",
        "syncTime": "2019-02-28 19:00:00",
        "syncStatus": "transferred"
      }
    ]
  },
  "message": "success"
}
```

权限状态说明

状态	说明
toBeTransferred	待下发，根据is_delete区分新增下发和删除下发
transferring	下发中，根据is_delete区分新增下发中和删除下发中

deviceOffline	设备离线
transferred	下发成功
faceCheckError	下发失败，提取特征值失败或其他未知原因
faceCheckTimeout	下发失败，提取特征值超时，当前仅用于边缘
faceDlError	下发失败，下载人脸图片失败，当前仅用于边缘
facePushError	下发失败，推送到终端设备出错，当前仅用于边缘
unknownError	下发失败，设备端未返回结果，原因未知
transferDeleted	删除下发成功
deleteFailed	删除下发失败
transferTimeout	下发超时，云端会自动重试
deleteTimeout	删除下发超时，云端会自动重试
needManual	经过多次重试后仍然下发失败，需要人工处理

设置测温配置

设置设备的测温配置，包括是否开启测温以及正常的测温范围。该接口需要与门禁边缘驱动（2.8.0版本及以上）配合使用。

path	版本
/entrance/paas/face/temperature/config/set	1.0.0

参数	类型	是否必填	备注

参数	类型	是否必填	备注
iotId	String	否	设备ID，若传入iotId，则忽略productKey和deviceName，否则productKey和deviceName必填
productKey	String	否	产品标识，若不填iotId，则该字段必填
deviceName	String	否	设备名称，若不填iotId，则该字段必填
checkTemperature	Boolean	是	是否开启测温功能
maxThreshold	Float	否	正常温度范围上限，单位：摄氏度，取值范围0~100，开启测温功能时必填
minThreshold	Float	否	正常温度范围下限，单位：摄氏度，取值范围0~100，开启测温功能时必填

返回结果使用通用结果类型，不使用data域。

入参示例

```
{
  "iotId": "1WJ0io2kvh3e5wtSCTuV000000",
  "checkTemperature": true,
  "maxThreshold": 37.5,
  "minThreshold": 0
}
```

出参示例

```
{
    "code": 200,
    "message": "success"
}
```

获取测温配置

获取设备的测温配置，包括是否开启测温以及正常的测温范围。该接口需要与门禁边缘驱动（2.8.0版本及以上）配合使用。

path	版本
/entrance/paas/face/temperature/config/get	1.0.0

参数	类型	是否必填	备注
iotId	String	否	设备ID，若传入iotId，则忽略productKey和deviceName，否则productKey和deviceName必填
productKey	String	否	产品标识，若不填iotId，则该字段必填
deviceName	String	否	设备名称，若不填iotId，则该字段必填

返回结果使用通用结果类型，data域是对象，见下表的详细说明：

参数	类型	备注
checkTemperature	Boolean	是否开启测温功能
maxThreshold	Float	正常温度范围上限，单位：摄氏度，取值范围0~100
minThreshold	Float	正常温度范围下限，单位：摄氏度，取值范围0~100

入参示例

```
{
  "iotId": "1WJ0io2kvh3e5wtSCTuV000000"
}
```

出参示例

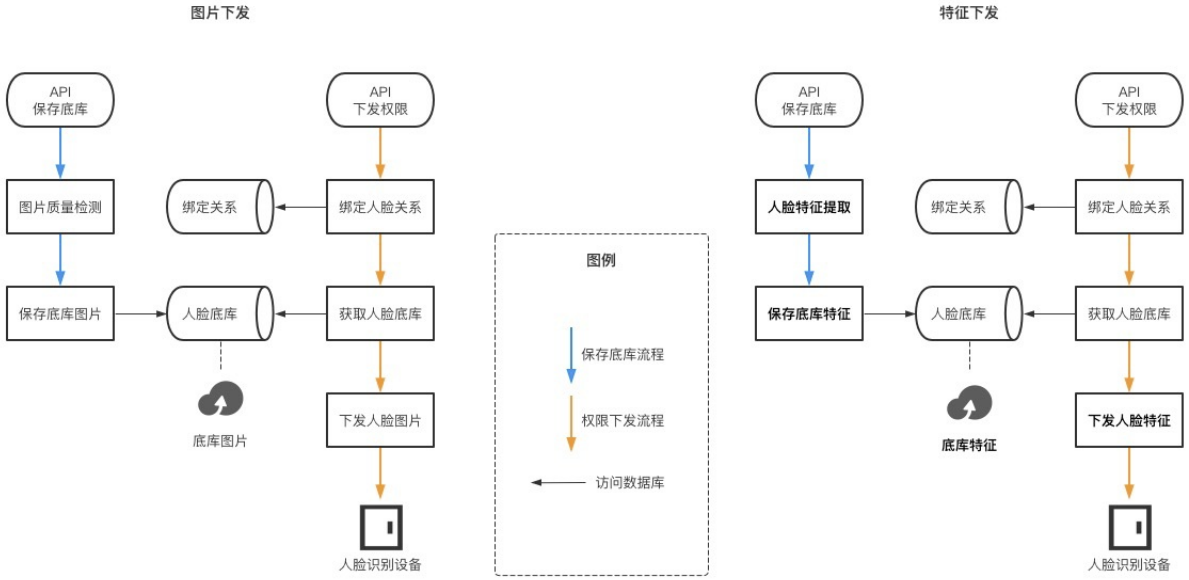
```
{
  "code": 200,
  "message": "success",
  "data": {
    "checkTemperature": true,
    "maxThreshold": 37.5,
    "minThreshold": 0
  }
}
```

1.2.3. 人脸特征管理

概述

人脸权限服务介绍了完整的从人脸底库图片录入，到下发给设备的使用流程。本文档在此基础上，介绍在底库录入时完成特征提取的方式，一方面可以提高下发效率，另一方面可以避免人脸底库图片保存在IoT平台，造成隐私安全性方面的隐患。

两种方案流程的对比：



其中特征下发主要涉及3方面：

1. 项目算法配置
2. 底库特征管理
3. 底库下发管理（API参考增加人脸权限）

兼容性说明

1. 本文档描述的特征管理方案需要基于物联网应用服务平台使用。
2. **单个项目使用的设备必须用同一个算法版本，或者至少保证不同算法版本之间特征值格式是兼容的。**
3. 项目中如果已经保存了底库图片，则不允许配置算法版本。
4. 项目中如果已经保存了底库特征，则不允许变更算法版本。
5. 底库录入：项目如已配置算法版本，则不允许保存底库图片，只允许保存底库特征；项目如未配置算法版本，则不允许保存底库特征，只允许保存底库图片。
6. 权限下发：项目如已配置算法版本，则只取底库特征；项目如未配置算法版本，则只取底库图片。

API：查询可用的算法

查询平台当前可用的人脸特征提取算法列表。用于配置项目使用的算法版本。

path	版本
/face/paas/feature/algorithm/list	1.0.0

入参：无

出参：使用通用结果类型，data域是列表，见下表的详细说明：

字段	类型	备注
id	String	算法ID
provider	String	算法供应商
name	String	算法名称
version	String	算法版本号

示例：

```
{
  "code": 200,
  "message": "success",
  "data": [
    {
      "id": "1A81908C",
      "name": "HV09.1",
      "provider": "成都华安视讯有限公司",
      "version": "HV09.1"
    },
    {
      "id": "2068536E",
      "name": "HV10.1",
      "provider": "成都华安视讯有限公司",
      "version": "HV10.1"
    }
  ]
}
```

API：获取算法配置

获取当前项目已经配置的人脸特征提取算法版本。

path	版本
/face/paas/feature/algorithm/get	1.0.0

入参：无（项目信息由appKey决定）

出参：使用通用结果类型，data域是对象，见下表的详细说明：

字段	类型	备注
id	String	算法ID
provider	String	算法供应商
name	String	算法名称
version	String	算法版本号

示例：

```
{
  "code": 200,
  "message": "success",
  "data": {
    "id": "1A81908C",
    "name": "HV09.1",
    "provider": "成都华安视讯有限公司",
    "version": "HV09.1"
  }
}
```

API：设置算法配置

设置当前项目的人脸特征提取算法版本。

path	版本
/face/paas/feature/algorithm/set	1.0.0

入参：

入参	类型	必填	说明

algorithmId	String	是	算法ID

示例：

```
{
  "algorithmId": "1A81908C"
}
```

出参：使用通用结果类型，无data域。

示例：

```
{
  "code": 200,
  "message": "success"
}
```

API：清空算法配置

清空当前项目已经配置的人脸特征提取算法版本。

path	版本
/face/paas/feature/algorithm/clear	1.0.0

入参：无（项目信息由appKey决定）

出参：使用通用结果类型，无data域。

示例：

```
{
  "code": 200,
  "message": "success"
}
```

API：保存人脸特征值

传入人脸底库图片，使用项目已配置的算法进行特征提取，并把特征值保存在平台。

说明：

1. 图片以URL或base64编码的格式传入，宽、高不超过800像素，文件大小不超过1MB。
2. 一个用户账号只保存最新传入的一张底库，该接口兼具新增和更新底库的功能。
3. 通过该接口更新底库以后，会自动把新底库下发给已关联的设备。

path	版本
/face/paas/feature/save	1.0.0

入参：

入参	类型	必填	说明
identityId	String	是	用户统一身份ID
imageUrl	String	否	图片URL（图片URL和图片base64数据，两者必填其一，优先使用imageUrl）
imageBase64	String	否	图片base64数据（图片URL和图片base64数据，两者必填其一，优先使用imageUrl）
userName	String	否	用户名

示例：

```
{
  "identityId": "50abop0914f8552de72e31e9ef1b96e5a5f90509",
  "imageUrl": "https://getwallpapers.com/wallpaper/full/4/0/4/71062.jpg",
  "userName": "阿里云"
}
```

出参：使用通用结果类型，无data域。

示例：

```
{
  "code": 200,
  "message": "success"
}
```

API：删除人脸特征值

删除已保存在人脸底库特征数据。

说明：

1. 通过该接口删除底库以后，会自动把底库从已关联的设备中移除。

path	版本
/face/paas/feature/delete	1.0.0

入参：

入参	类型	必填	说明
identityId	String	是	用户统一身份ID

示例：

```
{
  "identityId": "50abop0914f8552de72e31e9ef1b96e5a5f90509"
}
```

出参：使用通用结果类型，无data域。

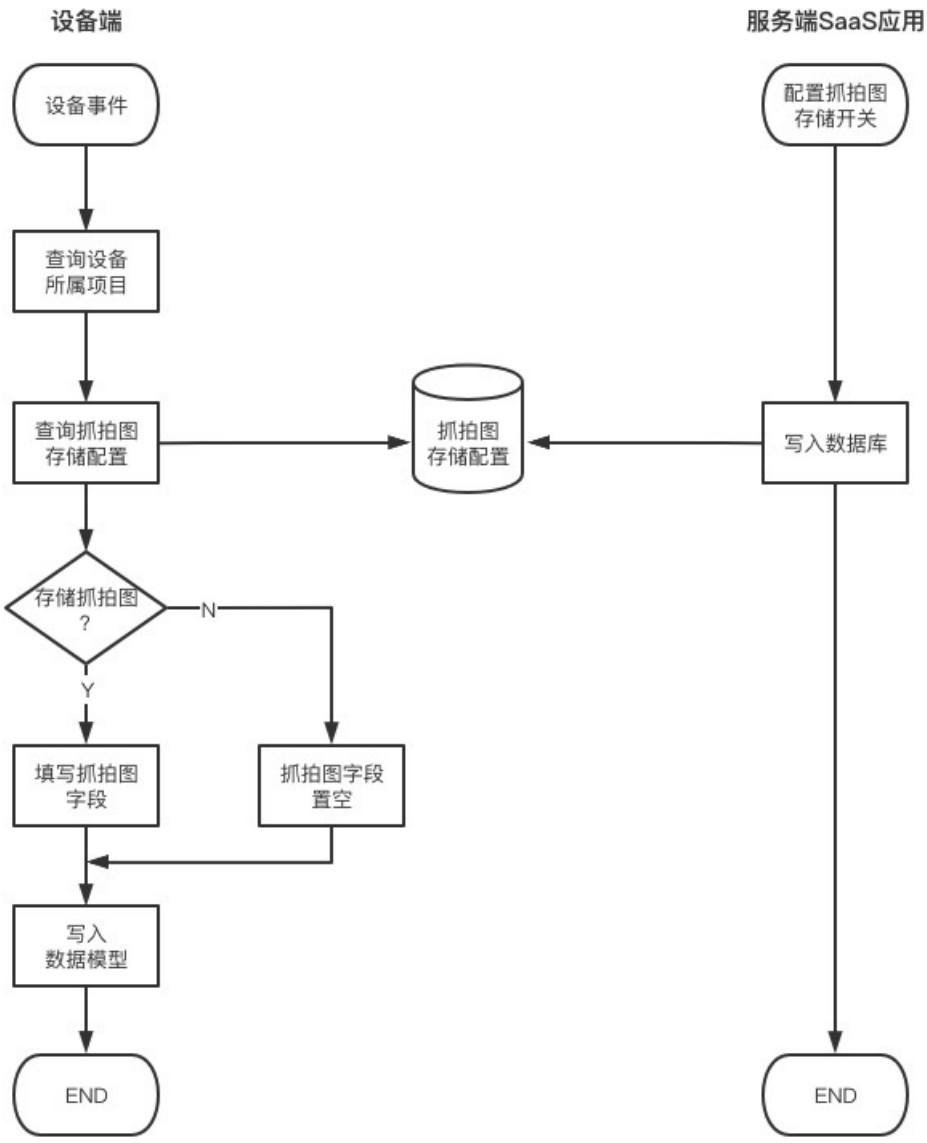
示例：

```
{
  "code": 200,
  "message": "success"
}
```

1.2.4. 人脸抓拍图存储控制

概述

在人脸识别门禁服务中，每一条由设备端上报的通行记录，都携带一张用户在设备上刷脸留下的抓拍图（详见数据模型人脸通行事件）。根据最新的网络安全要求，人脸图片属于用户的敏感数据，是否上报到云端，可以由上层的业务应用进行控制。



项目级配置与租户级配置

抓拍图的存储控制开关可以配置在项目维度（比如SI项目、数字社区的小区）或者租户维度（阿里云账号）。项目级配置优先级高于租户级配置：**如果设备属于项目，则优先查询项目级配置，项目级配置不存在的情况下，查询租户级配置；如果设备不属于项目，则查询租户级配置。**

兼容性说明

2021年11月15日之前接入的项目和租户，默认开启此存储开关。2021年11月15日及之后接入的新租户和项目，没有配置默认开关。在查不到开关配置的情况下，默认作为关闭来处理。

由于上述项目级配置和租户级配置的优先级的存在，2021年11月15日以后，如果先前接入的租户（阿里云账号）创建新的项目，虽然该项目没有配置开关关闭，但由于租户级开关是开启的，因此这个新项目的开关实际上依然是开启状态。

API：设置抓拍图存储开关

路径：/face/paas/snapshot/config/set

版本：1.0.0

入参	类型	必填	说明
configLevel	String	是	配置级别：TENANT - 租户；PROJECT - 项目
recordSnapshot	Boolean	是	是否在通行记录中存储人脸抓拍图

出参	类型	说明
code	Integer	响应码
message	String	响应消息

参数示例：

入参

```
{
  "configLevel": "PROJECT",
  "recordSnapshot": true
}
```

出参

```
{
    "code": 200,
    "message": "success"
}
```

API：获取抓拍图存储开关

路径：/face/paas/snapshot/config/get

版本：1.0.0

入参	类型	必填	说明
configLevel	String	是	配置级别：TENANT - 租户；PROJECT - 项目

出参	类型	说明

code	Integer	响应码
message	String	响应消息
data	Boolean	是否在通行记录中存储人脸抓拍图

参数示例：

入参

```
{
  "configLevel": "PROJECT"
}
```

出参

```
{
    "code": 200,
    "message": "success",
    "data": true
}
```

1.2.5. 人梯联动

使用说明

- 本文档提供的接口用于触发梯控设备呼梯，为设备联动或呼梯行为提供所需的数据。
- 本文档提供的接口需要使用应用服务平台的项目appKey调用，否则可能会导致查询不到对应的边缘集群，数据无法正常下发。
- 相关的门禁设备和梯控设备需要绑定到边缘集群所在的项目。

接口列表

梯控设备数据管理

接口名称	接口路径	当前版本
新增设备数据	/elevator/entrance/device/add	1.0.2
更新设备数据	/elevator/entrance/device/update	1.0.2
删除设备数据	/elevator/entrance/device/delete	1.0.2

查询设备数据	/elevator/entrance/device/query	1.0.2

梯控用户数据管理

接口名称	接口路径	当前版本
保存用户数据	/elevator/entrance/user/save	1.0.2
删除用户数据	/elevator/entrance/user/delete	1.0.2
查询用户数据	/elevator/entrance/user/query	1.0.2

梯控操作接口

接口名称	接口路径	当前版本
远程呼梯	/elevator/control/elevator/call	1.0.1
调用设备服务（梯控泛化接口）	/cloud/thing/service/invoke	1.0.1

API：新增设备数据

新增梯控设备的联动配置。

path	版本
/elevator/entrance/device/add	1.0.2

入参

名称	类型	是否必填	说明

controlMode	String	是	梯控联动模式枚举值 CALL_ONLY：呼梯到起始楼层 CALL_WITH_TARGET：呼梯到起始楼层，并指定一个或多个目标楼层 RELEASE_ONLY：轿厢内，释放一个或多个目标楼层 ACTIVATE_ONLY：轿厢内，点亮单个目标楼层
elevatorIotId	String	是	梯控设备id
triggerIotId	String	是	触发联动场景的设备id，比如门禁设备
triggerFloor	String	否	触发联动场景的设备所在楼层，对于轿厢外的联动模式（CALL_ONLY和CALL_WITH_TARGET），该参数必填
triggerRoomNo	String	否	触发联动场景的设备所在楼层的房间号（多门电梯使用）
triggerElevatorId	String	否	触发联动场景的设备所在的轿厢编号，用于轿厢内的联动场景，对于轿厢内的联动模式（RELEASE_ONLY和ACTIVATE_ONLY），该参数必填
delaySeconds	Integer	否	延迟执行的秒数，不填则表示立即执行，最大支持300秒
executionNode	String	否	执行节点，当前仅支持EDGE，即边缘执行

出参

名称	类型	说明
code	Integer	响应码
message	String	响应消息
data	String	数据项id

API：更新设备数据

新增或更新门禁设备和梯控设备的联动关系，并指定门禁设备所在的楼层和联动呼梯的延迟执行时间。

path	版本
/elevator/entrance/device/update	1.0.2

入参

入参	类型	是否必填	说明
dataId	String	是	新增设备数据接口返回的数据项id
triggerFloor	String	否	触发联动场景的设备所在楼层
triggerRoomNo	String	否	触发联动场景的设备所在楼层的房间号（多门电梯使用）
delaySeconds	Integer	否	延迟执行的秒数

出参

出参	类型	说明
code	Integer	响应码
message	String	响应消息

API：删除设备数据

删除门禁设备和梯控设备的联动关系。

path	版本
/elevator/entrance/device/delete	1.0.2

入参

入参	类型	是否必填	说明
dataId	String	是	代表联动关系的数据项id

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息

API：查询设备数据

查询门禁设备和梯控设备的联动关系。

path	版本
/elevator/entrance/device/query	1.0.2

入参

入参	类型	是否必填	说明
dataId	String	是	代表联动关系的数据项id

出参

出参	类型	说明
code	Integer	响应码
message	String	响应消息
data	JSONObject	响应数据
- dataId	String	代表联动关系的数据项id
- controlMode	String	梯控联动模式
- elevatorIotId	String	梯控设备id

- triggerIotId	String	触发联动场景的设备id
- triggerFloor	String	触发联动场景的设备所在楼层
- triggerRoomNo	String	触发联动场景的设备所在楼层的房间号
- triggerElevatorId	String	触发联动场景的设备所在的轿厢编号
- delaySeconds	Integer	延迟执行的秒数
- executionNode	String	执行节点，当前仅支持EDGE，即边缘执行

API: 保存用户数据

在指定的联动关系下，保存用户目标楼层和参数过期时间。

注意：该接口不会把权限下发给对应的门禁设备，调用方需另外使用门禁服务的接口下发门禁权限，否则无法实现联动。

path	版本
/elevator/entrance/user/save	1.0.2

入参

入参	类型	是否必填	说明
dataId	String	是	代表联动关系的数据项id
identityId	String	是	用户统一身份id
expiryTime	String	否	参数过期时间，过期后参数自动失效，用于门禁权限（如二维码）带有效期的使用场景，不填则表示永久有效。格式：YYYY-MM-DD hh:mm:ss
userTargetList	List	否	用户的目标楼层列表： 在CALL_ONLY模式下可空，其他模式必填；在ACTIVATE_ONLY模式下，只支持1条记录；其他模式最多支持100条记录。

-targetFloor	String	是	目标楼层
-targetRoomNo	String	否	目标楼层房间号

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息

API：删除用户数据

在指定的联动关系下，删除用户目标楼层和参数过期时间。该接口不会删除已经下发给门禁设备的权限数据。

path	版本
/elevator/entrance/user/delete	1.0.2

入参

入参	类型	是否必填	说明
dataId	String	是	代表联动关系的数据项id
identityId	String	是	用户统一身份id

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息

API：查询用户数据

在指定的联动关系下，查询用户目标楼层和参数过期时间。

path	版本
/elevator/entrance/user/query	1.0.2

入参

入参	类型	是否必填	说明
dataId	String	是	代表联动关系的数据项id
identityId	String	是	用户统一身份id

出参

出参	类型	说明
code	Integer	响应码
message	String	响应消息
data	JSONObject	响应数据
- dataId	String	代表联动关系的数据项id
- identityId	String	用户统一身份id
- expiryTime	String	参数过期时间
- userTargetList	List	用户的目标楼层列表
-- targetFloor	String	目标楼层
-- targetRoomNo	String	目标楼层房间号

API：远程呼梯

指定起始楼层和目标楼层进行呼梯操作。

path	版本
/elevator/control/elevator/call	1.0.1

入参

入参	类型	是否必填	说明
iotId	String	是	梯控设备id
startFloor	String	是	起始楼层
startRoomNo	String	否	起始楼层房间号
targetFloor	String	否	目标楼层
targetRoomNo	String	否	目标楼层房间号

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息

API: 调用设备服务（梯控泛化接口）

接口定义

path	版本	是否需要登录
/cloud/thing/service/invoke	1.0.1	否

入参

字段	类型	是否必传	备注
iotId	String	是	门禁设备的iotId
identifier	String	是	设备服务标识符
args	JSNOObject	是	服务的输入参数

出参

返回结果使用通用结果类型，data域为设备服务的输出参数。

> **说明**
>
> 该接口适用于梯控相关的所有物模型服务。

梯控物模型服务定义

1. **云端呼梯**

CloudCallElevator

功能类型：服务

调用方式：同步

输入参数：

参数名称	参数标识	数据类型	长度限制
电梯编号	ElevatorID	text	2048
起始楼层	StartFloor	text	2048
起始楼层房间号	StartRoomNumber	text	2048
目标楼层	TargetFloor	text	2048
目标楼层房间号	TargetRoomNumber	text	2048
目标方向	TargetDirection	enum (枚举型) 1：上行；2：下行	-

输出参数：

参数名称	参数标识	数据类型
结果编码	code	int
结果说明	message	text

结果说明（下同）：

结果编码（code）	结果说明（message）	
0	成功	success
4	发送到设备失败	send to device fail

2. **获取电梯状态**

QueryElevatorStatus

功能类型：服务

调用方式：同步

输入参数：无

输出参数：

参数名称	参数标识	数据类型
电梯状态	ElevatorStatus	array (数组) 元素类型：STRUCT，元素定义详见下方
结果编码	code	int
结果说明	message	text

电梯状态（ElevatorStatus）数组元素定义

参数名称	参数标识	数据类型
电梯编号	ElevatorID	text

轿厢当前楼层	CurrentPosition	text
轿厢运行状态	CarStatus	ENUM -1：未知；0：停止；1：运行
轿厢运行方向	CarDirection	ENUM -1：未知；0：无方向；1：上行；2：下行

结果说明（下同）：

结果编码（code）	结果说明（message）	
0	成功	success
4	发送到设备失败	send to device fail

3. **获取电梯信息**

QueryElevatorInfo

功能类型：服务

调用方式：同步

输入参数：无

输出参数：

参数名称	参数标识	数据类型
电梯控制模式	ElevatorMode	text 当前定义列表： "SINGLE"：单控；"GROUP"：群控；"ALL"：双模（单控和群控）
电梯信息	ElevatorInfo	array 元素类型：STRUCT，元素定义详见下方
电梯楼层房间信息	FloorRoomInfo	array 元素类型：STRUCT，元素定义详见下方
结果编码	code	int

结果说明	message	text

电梯信息(ElevatorInfo)数组元素定义

参数名称	参数标识	数据类型
电梯编号	ElevatorID	text
电梯名称	ElevatorName	text
双开门	IsDualCardDoor	Bool

电梯楼层房间信息（FloorRoomInfo）数组元素定义

参数名称	参数标识	数据类型
电梯编号	ElevatorID	text
楼层	Floor	text
房间号	RoomNumber	text

结果说明（下同）：

结果编码（code）	结果说明（message）	
0	成功	success
4	发送到设备失败	send to device fail

4. 设置人梯联动开关

SetElevatorRuleState

功能类型：服务

调用方式：同步

输入参数：

参数名称	参数标识	数据类型

使能状态	Availability	text "disable"：禁用; "enable"：使能

输出参数：

参数名称	参数标识	数据类型
结果编码	code	int
结果说明	message	text

结果说明（下同）：

结果编码（code）	结果说明（message）	
0	成功	success
4	发送到设备失败	send to device fail

5. **获取人梯联动开关**

GetElevatorRuleState

功能类型：服务

调用方式：同步

输入参数：无

输出参数：

参数名称	参数标识	数据类型
使能状态	Availability	text "disable"：禁用; "enable"：使能
结果编码	code	int
结果说明	message	text

结果说明（下同）：

结果编码（code）	结果说明（message）	
0	成功	success
4	发送到设备失败	send to device fail

使用场景示例

场景1：轿厢内的身份验证与自动点亮目标楼层

场景描述：

电梯轿厢内装有刷卡设备，当用户进入轿厢刷卡时，系统自动为用户点亮目标楼层。

相关数据：

梯控系统iotId：A

轿厢编号：X

刷卡设备iotId：B

用户id：500eopb710fa6e2af4e85fcae3388cdb3dad2900

用户楼层房间号：10-01

使用接口：

1、新增设备数据，指定梯控设备与刷卡设备的联动关系，获取dataId

```
{
  "controlMode": "ACTIVATE_ONLY",
  "elevatorIotId": "A",
  "triggerIotId": "B",
  "triggerElevatorId": "X"
}
```

2、保存用户数据，指定用户的目标楼层（该场景下至多为用户指定1个目标楼层）

```
{
  "dataId": "xxx",
  "identityId": "500eopb710fa6e2af4e85fcae3388cdb3dad2900",
  "userTargetList": [
    {
      "targetFloor": "10",
      "targetRoomNo": "01"
    }
  ]
}
```

场景2：轿厢内的身份验证与自动释放目标楼层

场景描述：

电梯轿厢内装有刷二维码设备，当用户进入轿厢刷二维码时，系统自动为用户释放多个目标楼层，用户可以按需点选需要到达的目标楼层。

相关数据：

梯控系统iotId：A

轿厢编号：X

刷二维码设备iotId：C

用户id：500eopb710fa6e2af4e85fcae3388cdb3dad2900

用户楼层房间号：10-01、11-02、12（房间号未知或不需要指定)

使用接口：

1、新增设备数据，指定梯控设备与刷二维码设备的联动关系，获取dataId

```
{
  "controlMode": "RELEASE_ONLY",
  "elevatorIotId": "A",
  "triggerIotId": "C",
  "triggerElevatorId": "X"
}
```

2、保存用户数据，指定用户的目标楼层（该场景下至多为用户指定100个目标楼层）

```
{
  "dataId": "xxx",
  "identityId": "500eopb710fa6e2af4e85fcae3388cdb3dad2900",
  "userTargetList": [
    {
      "targetFloor": "10",
      "targetRoomNo": "01"
    },
    {
      "targetFloor": "11",
      "targetRoomNo": "02"
    },
    {
      "targetFloor": "12"
    }
  ]
}
```

场景3：单元门禁的身份验证与联动呼梯

场景描述：

楼栋单元内装有人脸识别门禁，用户需要先通过门禁才能进入电梯。此时，定义门禁设备所在的楼层为起始楼层，用户居住或办公需要到达的楼层为目标楼层。

该场景下当用户刷脸通过门禁设备时，门禁自动向梯控系统发送呼梯指令，梯控系统自动把其中一部电梯呼叫到起始楼层，并在电梯到达后自动**点亮**轿厢内的目标楼层或者**释放**轿厢内的多个目标楼层权限。

注意：在电梯到达后，梯控系统的具体行为是“点亮”还是“释放”，这取决于梯控系统的本地化配置，且在一般情况下，如果本地配置为“点亮”，那么不支持用户录入多个目标楼层。

相关数据：

梯控系统iotId：A

人脸识别门禁iotId：D

人脸识别门禁所在楼层：1

用户ID：500eopb710fa6e2af4e85fcae3388cdb3dad2900

用户楼层房间号：10-01

使用接口：

1、新增设备数据，指定梯控设备与刷二维码设备的联动关系，获取dataId

```
{
  "controlMode": "CALL_WITH_TARGET",
  "elevatorIotId": "A",
  "triggerIotId": "D",
  "triggerFloor": "1"
}
```

2、保存用户数据，指定用户的目标楼层

```
{
  "dataId": "xxx",
  "identityId": "500eopb710fa6e2af4e85fcae3388cdb3dad2900",
  "userTargetList": [
    {
      "targetFloor": "10",
      "targetRoomNo": "01"
    }
  ]
}
```

场景4：单元门禁的身份验证与联动呼梯到起始楼层

场景描述：

楼栋单元内装有人脸识别门禁，用户需要先通过门禁才能进入电梯。此时，定义门禁设备所在的楼层为起始楼层，用户居住或办公需要到达的楼层为目标楼层。

该场景下当用户刷脸通过门禁设备时，门禁自动向梯控系统发送呼梯指令，梯控系统自动把其中一部电梯呼叫到起始楼层。该场景下通过接口录入的用户数据仅用于判断是否触发联动规则，其中的目标楼层并不会生效。

相关数据：

梯控系统iotId：A

人脸识别门禁iotId：D

人脸识别门禁所在楼层：1

用户ID：500eopb710fa6e2af4e85fcae3388cdb3dad2900

用户楼层房间号：10-01

使用接口：

同场景3

场景5：室内呼梯

场景描述：

用户在家里通过手机或语音助手主动呼叫电梯，并指定要到达1楼。此时，定义用户居住或办公所在的楼层为起始楼层，用户需要到达的楼层为目标楼层。

相关数据：

梯控系统iotId：A

用户所在楼层房间号：10-01

使用接口：

1、远程呼梯

```
{
  "iotId": "A",
  "startFloor": "10",
  "startRoomNo": "01",
  "targetFloor": "1"
}
```

历史版本

保存用户数据（1.0.1）

在指定的联动关系下，保存用户权限以及用户目标楼层和参数过期时间。

注意：该接口不会把权限下发给对应的门禁设备，调用方需另外使用门禁服务的接口下发门禁权限，否则无法实现联动。

path	版本
/elevator/entrance/user/save	1.0.1

入参

入参	类型	是否必填	说明
dataId	String	是	代表联动关系的数据项id
paramType	String	是	用户参数类型，支持IDENTITY/QRCODE/CARD，分别表示人脸、二维码、刷卡权限

paramValue	String	是	用户参数值，由paramType决定其取值含义： IDENTITY - 人脸用户的identityId QRCODE - 二维码 CARD - 卡号 最大长度支持1024位。
expiryTime	String	否	参数过期时间，过期后参数自动失效，用于门禁权限（如二维码）带有效期的使用场景，不填则表示永久有效。格式：YYYY-MM-DD hh:mm:ss
userTargetList	List	是	用户的目标楼层列表： 在ACTIVATE_ONLY模式下，只支持1条记录；其他模式最多支持100条记录
-targetFloor	String	是	目标楼层
-targetRoomNo	String	否	目标楼层房间号

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息

删除用户数据（1.0.1）

在指定的联动关系下，删除门禁用户权限以及用户目标楼层和参数过期时间。该接口不会删除已经下发给门禁设备的权限数据。

path	版本
/elevator/entrance/user/delete	1.0.1

入参

入参	类型	是否必填	说明

dataId	String	是	代表联动关系的数据项id
paramType	String	是	用户参数类型，支持IDENTITY/QRCODE/CARD，分别表示人脸、二维码、刷卡权限
paramValue	String	是	用户参数值，由paramType决定其取值含义： IDENTITY - 人脸用户的identityId QRCODE - 二维码 CARD - 卡号

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息

查询用户数据（1.0.1）

在指定的联动关系下，查询门禁用户权限以及用户目标楼层和参数过期时间。

path	版本
/elevator/entrance/user/query	1.0.1

入参

入参	类型	是否必填	说明
dataId	String	是	代表联动关系的数据项id
paramType	String	是	用户参数类型，支持IDENTITY/QRCODE/CARD，分别表示人脸、二维码、刷卡权限

paramValue	String	是	用户参数值，由paramType决定其取值含义： IDENTITY - 人脸用户的identityId QRCODE - 二维码 CARD - 卡号

出参

出参	类型	说明
code	Integer	响应码
message	String	响应消息
data	JSONObject	响应数据
- dataId	String	代表联动关系的数据项id
- paramType	String	用户参数类型
- paramValue	String	用户参数值
- expiryTime	String	参数过期时间
- userTargetList	List	用户的目标楼层列表
-- targetFloor	String	目标楼层
-- targetRoomNo	String	目标楼层房间号

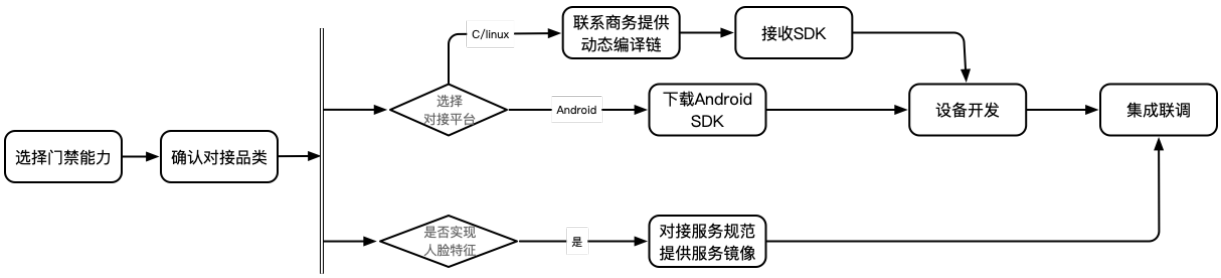
1.3. 设备接入

1.3.1. 接入流程

1. 获取支持

厂商对接时先联系商务确认对接支持钉钉群。

2. 对接流程



1. 根据实际硬件功能和业务需求选择门禁能力*（见《门禁物模型》中第1节关于门禁能力的说明）*
2. 确认对接品类 *（见《门禁物模型》中第2节关于物模型能力的说明）*
3. 选择对接平台，如果是C/Linux平台需要提供动态编译链，IOT支持编译SDK后返回给厂商；如果是Android平台可以直接下载。

   厂商获取到SDK后进入开发；
4. 如果要实现人脸特征下发，根据《人脸特征算法上云》的说明提供云端托管服务镜像，IOT配合部署；
5. 最后进入联调环节。

1.3.2. 门禁物模型

本文档说明门禁设备对接的物模型和功能分类，目的是指导厂商如何根据设备能力选择产品品类和能力集。

厂商对接时参考文档中第1节物模型选择需要的能力，再到第2节的品类支持能力中选择可以满足需求的品类。

1. 物模型

门禁设备的能力各不相同，很多设备会是几个能力的组合。IOT针对不同的品类也定义了不同的能力。厂商要根据设备的能力需求，确定要对接的品类是否能满足需求。

下面梳理了各种能力的组合，选定后再根据第2节的品类的能力列表选择适配对接的品类。

物模型中包含了服务、属性和事件定义。为了描述方案，下面用前缀标识出需要的物模型成员类型：

[P] - 属性，[E] - 事件，[S] - 服务

对于物模型中服务、属性和事件更详细的描述，在对接品类时可以查看。

能力	需要的物模型 [P] - 属性，[E] - 事件，[S] - 服务	说明
二维码直发	[S] 新增二维码 addQrCode [S] 删除二维码 delQrCode [E] 通行事件 passEvent [E] 报警事件 alarmEvent	早期直接通过物模型下发的二维码，目前不再使用，只维护存量设备。
密码直发	[S] 设置密码 setPassword [E] 通行事件 passEvent [E] 报警事件 alarmEvent	早期直接通过物模型下发的密码门禁，目前不再使用，只维护存量设备。

能力	需要的物模型 [P] - 属性，[E] - 事件，[S] - 服务	说明
卡同步	[S] 同步权限 syncPermissions [E] 异常刷卡 abnormalCreditCard [E] 通行事件 passEvent [E] 报警事件 alarmEvent	用权限文件下发的同步门禁，有独立事件上报异常刷卡；
蓝牙同步	[S] 同步权限 syncPermissions [E] 异常刷卡 abnormalBleCard [E] 通行事件 passEvent [E] 报警事件 alarmEvent	用权限文件下发的同步门禁，有独立事件上报异常刷卡；
二维码同步	[S] 同步权限 syncPermissions [E] 通行事件 passEvent [E] 报警事件 alarmEvent	用权限文件下发的同步门禁
加密二维码	[P] RSA公钥（加密二维码）RSAPublicKey	在二维码同步门禁基础上，增加加密功能
人脸 1.0	[S] 同步人脸库图片 SyncFacePictures [S] 查询人脸布控进度 QuerySyncPicSchedule [S] 查询布控成功的人脸图 QueryAddedUserInfo [S] 查询权限数 QueryFacePermTotal [E] 人脸比对事件上报 OnMatched [E] 人脸检测事件上报 OnDetect	门禁设备对接推荐的接入版本 通过“部署”、“查询部署进度”和“查询全量权限”完成整个部署流程。 支持人脸识别和陌生人事件上报； 支持设备权限数查询；
人脸 2.0	[S] 查询设备数据校验码 QueryDataCheckCode [S] 查询用户是否存在 QueryUserData [S] 同步人脸数据 SyncFaceData [S] 同步人脸图片失败 OnFacePicError [E] 人脸库同步确认 OnSync [E] 上报设备数据校验码 PublishDataCheckCode [E] 智能告警 IntelligentAlarm	门禁设备2.0版本对接，无特殊需求不推荐对接。

能力	需要的物模型 [P] - 属性，[E] - 事件，[S] - 服务	说明
人脸特征下发[1]	[P] 算法版本 FaceAlgorithmVersion [P] 固件版本 FirmwareVersion [P] 支持人脸特征开关 SupportFaceFeature [S] 部署人脸特征 SyncFaceFeatures	在人脸1.0基础上，增加人脸特征下发功能
可视对讲	[S] 下发手机号校验结果 NotifyPhoneNumberCheckResult [S] 颁发 SyncSipNumber [S] 同步组号和房间号 SyncGroupIdAndRoomId [S] 查询 querySipNumber [E] 开门事件 doorOpenEvent [S] 校验手机号 checkPhoneNumber	在人脸1.0基础上，增加可视对讲功能。可与室内天猫精灵对接。
远程开门-参数版	[S] 远程开门 RemoteControl	在健康码对接的基础上，支持提示消息的开门操作
远程开门-无参版	[S] 远程开门 remoteControl	无参数版本，只完成开门动作。

注：

- *[1] 人脸特征下发只适用于人脸1.0物模型。*
- *[2] 物的能力不一定只适用唯一的能力，有些服务或事件是通用的，可以支持多种能力。*

2. 支持的品类

下面列出门禁服务支持的品类和对应的能力集合

分类	品类	功能
智能生活 -> 家居安防	人脸识别能力模型/FaceRecognitionCapabilityModel	人脸下发 1.0（不推荐）
	摄像头/Camera	
智能城市 -> 公共服务	人脸识别门禁/FaceRecognition	人脸下发 1.0 人脸特征下发

分类	品类	功能
智能园区	刷卡门禁/CardAccess	卡同步门禁 同步门禁
	密码门禁/PasswordAccess	密码直发 远程开门（无参版）
	二维码门禁/QrCodeAccess	二维码直发 远程开门（无参版）
	二维码门禁机/QrCodeAccessControl	同步门禁 远程开门
	蓝牙门禁/BluetoothAccess	蓝牙同步门禁 同步门禁
	多功能门禁/MultiAccessControl	人脸下发 1.0 人脸特征下发 人脸下发 2.0-不推荐 SIP可视对讲 同步门禁 加密二维码 远程开门
	可视对讲机门口机/VideoIntercomDoor	可视对讲
边缘计算	人脸识别机/FaceRecognizeDevice	人脸下发 2.0
	摄像头边缘节点/VisionAccessNode	AIBox，支持人脸下发 2.0
自定义品类	自定义品类/CustomCategory	无预定义功能，在SI中选择能力创建

注：上面的能力列表只面向门禁领域专用能力，实际品类物模型可能包含其它能力，不在上表中列出。

权限计数服务

以服务形式从云端调用设备端执行实时统计。此统计直接查询设备存储，不受SDK影响。

```
{
    "services":[
        {
            "identifier":"QueryFacePermTotal",
            "method":"thing.service.QueryFacePermTotal",
            "name":"查询设备上人脸权限总数",
            "required":true,
            "callType":"sync",
            "inputData":[

            ],
            "outputData":[
                {
                    "name":"当前人脸库权限总数",
                    "identifier":"FacePermTotal",
                    "dataType":{
                        "specs":{
                            "min":"0",
                            "unitName":"无",
                            "max":"1000000",
                            "step":"1"
                        },
                        "type":"int"
                    }
                }
            ]
        }
    ]
}
```

人脸特征物模型

属性：

- 特征下发开关属性 SupportFaceFeature
- 人脸算法版本 FaceAlgorithmVersion
- 设备固件版本 FirmwareVersion

```
{
    "properties":[
       {
            "identifier":"SupportFaceFeature",
            "dataType":{
                "specs":{
                    "0":"支持",
                    "1":"不支持"
                },
                "type":"bool"
            },
            "name":"设备是否支持人脸特征下发"
        },
        {
            "identifier":"FaceAlgorithmVersion",
            "dataType":{
                "specs":{
                    "length":"65"
                },
                "type":"text"
            },
            "name":"人脸库算法版本",
            "accessMode":"rw",
            "required":true
        },
        {
            "identifier":"FirmwareVersion",
            "dataType":{
                "specs":{
                    "length":"65"
                },
                "type":"text"
            },
            "name":"设备固件版本",
            "accessMode":"rw",
            "required":true
        }
    ]
}
```

服务

- 人脸特征下发 SyncFaceFeatures

```
{
    "services":[
        {
            "name":"同步人脸库特征",
            "identifier":"SyncFaceFeatures",
            "method":"thing.service.SyncFaceFeatures",
            "required":false,
            "callType":"sync",
            "inputData":[
                {
                    "identifier":"FacePicFeaturesURL",
                    "dataType":{
                        "specs":{
                            "length":"1024"
                        },
                        "type":"text"
                    },
                    "name":"同步特征文件URL地址"
                }
            ],
            "outputData":[
                {
                    "identifier":"SyncPicStatus",
                    "dataType":{
                        "specs":{
                            "0":"成功",
                            "1":"布控中",
                            "2":"下载文件失败",
                            "3":"解析文件失败"
                        },
                        "type":"enum"
                    },
                    "name":"设备同步图片状态值"
                }
            ]
        }
    ]
}
```

参数示例：

```
{"FacePicFeaturesURL":"http://xxxxxx"}
```

特征文件

新增人脸特信息，增加人脸特征字段，类型为字符串数组。

```
features: ["xxxxxxxxxxxxxxxxxx"]
```

完整示例：

```
{
  "groupId": "egqQkuFBPgh8pltWiw1S000100",
  "addFaceInfos": [
    {
      "faceId": "552f3570-4717-4a39-910b-d72542552d89",
      "faceMd5": "36c36fa96920e45ca5f00334675484bc",
      "faceUrl": "http://xxx",
      "userInfo": "{\"userName\": \"Xiao Wang\", \"expire\": \"2020-12-31 23:59:59\", \"pol
icy\": \"DENY\", \"userExtInfo\": \"BLACKLIST\"}",
      "features": [ "xxxxxxxxxxxxxxxxxx" ],
      "index": 0,
      "clientTag": ""
    }
  ],
  "deleteFaceInfos": [
    {
      "faceId": "xxxx"
    },
    {
      "faceId": "yyyy"
    }
  ],
  "isCleanAll": false,
  "isCleanAllGroup": false
}
```

SIP可视对讲物模型

颁发SipNumber（SyncSipNumber）： SaaS应用从SIP server为门禁机申请到SIP number, 通过该服务将SIP number通过阿里LP平台下发到门禁机。

同步组号和房间号 (SyncGroupIdAndRoomId)：当SaaS应用完成SIP Group创建并录入房间息后，通过该服务将GroupID与RoomID的映射关系表格下发到门禁机。门禁机在有人发起可视对讲呼叫时，可以根据房间号查找到对应的GroupID，然后发给SIP server呼叫GroupID里的所有SIP number。

查询SipNumber（querySipNumber）：当SaaS应用想查询某个单元门禁机的SipNumber时，可以使用该服务

下发手机号校验结果该服务用在支持通过IPPBX直接呼叫手机的方案使用。

1.3.3. 同步文件指导

本文档介绍云端向门禁设备同步的文件。

同步权限 syncPermissions

二维码和刷卡门禁通过物模型服务向设备端发送二维码和卡号数据。

服务名

- syncPermissions：同步权限

请求参数

- permissionUrl：String(2048)

响应参数（无）

文件通过URL的形式下发给设备端的LinkKit SDK，由设备端对接LinkKit SDK的物模型回调函数处理下发数据。

二维码文件格式说明：

- qrCodePermissionList：要下发的二维码权限列表。
  - type：变更类型，1-增，2-删，3-改
  - qrCode：二维码
  - startTime：二维码生效时间
  - endTime：二维码失效时间
  - maxScanTimes：最大扫码次数
  - maxScanScope：最大刷码次数作用范围，SHARE - 同一网关下共享; DEVICE - 设备

二维码文件格式示例：

```
{
    "qrCodePermissionList":[
        {
            "type":1,
            "qrCode":"ALIF24DD9END8CD3",
            "startTime":"1568270392000",
            "endTime":"1599892792000",
            "maxScanTimes":600,
            "maxScanScope":"SHARE"
        },
        {
            "type":2,
            "qrCode":"ALIF24DD9END8CD3"
        },
        {
            "type":3,
            "qrCode":"ALIF24DD9END8CD3",
            "startTime":"1568270392000",
            "endTime":"1599892792000",
            "maxScanTimes":600,
            "maxScanScope":"SHARE"
        }
    ]
}
```

发卡文件格式说明：

- tradCardPermissionList：要下发的卡权限列表。
  - type：变更类型，1-增，2-删，3-改
  - cardId：卡号
  - endTime：卡号失效时间
  - userId：用户id

发卡文件格式示例：

```
{
  "tradCardPermissionList": [
    {
      "cardId": "card001",
      "endTime": "1618625794000",
      "type": 3,
      "userId": "50a0a1c68805f48a59b6879e837e684f67171474"
    }
  ]
}
```

下发人脸图 syncFacePictures

人脸门禁1.0物模型的同步人脸库图片服务（SyncFacePictures）的下发参数（FacePicURL）

服务名

- SyncFacePictures：人脸特征下发

请求参数

- FacePicURL: String(2048)

响应参数

- SyncPicStatus：
  - 0 - 成功
  - 1 - 布控中
  - 2 - 下载文件失败
  - 3 - 解析文件失败

文件通过URL的形式下发给人脸SDK，解析后调用SDK开放的回调函数通知设备处理。

【说明】：

- groupId：人脸分组ID，一个分组Link Face支持5W人脸，目前地产线使用门禁机子设备的iotid作为分组ID。
- addFaceInfos：要增加的用户信息列表
  - faceId：要增加的人脸信息唯一标识符（由云端生成）
  - userName：用户姓名
  - index：增加用户信息列表中的索引（一般从0开始，依次累加）
- deleteFaceInfos：要删除的用户信息列表
  - faceId：要删除的人脸信息唯一标识符
- isCleanAll：标识是否删除设备内的groupId指定的分组内所有人脸信息的（false - 否，true - 是）
- isCleanAllGroup：标识是否删除设备内全部人脸信息的标识（false - 否，true - 是）

```
{
  "groupId": "egqQkuFBPgh8pltWiw1S000100",
  "addFaceInfos": [
    {
      "faceId": "552f3570-4717-4a39-910b-d72542552d89",
      "faceMd5": "36c36fa96920e45ca5f00334675484bc",
      "faceUrl": "",
      "userInfo": "{\"userName\": \"Xiao Wang\", \"expire\": \"2020-12-31 23:59:59\", \"pol
icy\": \"DENY\", \"userExtInfo\": \"BLACKLIST\"}",
      "index": 0,
      "clientTag": ""
    }
  ],
  "deleteFaceInfos": [
    {
      "faceId": "xxxx"
    },
    {
      "faceId": "yyyy"
    }
  ],
  "isCleanAll": false,
  "isCleanAllGroup": false
}
```

同步人脸特征 syncFaceFeatures

人脸特征下发是通过单独的物模型服务提供的，形式和参数与同步人脸图非常类似。

服务名

- SyncFaceFeatures：人脸特征下发

请求参数

- FacePicFeaturesURL: String(2048)

响应参数

- SyncPicStatus:
  - 0 - 成功
  - 1 - 布控中
  - 2 - 下载文件失败
  - 3 - 解析文件失败

下面是FacePicFeaturesURL的格式定义（对比人脸图下发增加了features字段，其它相同）。features字段是一个泛化的特征字段，不是狭义的特征向量，而是从原人脸图中解析出来的所有数据，如特征向量、缩略图等。

```
{
  "groupId": "egqQkuFBPgh8pltWiw1S000100",
  "addFaceInfos": [
    {
      "faceId": "552f3570-4717-4a39-910b-d72542552d89",
      "faceMd5": "36c36fa96920e45ca5f00334675484bc",
      "faceUrl": "",
      "features": ["xxxx","xxxx"]
      "userInfo": "{\"userName\": \"Xiao Wang\", \"expire\": \"2020-12-31 23:59:59\", \"pol
icy\": \"DENY\", \"userExtInfo\": \"BLACKLIST\"}",
      "index": 0,
      "clientTag": ""
    }
  ],
  "deleteFaceInfos": [
    {
      "faceId": "xxxx"
    },
    {
      "faceId": "yyyy"
    }
  ],
  "isCleanAll": false,
  "isCleanAllGroup": false
}
```

上报已下发权限

云端通过此服务查询设备上的全量权限。一般情况下由SDK响应。

服务名

- QueryAddedUserInfo 查询布控成功的人脸图

请求参数

- GroupID: String(64) 人脸库组ID

响应参数

- StoreID: String(2048) 人脸布控文件storeID
- Type: Integer 类型
  - 0 - StoreID
  - 1 - FileName
  - 2 - URL
- SyncPicStatus: Integer 查询添加用户信息的布控状态
  - 0 - 成功
  - 1 - 布控中

下面是StoreID指向的文件说明：

- groupId：人脸分组ID;

- currentFaceInfos：当前所有人脸信息列表
- failedFaceInfos：最新一次人脸库同步中失败的人脸信息列表

注意：此数组中每个元素的内容和同步人脸库文件中addFaceInfos数组中的内容一致。


```
{
  "groupId": "egqQkuFBPgh8pltWiw1S000100",
  "currentFaceInfos": [
    {
      "faceId": "552f3570-4717-4a39-910b-d72542550001",
      "userInfo": "{\"userName\": \"Zhang San\", \"expire\": \"2020-12-31 23:59:59\"}",
      "faceMd5": "f6da8b54a927f32378792179193f2133",
      "index": 0,
      "clientTag": "Demo"
    },
    {
      "faceId": "552f3570-4717-4a39-910b-d72542550002",
      "userInfo": "{\"userName\": \"Li si\", \"expire\": \"2019-12-31 23:59:59\"}",
      "faceMd5": "36c36fa96920e45ca5f00334675484bc",
      "index": 2,
      "clientTag": "Demo"
    }
  ],
  "failedFaceInfos": [
    {
      "faceId": "552f3570-4717-4a39-910b-d72542552d89",
      "faceMd5": "36c36fa96920e45ca5f00334675484bc",
      "faceUrl": "",
      "userInfo": "{\"userName\": \"Xiao Wang\", \"expire\": \"2020-12-31 23:59:59\", \"pol
icy\": \"DENY\", \"userExtInfo\": \"BLACKLIST\"}",
      "index": 0,
      "clientTag": ""
    }
  ]
}
```


1.3.4. 人脸特征算法上云

本文档用来指导厂商提供特征算法上云服务镜像，在保持算法独立性的前提下实现人脸特征下发。

IOT门禁服务向设备下发人脸权限时默认使用图片的方式，核心原因是为了更好的兼容性，下发图片后由端侧设备计算人脸特征值。之所以采用这种方式，是因为云、端算法不容易统一。

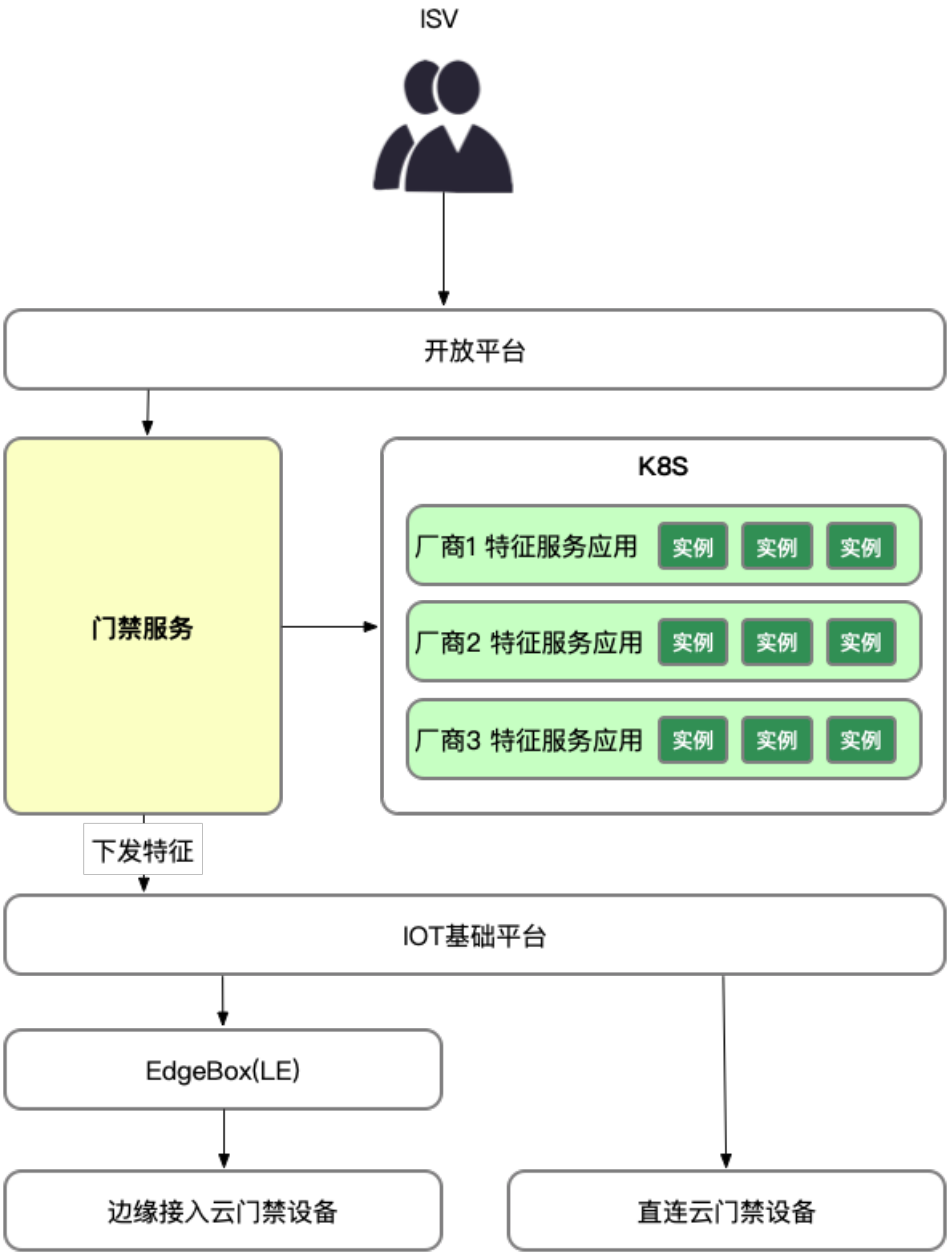
直接下发图片的方式由于需要多次网络交互和本地计算，部署性能较低，影响实施效率。长时间下发容易受到网络状态影响导致下发失败。

厂商根据本文档提供的服务规范，实现托管服务镜像。IOT集成托管服务完成云端部署服务集群，就可以实现特征下发，提高人脸权限的下发性能和成功率。

与人脸图片下发对比，特征下发可以提高10倍的部署性能，部署时间缩短为1/10。

> 注意
>
> 人脸特征算法上云能力只面向人脸1.0 SDK。

1. 架构



流程：

- 厂商按服务规范提供服务镜像；
- IOT负责部署在云端；
- 下发权限时，IOT根据服务提供的业务参数路由到对应的服务计算特征；

2. 服务定义

2.1. 通用定义

docker镜像，提供http标准服务

服务日志写到标准输出 stdout, stderror

协议：http | https

端口：8080

方法：POST

路径：见接口定义

例：

```
http://10.1.2.3:8080/device/face/features/get
```

IOT分配 appKey, appSecret

```
String appKey = "xxx";
String appSecret = "xxx";
long ts = System.currentTimeMillis();
String str = appKey + "." + appSecret + "." + ts;
String sign = md5(str);
```

ts校验：

```
assert( Math.abs( currentTime - ${ts} ) < 3600000 )
```

字段名	类型	非空	说明
sign	String	是	签名
appKey	String	是	IOT颁发的appKey，用于验证签名
ts	Long	是	时间戳，毫秒
requestId	String	是	请求ID，回填给响应。
params	Array|Object		请求参数，可以为空、数组或对象。具体见接口定义。

示例

```
{
  "sign": "32fvdsazxxr4234rdxfsdasdfsda",
  "appKey": "xxxx",
  "ts": 1604494967,
  "requestId": "xxxxxxx",
  "params": {
    ...
  }
}
```

字段名	类型	非空	说明
requestId	String	是	请求ID，回填给响应。
code	Integer	是	响应码，200成功
message	String		非成功时的错误消息，成功时为空。非成功时说明错误原因。
data	Array|Object		成功时的数据部，可以为空、对象或数组。

正常响应

```
{
    "requestId": "32fvdsazxxr4234rdxfsdasdfsda",
    "code": 200,
    "data": {
        ...
    }
}
```

错误响应

```
{
    "requestId":"JymsEenXhMQwMaDb2L68WFFFO3bkyAb7",
    "code":500,
    "message":"服务不可用"
}
```

错误码	说明
200	成功

401	签名错误
403	参数错误
500	系统错误

单服务节点性能基本要求

- **人脸提取特征**

  服务的性能指标：CPU架构不低于5 QPS，GPU架构不低于30 QPS；支持并行处理，并发限制由厂商提供；

- **人脸质量检查**

  服务的性能指标：不低于10 QPS，支持并行处理，并发限制由厂商提供；

- 其它服务请求rt不大于100ms；

厂商提供：

- **人脸提取特征**

  服务的性能指标（qps）和并发数限制；

- **人脸质量检查**

  服务的性能指标（qps）和并发数限制；

- 提供托管服务配置要求，CPU、GPU、存储、网络等；

2.2. 计算人脸特征

根据输入的固件和算法版本，计算人脸特征值，批量接口。

```
/device/face/features/get
```

请求参数的数据部为JSON对象

字段名	类型	非空	说明
productKey	String	是	IOT productKey
firmwareVersion	String	是	固件版本
algorithmVersion	String	是	算法版本
faces	Array<Object>	是	人脸信息，最大长度100

faceId	String	是	人脸ID
faceImageUrl	String	是	人脸下载URL
faceImageMd5	String	是	人脸文件MD5值

响应参数的数据部为JSON对象**数组**

字段名	类型	非空	说明
faceId	String	是	人脸ID
faceFeatures	Array<String>	是	人脸特征，或其它特征预处理结果。 如果返回内容是json，需要的前处理json转义。 该字段会直接透传给设备端，由厂商保证每个特征的顺序和含义。云端不会改变特征字段的值和编码。
code	Integer		异常时的错误码
message	String		异常时的报错信息

请求

```
{
    "sign":"MczHCDuwnfLdOuIWSgSHwby9bLRadLul",
    "appKey":"xxx",
    "ts":1234567890,
    "requestId":"JymsEenXhMQwMaDb2L68WFFFO3bkyAb7",
    "params":{
        "productKey":"XXXX",
        "firmwareVersion":"1.2.3",
        "algorithmVersions":"1.3.2",
        "faces":[
            {
                "faceId":"xxx1",
                "faceImageUrl":"http://xxxx",
                "faceImageMd5":"xxxxxxx"
            },{
                "faceId":"xxx2",
                "faceImageUrl":"http://xxxx",
                "faceImageMd5":"xxxxxxx"
            }
        ]
    }
}
```

响应

```
{
    "requestId":"JymsEenXhMQwMaDb2L68WFFFO3bkyAb7",
    "code":200,
    "message":"xxxx",
    "data":[
        {
            "faceId":"xxx1",
            "faceFeatures":["0.342,343243,243243,432","xxx"]
        },
        {
            "faceId":"xxx2",
            "code":204,
            "msg":"人脸质量低"
        }
    ]
}
```

2.3. 人脸质量检查

人脸图质量检查，批量接口。

这个接口会开放给项目ISV，当项目有人脸检测需求时可以使用云端服务，避免厂商重复部署。ISV使用服务时，可以向厂商咨询使用的算法版本。也可以支持无版本的请求，使用默认的检查算法。

```
/device/face/quality/check
```

请求参数的数据部为JSON对象

字段名	类型	非空	说明
algorithmVersion	String	否	算法版本，如果没有版本号可以不填
faces	Array<Object>	是	人脸信息。最大长度100
faceId	String	是	人脸ID
faceImageUrl	String	是	人脸下载URL
faceImageMd5	String	是	人脸文件MD5值

响应参数的数据部为JSON对象数组

字段名	类型	非空	说明
faceId	String	是	人脸ID
status	String	是	状态，字符串枚举 ACCEPT 接受REJECT 拒绝
message	String		REJECT状态时的错误消息. • Low quality. 质量低 • No face found. 没找到人脸 • More faces. 存在多张人脸厂商可自定义

请求

```
{
    "sign":"MczHCDuwnfLdOuIWSgSHwby9bLRadLul",
    "appKey":"xxx",
    "ts":1234567890,
    "requestId":"JymsEenXhMQwMaDb2L68WFFFO3bkyAb7",
    "params":{
        "faces":[
            {
                "faceId":"xxx1",
                "faceImageUrl":"http://xxxx",
                "faceImageMd5":"xxxxxxx"
            },{
                "faceId":"xxx2",
                "faceImageUrl":"http://xxxx",
                "faceImageMd5":"xxxxxxx"
            }
        ]
    }
}
```

响应

```
{
    "requestId":"JymsEenXhMQwMaDb2L68WFFFO3bkyAb7",
    "code":200,
    "message":"xxxx",
    "data":[
        {
            "faceId":"xxx1",
            "status":"ACCEPT"
        },
        {
            "faceId":"xxx2",
            "status":"REJECT",
            "message":"More faces."
        }
    ]
}
```

2.4.查询服务信息

获取厂商托管服务的基本信息.

```
/service/info/get
```

请求参数的数据部为空。

响应参数的数据部为JSON对象

字段名	类型	非空	说明

manufacturerName	String	是	厂商名称（注册服务用） 支持汉字、字母、数字、横线和下划线，必须由汉字、字母开头，长度不超过64
serviceReleaseTime	String	是	服务发布日期（注册服务的标识信息）
version	String	是	服务版本号（注册服务的标识信息）
qpsLimit	Integer	是	服务的秒级限流
https	Boolean	是	是否支持https，默认为否 false

请求

```
{
    "sign":"MczHCDuwnfLdOuIWSgSHwby9bLRadLul",
    "appKey":"xxx",
    "ts":1234567890,
    "requestId":"JymsEenXhMQwMaDb2L68WFFFO3bkyAb7"
}
```

响应

```
{
    "requestId":"JymsEenXhMQwMaDb2L68WFFFO3bkyAb7",
    "code":200,
    "data":{
        "manufacturerName":"XX公司",
        "serviceReleaseTime":"2020-10-31",
        "version":"1.0.0",
        "qpsLimit":30,
        "https: false
    }
}
```

2.5.查询算法版本列表

查询托管服务支持的算法版本信息。支持配置多个产品的可用算法。

云端缓存算法关系时，忽略固件版本。固件版本信息会作为参数输入给人脸特征服务，具体适配由厂商处理。

```
/service/algorithmversions/supports/get
```

请求参数的数据部为空

响应参数的数据部为JSON对象**数组**。

字段名	类型	非空	说明
productKey	String	是	IOT productKey
algorithmVersions	Array<String>	是	支持版本列表

请求

```
{
    "sign":"MczHCDuwnfLdOuIWSgSHwby9bLRadLul",
    "appKey":"xxx",
    "ts":1234567890,
    "requestId":"JymsEenXhMQwMaDb2L68WFFFO3bkyAb7"
}
```

响应

```
{
    "requestId":"JymsEenXhMQwMaDb2L68WFFFO3bkyAb7",
    "code":200,
    "data":[
        {
            "productKey":"iLxhvTig",
            "algorithmVersions":[
                "v1.xx.xx",
                "v2.xx.xx",
                "v3.xx.xx",
                "v4.xx.xx"
            ]
        },
        {
            "productKey":"3rb5apOH",
            "algorithmVersions":[
                "v1.xx.xx",
                "v2.xx.xx",
                "v3.xx.xx"
            ]
        }
    ]
}
```

2.6.健康检查

查询服务的健康状态。

由K8S集群检查服务状态。

```
/service/health
```

请求参数的数据部为空

响应参数的数据部为JSON对象。

字段名	类型	非空	说明
status	String	是	健康状态，字符串枚举 • UP 健康 • DOWN 故障

请求

```
{
    "sign":"MczHCDuwnfLdOuIWSgSHwby9bLRadLul",
    "appKey":"xxx",
    "ts":1234567890,
    "requestId":"JymsEenXhMQwMaDb2L68WFFFO3bkyAb7"
}
```

响应

```
{
    "requestId":"JymsEenXhMQwMaDb2L68WFFFO3bkyAb7",
    "code":200,
    "data":{
        "status":"UP"
    }
}
```

3. 设备物模型

3.1. 物模型 - 属性

属性定义，上报设备端版本号

- 人脸算法版本（新增）
  - FaceAlgorithmVersion
- 设备固件版本（新增）
  - FirmwareVersion

```
{
    "properties":[
        {
            "identifier":"FaceAlgorithmVersion",
            "dataType":{
                "specs":{
                    "length":"65"
                },
                "type":"text"
            },
            "name":"人脸库算法版本",
            "accessMode":"rw",
            "required":true
        },
        {
            "identifier":"FirmwareVersion",
            "dataType":{
                "specs":{
                    "length":"65"
                },
                "type":"text"
            },
            "name":"设备固件版本",
            "accessMode":"rw",
            "required":true
        }
    ]
}
```

增加设备支持特征下发开关属性

- SupportFaceFeature

```
{
    "properties":[
        {
            "identifier":"SupportFaceFeature",
            "dataType":{
                "specs":{
                    "0":"支持",
                    "1":"不支持"
                },
                "type":"bool"
            },
            "name":"设备是否支持人脸特征下发"
        }
    ]
}
```

3.2. 物模型 - 服务

人脸特征下发

新增人脸特征下发服务

```
{
    "services":[
        {
            "name":"同步人脸库特征",
            "identifier":"SyncFaceFeatures",
            "method":"thing.service.SyncFaceFeatures",
            "required":false,
            "callType":"sync",
            "inputData":[
                {
                    "identifier":"FacePicFeaturesURL",
                    "dataType":{
                        "specs":{
                            "length":"1024"
                        },
                        "type":"text"
                    },
                    "name":"同步特征文件URL地址"
                }
            ],
            "outputData":[
                {
                    "identifier":"SyncPicStatus",
                    "dataType":{
                        "specs":{
                            "0":"成功",
                            "1":"布控中",
                            "2":"下载文件失败",
                            "3":"解析文件失败"
                        },
                        "type":"enum"
                    },
                    "name":"设备同步图片状态值"
                }
            ]
        }
    ]
}
```

参数示例：

```
{"FacePicFeaturesURL":"http://xxxxxx"}
```

特征文件

新增人脸特信息，增加人脸特征字段，类型为字符串数组。

```
features: ["xxxxxxxxxxxxxxxxxx"]
```

完整示例：

```
{
    "groupId":"egqQkuFBPgh8pltWiw1S000100",
    "addFaceInfos":[
        {
            "faceId":"552f3570-4717-4a39-910b-d72542552d89",
            "faceMd5":"36c36fa96920e45ca5f00334675484bc",
            "faceUrl":"http://xxx",
            "userInfo":"{"userName": "Xiao Wang", "expire": "2020-12-31 23:59:59", "policy"
: "DENY", "userExtInfo": "BLACKLIST"}",
            "features":[
                "xxxxxxxxxxxxxxxxx"
            ],
            "index":0,
            "clientTag":""
        }
    ],
    "deleteFaceInfos":[
        {
            "faceId":"xxxx"
        },
        {
            "faceId":"yyyy"
        }
    ],
    "isCleanAll":false,
    "isCleanAllGroup":false
}
```

3.3. SDK

见SDK文档《**LinkFace C SDK Linux设备端V1版开发指南**》

3.4. 适配品类

人脸特征支持的品类：

- 智能城市 -> 公共服务 -> **人脸识别门禁/FaceRecognition**
- 智慧园区 -> **多功能门禁/MultiAccessControl**

1.3.5. 门禁SDK

概述

门禁SDK主要是实现端侧的人脸下发逻辑和端智能的逻辑封装。目的是简化厂商的对接流程，降低成本。

主要版本

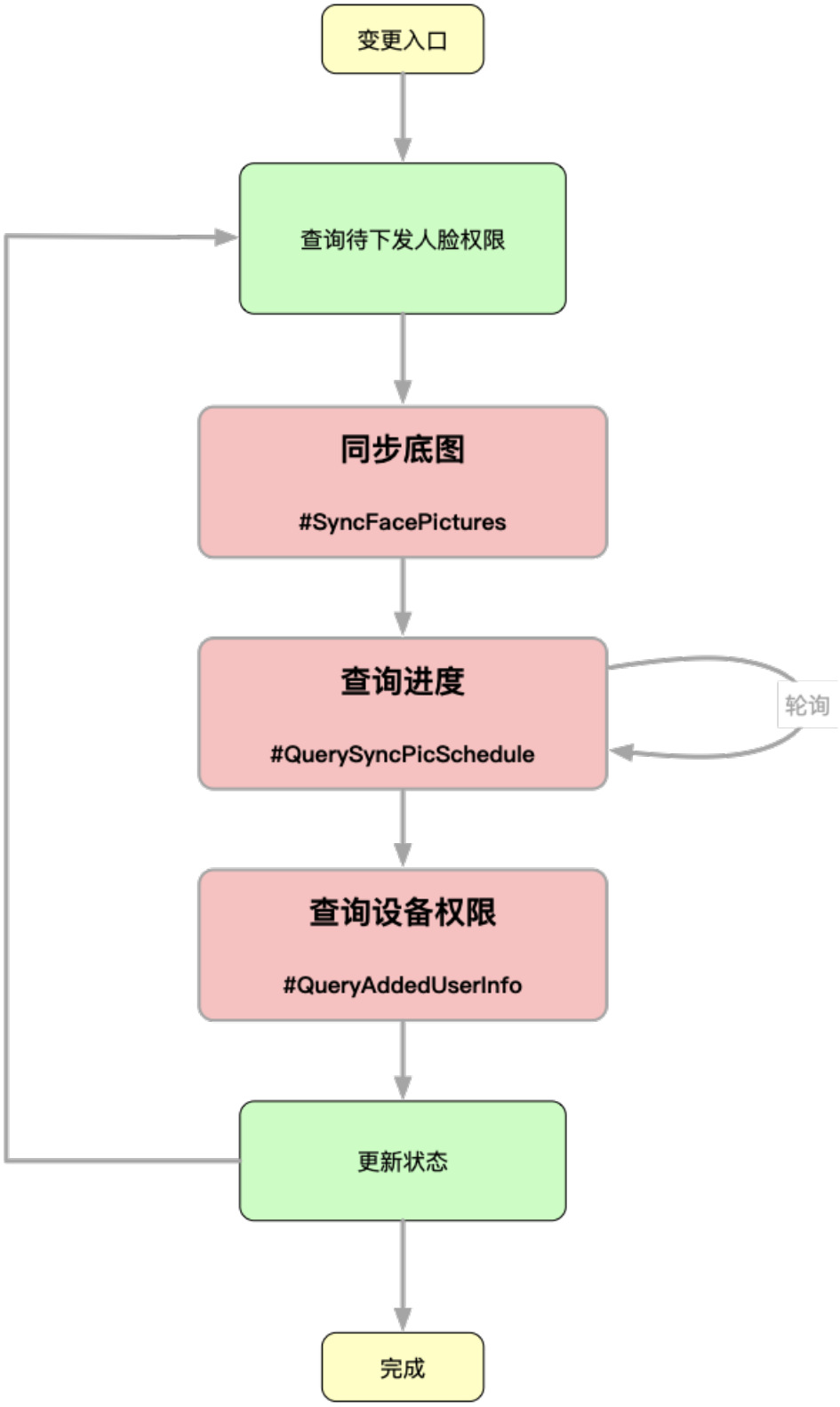
门禁服务SDK支持

- C/Linux：面向Linux系统设备，对接时需要厂商提供动态编译链，打包后分发完整SDK。
  - V1.x：支持C Link SDK 2.3
  - V4.x：支持C Link SDK 4.0（推荐）

- Android：面向移端版本应用系统

主要流程

人脸1.0主要下发流程



1.3.5.1. C SDK V1.x版开发指南

LinkFace SDK封装了设备与物联网云平台的通讯协议，实现了人员信息增删查，人脸信息增删查和人脸检测/识别事件上云等功能。SDK采用C语言开发，对外提供静态库和动态库版本。

版本历史

详见发布记录

SDK头文件

源码如下：

```
#ifndef __LINKFACE_H_V1__
#define __LINKFACE_H_V1__

#include <stdint.h>

#ifdef __cplusplus
extern "C"
{
#endif

/// 人员角色。
typedef enum AliPersonRole {
    /*! 未知。 */
    ALI_PERSON_ROLE_UNKNOWN = -2,

    /*! 所有角色。 */
    ALI_PERSON_ROLE_ALL = -1,

    /*! 普通人员。 */
    ALI_PERSON_ROLE_NORMAL = 0,

    /*! 白名单人员。 */
    ALI_PERSON_ROLE_WHITE = 1,

    /*! 黑名单人员。 */
    ALI_PERSON_ROLE_BLACK = 2,

    /*! 非普通人员。 */
    ALI_PERSON_ROLE_NOT_NORMAL = 3,

    /*! 非白名单人员。 */
    ALI_PERSON_ROLE_NOT_WHITE = 4,

    /*! 非黑名单人员。 */
    ALI_PERSON_ROLE_NOT_BLACK = 5
} AliPersonRole;

/*注册人脸需要的字段*/
struct AliPerson {
    /*! 人员ID。 */
    char id[64];
```

```
    /*! 人员姓名。 */
    char name[1024];

    /*! 人员角色。 */
    enum AliPersonRole role;

    /*! 人脸图片数据大小。单位：字节。 */
    int faceimg_size;

    /*! 人脸图像数据。 */
    unsigned char *faceimg;

    /*! 人脸特征数据 */
    unsigned char **face_feature;

    /*! 人脸特征数据个数 */
    int face_feature_size;

    /*! 当wg_card_id为0时，使用此字段作为韦根门禁卡号 */
    int64_t long_card_id;

    /*! 身份证号码 */
    char id_card[36];

    /*! 有效期起始时间。从1970年1月1日0时0分0秒到截止时间的秒数（0表示不使用）。 */
    unsigned int term_start;

    /*! 有效期截止时间。从1970年1月1日0时0分0秒到截止时间的秒数（-1表示无限期，0表示永久失效）。 */
    unsigned int term;
};

/// 人脸比对状态。
enum AliMatchStatus {
    /*! 比对失败。 */
    ALI_MATCH_STATUS_FAILED = -1,

    /*! 未进行比对。 */
    ALI_MATCH_STATUS_NULL = 0,

    /*! 比对成功。 */
    ALI_MATCH_STATUS_SUCCESS = 1,
};

/// 图片格式。
enum AliPictureFormat {
    /*! 未知 */
    ALI_PICTURE_FORMAT_NULL = -1,

    /*! Jpg。 */
    ALI_PICTURE_FORMAT_JPG = 0,
};

/// 性别。
```

```
enum AliSex {
    /*! 未知 */
    ALI_SEX_NULL = -1,

    /*! 男。 */
    ALI_SEX_MALE = 0,

    /*! 女。 */
    ALI_SEX_FEMALE = 1,
};

/*人脸对比上报*/
/*注册人脸需要的字段*/
struct AliMatcheResult {
    /*! 点位编号。 */
    char addr_id[32];

    /*! 点位名称。 */
    char addr_name[96];

    /*! 设备编号。 */
    char device_id[32];

    /*! 抓拍时间（毫秒）。 */
    uint64_t time_msec;

    /*! 实时抓拍标志，历史识别事件传递参数0。[0:非实时抓拍][1:实时抓拍] */
    int is_realtime;

    /*! 比对状态。 必要字段。*/
    enum AliMatchStatus match_status; //对比成功才会有人脸(即person_id，person_name，person_role，match_score)的值才有意义

    /*! 人脸比对到的人员ID。 抓拍成功时，必要字段。*/
    char person_id[64];

    /*! 人脸比对到的人员姓名。 抓拍成功时，必要字段。*/
    char person_name[255];

    /*! 人员角色。 抓拍成功时，必要字段。*/
    int person_role;

    /*! 对比相似度。 抓拍成功时，必要字段。*/
    int match_score;

    /*! 全景图大小。 */
    int panorama_img_size; //大小大于0时全景图的数据才有意义，否则全景图所有数据全部置空

    /*! 全景图格式。 */
    enum AliPictureFormat panorama_format;

    /*! 人脸位于全景图像的X坐标。 */
    unsigned short panorama_face_x;
```

```
    /*! 人脸位于全景图像的Y坐标。 */
    unsigned short panorama_face_y;

    /*! 人脸位于全景图像的宽度。 */
    unsigned short panorama_face_w;

    /*! 人脸位于全景图像的高度。 */
    unsigned short panorama_face_h;

    /*! 全景图数据。必要字段。 */
    char *panorama_img;

    /*! 全景图名称。 必要字段。 */
    char *panorama_img_name;

    /*! 特写图大小。 */
    int closeup_size; //大小大于0时特写图的数据才有意义，否则特写图所有数据全部置空

    /*! 特写图格式。 */
    enum AliPictureFormat closeup_format;

    /*! 人脸位于特写图像的X坐标。 */
    unsigned short closeup_face_x;

    /*! 人脸位于特写图像的Y坐标。 */
    unsigned short closeup_face_y;

    /*! 人脸位于特写图像的宽度。 */
    unsigned short closeup_face_w;

    /*! 人脸位于特写图像的高度。 */
    unsigned short closeup_face_h;

    /*! 特写图数据。 */
    char *closeup_img;

    /*! 性别。 */
    enum AliSex sex;

    /*! 年龄。 */
    unsigned char age;

    /*! 是否佩戴安全帽 */
    unsigned char hat;

    /*! 安全帽颜色 */
    unsigned char hat_color;

    /*! 人脸扭转角度。 */
    unsigned char turn_angle;

    /*! 人脸平面旋转角度。 */
    unsigned char rotate_angle;
```

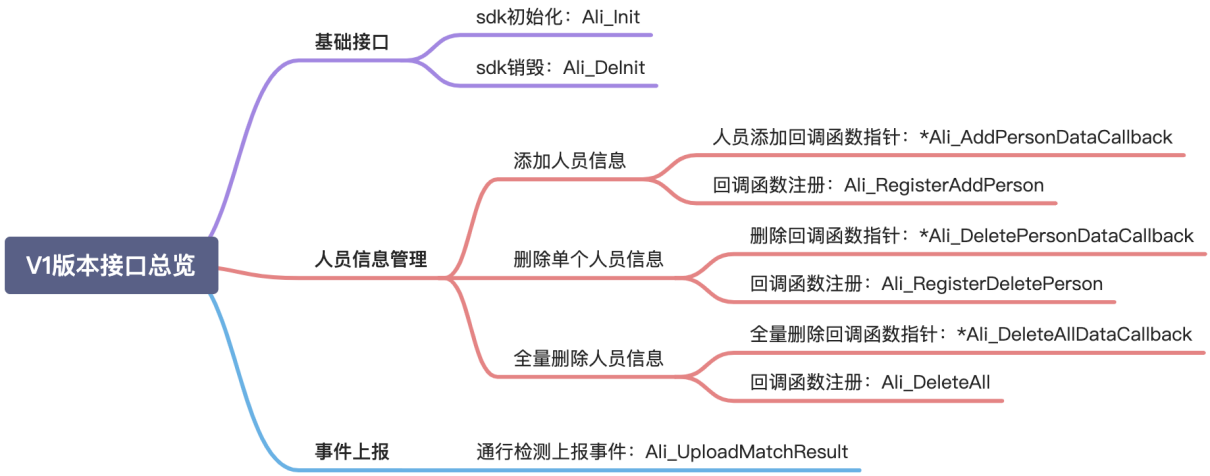
```
    /* 备用字段*/
    unsigned char resv[3];

    /*! 注册图大小。 */
    int register_size;

    /*! 注册图格式。 */
    enum AliPictureFormat register_format;

    /*! 注册图数据。 */
    char *register_img;

    /*! 设备序列号，唯一标识符。 */
    char device_sn[32];

    /*! 身份证信息标志。 */
    int idcard_mark; //值为1才表示身份证信息相关的数据才有意义

    /*! 号码。 */
    char idcard_number[36];

    /*! 姓名。 */
    char idcard_name[64];

    /*! 出生日期。 */
    char idcard_data[18];

    /*! 性别。 */
    enum AliSex idcard_sex;

    /*! 民族。 */
    char idcard_nation[20];

    /*! 住址。 */
    char idcard_address[120];

    /*! 签发机关。 */
    char idcard_issue[44];

    /*! 有效期起始时间。 */
    char idcard_validityterm_start[18];

    /*! 有效期截止时间。 */
    char idcard_validityterm_end[18];

    /*! 韦根门禁卡号 */
    uint64_t long_card_id;
};

/* SDK日志等级 */
enum LinkfaceLogLevel {
    /*! debug level */
    ALI_LF_LOG_DEBUG = 0,

    /*! info level */
```

```
    /*! info level */
    ALI_LF_LOG_INFO = 1,

    /*! warn level */
    ALI_LF_LOG_WARN = 2,

    /*! error level */
    ALI_LF_LOG_ERROR = 3,
};

/* SDK日志输出类型 */
enum LinkfaceLogDest {
    /*! 标准输出 */
    ALI_LF_STDOUT = 0,

    /*! 文件 */
    ALI_LF_FILE = 1,
};

/* 日志配置 */
struct LinkfaceLogCfg {

    /* 日志等级 */
    enum LinkfaceLogLevel lvl;

    /* 日志输出类型 */
    enum LinkfaceLogDest dst;

    /*! 日志文件保存路径 */
    char *logpath;

    /*! 日志存储目录大小Mb */
    int max_size_log_mb;
};

/* sdk配置信息 */
struct LinkfaceConfig {

    /*! productKey: 三元组之一，详见物联网平台官方文档. */
    char *productKey;

    /*! productSecret: 产品秘钥，一型一密时使用，否者填""， 详见物联网平台官方文档. */
    char *productSecret;

    /*! deviceName: 三元组之一，详见物联网平台官方文档. */
    char *deviceName;

    /*! deviceSecret: 三元组之一，详见物联网平台官方文档. */
    char *deviceSecret;

    /*! filePath: sdk会本地缓存一些配置信息，厂商需要确保此目录已存在且可写，预计占用物理空间2M以内。
*/
    char *filePath;

    /*! 日志配置 */
```

```
    struct LinkfaceLogCfg cfg_log;

};

/** 云端下发底库同步请求时，sdk会调用该接口将人脸底图信息逐个传递给厂商，厂商需要将该人脸添加到数据库中以
 * 进行人脸识别。
 * @brief   注册人脸数据接收回调函数格式定义。
 * @param   注册人脸结构体。
 * @param   user 用户参数，用户设置回调时指定的参数通过该参数再次回传给用户。
 */
typedef int (*Ali_AddPersonDataCallback)(const struct AliPerson *person, void *user);

/** 云端需要删除某个人脸底库时，sdk会调用该接口将人脸信息逐个传递给厂商，厂商需要将该人脸从数据库中删除，
 * 下次人脸检测时，该用户会因底库不存在该用户信息导致识别识别。
 * @brief   删除人脸数据接收回调函数格式定义。
 * @param   人脸id。
 * @param   user 用户参数，用户设置回调时指定的参数通过该参数再次回传给用户。
 */
typedef int (*Ali_DeletePersonDataCallback)(const char *person_id, void *user);

/** sdk调用本接口后，厂商需要清空人脸数据库。
 * @brief   全量删除人脸回调函数。
 * @param   用户数据。
 * @param   user 用户参数，用户设置回调时指定的参数通过该参数再次回传给用户。
 */
typedef int (*Ali_DeleteAllDataCallback)(void *user);

/** 注册添加人脸回调接口。
 * @param   callback 厂商需要实现的函数。
 * @param   user 用户参数，用户设置回调时指定的参数通过该参数再次回传给用户。
 */
void Ali_RegisterAddPerson(Ali_AddPersonDataCallback callback, void *user);

/** 注册删除人脸回调接口。
 * @param   callback 厂商需要实现的函数。
 * @param   user 用户参数，用户设置回调时指定的参数通过该参数再次回传给用户。
 */
void Ali_RegisterDeletePerson(Ali_DeletePersonDataCallback callback, void *user);

/** 注册全量删除人脸回调接口。
 * @param   callback 厂商需要实现的函数。
 * @param   user 用户参数，用户设置回调时指定的参数通过该参数再次回传给用户。
 */
void Ali_DeleteAll(Ali_DeleteAllDataCallback callback, void *user);

/**  当人脸门禁机检测到人脸后，无论是否能正确识别出该人脸，均需要调用本接口将识别结果上报到云端。
 * @param   result 识别结果，如果识别成功，需要包含已识别的人的基本信息。
 *
 * @return 成功返回0，其他值均为失败。
 */
int Ali_UploadMatchResult(const struct AliMatcheResult *result);
```

```
/**  init 接口，必须首先调用，如果初始化失败，请sleep 1s后重试，直到成功。
 * @param   cfg: sdk配置信息
 * 成功返回1，失败返回-1。
 * */
int Ali_Init(struct LinkfaceConfig *cfg);

/** 进程退出前，释放sdk资源。
 */
void Ali_DeInit();

#ifdef __cplusplus
}
#endif

#endif
```

接口总览



SDK包内容介绍

```
tree -L 4
.
├── CMakeLists.txt                          ---------演示如何链接linkface和linkkit的库，如何编译sample
├── LinkFace                                ---------linkface的库和使用的样例
│   ├── samples                             ---------linkface使用的样例
│   │   ├── v1                              ---------v1版本使用的样例
│   │   │   ├── 1.jpg                       ---------模拟人脸抓拍事件上云图片
│   │   │   ├── link_face_demo_v1           ---------v1版本演示可执行文件
│   │   │   ├── linkface_demo_v1.c          ---------v1版本sample
│   │   │   ├── linkkit_adapter_v1.c        ---------v1版本自定义集成linkkit的上云能力
│   │   │   ├── log_util.c
│   │   │   └── log_util.h
│   │   └── v2                              ---------v2版本使用的样例
│   │       ├── 1.jpg
│   │       ├── database.c                  ---------v2版本基于sqlite完成了人脸底库的本地存储
│   │       ├── database.h
```

```
│   │       ├── link_face_demo_v2              ---------v2版本演示可执行文件
│   │       ├── linkface_demo_v2.c             ---------v2版本sample
│   │       ├── linkkit_adapter_v2.c           ---------v2版本自定义集成linkkit的上云能力
│   │       └── list.h
│   └── sdk
│       ├── v1
│       │   ├── liblinkface_v1.a               ----------v1版本对外提供的动态库
│       │   ├── liblinkface_v1.so              ----------v1版本对外提供的静态库
│       │   └── linkface_sdk_v1.h              ----------v1版本对外提供的头文件
│       └── v2
│           ├── alarm_event_post_task.h
│           ├── event_cache_manager.h
│           ├── face_add_task.h
│           ├── facebase_manager.h
│           ├── face_cache_manager.h
│           ├── face_download_task.h
│           ├── liblinkface_v2.a               ----------v2版本对外提供的动态库
│           ├── liblinkface_v2.so              ----------v2版本对外提供的静态库
│           ├── linkface_sdk_v2.h              ----------v2版本对外提供的头文件
│           └── net_protocol.h
└── third_party
    ├── cJSON                                  ----------依赖的cjson库
    │   ├── cJSON.h
    │   ├── libcjson.a
    │   ├── libcjson.so
    │   ├── libcjson.so.1
    │   └── libcjson.so.1.7.7
    ├── cJSON-1.7.7.tar.gz
    ├── linkkit-sdk-c                          ----------依赖的linkkit库
    │   ├── exports
    │   │   ├── iot_export_awss.h
    │   │   ├── iot_export_coap.h
    │   │   ├── iot_export_compat.h
    │   │   ├── iot_export_errno.h
    │   │   ├── iot_export_event.h
    │   │   ├── iot_export_http2.h
    │   │   ├── iot_export_http2_stream.h
    │   │   ├── iot_export_http.h
    │   │   ├── iot_export_linkkit.h
    │   │   ├── iot_export_mqtt.h
    │   │   ├── iot_export_ota.h
    │   │   ├── iot_export_shadow.h
    │   │   ├── iot_export_state.h
    │   │   ├── linkkit_export.h
    │   │   └── linkkit_gateway_export.h
    │   ├── imports
    │   │   ├── iot_import_awss.h
    │   │   ├── iot_import_config.h
    │   │   ├── iot_import_crypt.h
    │   │   ├── iot_import_dtls.h
    │   │   ├── iot_import_ota.h
    │   │   ├── iot_import_product.h
    │   │   ├── iot_import_tcp.h
    │   │   ├── iot_import_tls.h
```

```
    │   │   └── iot_import_udp.h
    │   ├── iot_export.h
    │   ├── iot_import.h
    │   ├── libiot_hal.a
    │   ├── libiot_sdk.a
    │   └── libiot_tls.a
    ├── linkkit-sdk-c.tar.gz
    ├── release
    │   ├── apr
    │   │   ├── bin
    │   │   ├── build-1
    │   │   ├── include
    │   │   └── lib
    │   ├── apr_util
    │   │   ├── bin
    │   │   ├── include
    │   │   └── lib
    │   ├── curl
    │   │   ├── bin
    │   │   ├── include
    │   │   ├── lib
    │   │   └── share
    │   ├── expat
    │   │   ├── bin
    │   │   ├── include
    │   │   ├── lib
    │   │   └── share
    │   ├── iconv
    │   │   ├── bin
    │   │   ├── include
    │   │   ├── lib
    │   │   └── share
    │   ├── mxml
    │   │   ├── include
    │   │   ├── lib
    │   │   └── share
    │   ├── openssl
    │   │   ├── bin
    │   │   ├── include
    │   │   └── lib
    │   ├── oss_c_sdk
    │   │   ├── include
    │   │   └── lib
    │   └── zlib
    │       ├── include
    │       ├── lib
    │       └── share
    ├── sqlite                                  ----------依赖的sqlite库
    │   ├── libsqlite3.a
    │   ├── libsqlite3.so
    │   └── sqlite3.h
    └── sqlite-amalgamation-3310100.tar.gz
```

快速使用指南

下面来基于Ubuntu 16.04的环境来描述如何运行官方Sample。

- 解压

```
tar zxvf link_face_xxx_.tar.gz
```

- 编译

我们提供了CMakeLists.txt来演示如何使用sdk，若单独使用sdk，请参考CMakeLists.txt来完成库的链接。

```
cd link_face_xxx_

mkdir build

cd build

cmake ../

make
```

- 运行

```
cp link_face_demo_v1 ../LinkFace/samples/v1
./link_face_demo_v1 ${Product_Key} ${Device_Name} ${Device_Secret} ./
${Product_Key} ${Device_Name} ${Device_Secret} 分别替换成在物联网平台上申请的三元组即可。
```

1.3.5.2. C SDK V4.x版开发指南

LinkGuard C SDK封装了设备与物联网云平台的通讯协议，实现了人员信息增删查和人员通行检测/识别事件上云等功能。SDK采用C语言开发，对外提供静态库和动态库版本。

版本历史

详见发布记录

SDK头文件

具体源码如下：

```
#ifndef __LINKGUARD_SDK__
#define __LINKGUARD_SDK__

#include <stdio.h>
#include <stdint.h>

#ifdef __cplusplus
extern "C"
{
#endif

#define LG_API_PUBLIC __attribute__((visibility ("default")))
```

```
/**
 * 用户角色枚举
 */
typedef enum iotx_user_role_t {
    /* 未知 */
    USER_ROLE_UNKNOWN = -2,

    /* 所有角色 */
    USER_ROLE_ALL = -1,

    /* 普通用户 */
    USER_ROLE_NORMAL = 0,

    /* 白名单用户 */
    USER_ROLE_WHITE = 1,

    /* 黑名单用户 */
    USER_ROLE_BLACK = 2,

    /* 非普通用户 */
    USER_ROLE_NOT_NORMAL = 3,

    /* 非白名单用户 */
    USER_ROLE_NOT_WHITE = 4,

    /* 非黑名单用户 */
    USER_ROLE_NOT_BLACK = 5,
} iotx_user_role_t;

/**
 * 通行比对状态枚举
 */
typedef enum iotx_match_status_t {
    /* 比对失败 */
    MATCH_STATUS_FAILED = -1,

    /* 未进行比对 */
    MATCH_STATUS_NULL = 0,

    /* 比对成功 */
    MATCH_STATUS_SUCCESS = 1,
} iotx_match_status_t;

/**
 * 图片格式枚举
 */
typedef enum iotx_picture_format_t {
    /* 未知 */
    PICTURE_FORMAT_NULL = -1,

    /* JPG */
    PICTURE_FORMAT_JPG = 0,
} iotx_picture_format_t;
```

```
/**
 * SDK日志级别
 */
typedef enum iotx_log_level_t {
    /* debug level */
    LF_LOG_DEBUG = 0,

    /* info level */
    LF_LOG_INFO = 1,

    /* warn level */
    LF_LOG_WARN = 2,

    /* error level */
    LF_LOG_ERROR = 3,
} iotx_log_level_t;

/**
 * SDK日志输出类型
 */
typedef enum iotx_log_dest_t {
    /* 标准输出 */
    LF_STDOUT = 0,

    /* 文件 */
    LF_FILE = 1,
} iotx_log_dest_t;

/**
 * 注册用户需要的信息
 */
struct iotx_user_info_t {

    /* 用户ID */
    char id[64];

    /* 用户姓名 */
    char name[1024];

    /* 用户角色 */
    enum iotx_user_role_t role;

    /* 门禁身份认证图片数据大小 */
    int certified_pic_size;

    /* 门禁身份认证图片图像数据 */
    unsigned char *certified_pic;

    /* 图片特征数据 */
    unsigned char **pic_feature;

    /* 图片特征数据个数 */
    int pic_feature_size;
};
```

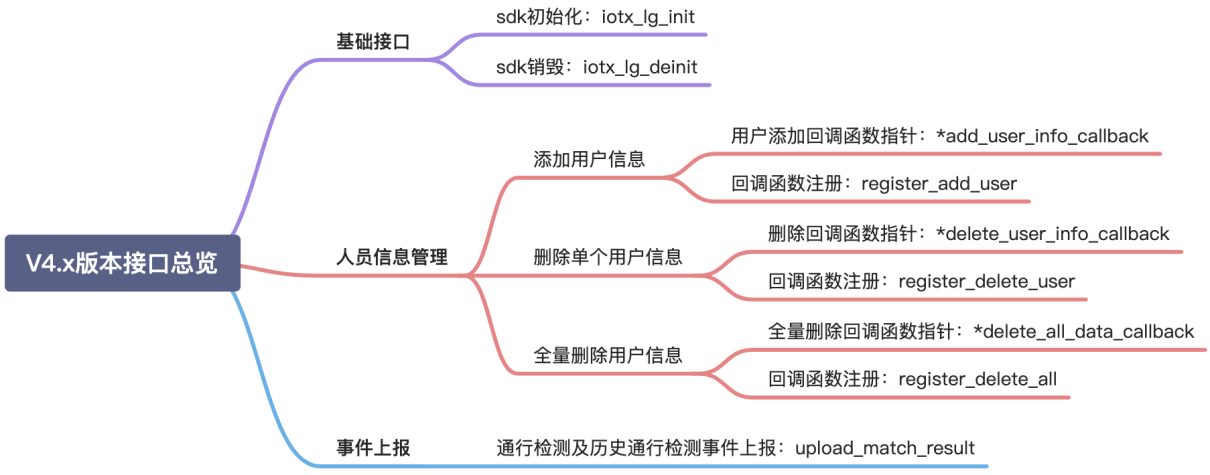
```
};

/**
 * 通行上报信息
 */
struct iotx_matche_result_t {

    /* 设备编号 */
    char device_id[32];

    /* 抓拍时间（毫秒） */
    uint64_t time_msec;

    /* 实时抓拍标志，历史识别事件传递参数0。[0:非实时抓拍][1:实时抓拍] */
    int is_realtime;

    /* 比对状态。必要字段。 */
    enum iotx_match_status_t match_status;

    /* 比对成功的用户ID。抓拍成功时，必填字段。 */
    char user_id[64];

    /* 比对成功的用户姓名。抓拍成功时，必填字段。 */
    char user_name[255];

    /* 比对成功的用户角色。抓拍成功时，必填字段。 */
    int user_role;

    /* 对比相似度，抓拍成功时，必填字段。 */
    int match_score;

    /* 全景图大小 */
    size_t panorama_pic_size;

    /* 全景图格式 */
    enum iotx_picture_format_t panorama_format;

    /* 全景图数据。必填字段。 */
    char *panorama_pic;

    /* 全景图名称。必填字段。 */
    char *panorama_pic_name;

};

/**
 * SDK日志配置
 */
struct iotx_log_cfg_t {

    /* 日志等级 */
    enum iotx_log_level_t lvl;

    /* 日志输出类型 */
    enum iotx_log_dest_t dst;
```

```
    /* 日志文件保存路径 */
    char *log_path;

    /* 日志存储目录大小为MB */
    int max_size_log_mb;
};

/**
 * SDK配置信息
 */
struct iotx_lg_config_t {

    /* 用公共实例，该参数要设置为1。若用独享实例，要将该参数设置为0。*/
    int8_t public_instance;

    /* 阿里云平台上海站点的域名后缀。如果是企业实例，要改成企业实例的接入点。*/
    char *mqtt_url;

    /* 阿里云平台上海站点的http2域名后缀 */
    char *http2_url;

    /* product_key：三元组之一，详见物联网平台官方文档。*/
    char *product_key;

    /* product_secret：产品秘钥，一型一密时使用，否者填""，详见物联网平台官方文档。*/
    char *product_secret;

    /* device_name：三元组之一，详见物联网平台官方文档。*/
    char *device_name;

    /* device_secret：三元组之一，详见物联网平台官方文档。*/
    char *device_secret;

    /* data_path: SDK会本地缓存配置信息，厂商需要确保此目录存在且可写，预计占用物理空间2M以内。*/
    char *data_path;

    /* 更新数据模式。[0:重复不更新][1:强制更新] */
    int update_mode;

    /* 日志配置 */
    struct iotx_log_cfg_t cfg_log;

    /* 外部初始化LinkSDK，需要把mqtt_handle赋值给LinkGuard SDK。 */
    void *mqtt_handle;

    /* 外部初始化LinkSDK，需要把dm_handle赋值给LinkGuard SDK。 */
    void *dm_handle;

};

/**
 * 获取不同环境mqtt host url
 * @param cfg SDK配置信息
```

```
 */
static inline void mqtt_host(const struct iotx_lg_config_t *cfg, char *host) {
#ifdef ON_PRE
    snprintf(host, 100, "%s", "iot-test-pre-ha.iot-as-mqtt.unify.aliyuncs.com");
#endif
#ifdef ON_ONLINE
    if (1 == cfg->public_instance) {
        snprintf(host, 100, "%s.%s", cfg->product_key, cfg->mqtt_url);
    } else {
        snprintf(host, 100, "%s", cfg->mqtt_url);
    }
#endif
}

/**
 * 获取不同环境http2 host url
 * @param cfg SDK配置信息
 */
static inline void http2_host(const struct iotx_lg_config_t *cfg, char *host, uint16_t *por
t) {
#ifdef ON_PRE
    snprintf(host, 100, "%s", "121.196.243.216");
    *port = 8443;
#endif
#ifdef ON_ONLINE
    snprintf(host, 100, "%s.%s", cfg->product_key, cfg->http2_url);
#endif
}

/**
 * SDK初始化接口，需首先调用，如果初始化失败，请sleep 1s后重试，直到成功。
 * @param cfg SDK配置信息
 * @return [1:成功][-1:失败][-2:参数错误]
 */
LG_API_PUBLIC int iotx_lg_init(struct iotx_lg_config_t *cfg);

/**
 * 进程退出时，可释放SDK资源。
 */
LG_API_PUBLIC void iotx_lg_deinit();

/**
* 云端添加用户信息请求时，SDK调用回调将用户信息逐个传递给厂商，厂商需要将该用户添加到数据库中进行身份识
别。
* @param user_info 用户信息
*/
typedef int (*add_user_info_callback)(const struct iotx_user_info_t *user_info);

/**
* 云端删除用户信息请求时，SDK调用回调接口将用户信息逐个传递给厂商，厂商需要将用户从数据库中删除。
* @param user_id 删除用户信息ID
*/
typedef int (*delete_user_info_callback)(const char *user_id);
```

```
/**
* SDK调用回调接口，厂商需要清空用户数据库。
*/
typedef int (*delete_all_data_callback)();

/**
 * 注册用户信息添加回调接口
 * @param callback 添加回调接口
 */
LG_API_PUBLIC void register_add_user(add_user_info_callback callback);

/**
* 注册用户信息删除回调接口
* @param callback 删除回调接口
*/
LG_API_PUBLIC void register_delete_user(delete_user_info_callback callback);

/**
* 注册用户信息全量删除回调接口
* @param callback 全量删除回调接口
*/
LG_API_PUBLIC void register_delete_all(delete_all_data_callback callback);

/**
 * 当门禁机检测到用户后，无论是否能正确识别出该用户，均需要调用本接口将识别结果上报到云端。如果断网续传的历史识别事件，需要设置is_realtime为0。
 * @param result 识别结果，如果识别成功，需要包含已识别的人的基本信息。
 * @return 上报结果[0:成功][-1:失败]
 */
LG_API_PUBLIC int upload_match_result(const struct iotx_matche_result_t *result);

#ifdef __cplusplus
}
#endif

#endif
```

接口总览



SDK包内容介绍

```
tree -L 3
.
├── CMakeLists.txt                ---------演示如何链接LinkGuard SDK和Link SDK的库，如何编译samp
le
├── LinkGuard
│   ├── samples                   ---------LinkGuard的库和使用的样例
│   │   ├── aiot.jpg              ---------模拟人脸抓拍事件上云图片
│   │   ├── linkguard_demo        ---------演示可执行文件
│   │   ├── linkguard_demo.c      ---------sample代码
│   │   └── linksdk_adapter.c     ---------自定义集成Link SDK的上云能力
│   └── sdk
│       ├── common                ----------基础库头文件
│       ├── liblinkguard.a        ----------LinkGuard动态库
│       ├── liblinkguard.so       ----------LinkGuard静态库
│       ├── linkguard_sdk.h       ----------LinkGuard头文件
│       ├── portfiles             ----------底层软硬件资源接口
│       └── version.h             ----------版本号信息
└── third_party
    └── release
        ├── LinkSDK-v4x           ----------依赖LinkSDK v4x头文件
        └── cJSON               ----------依赖cJSON头文件
```

快速使用指南

下面来基于Ubuntu 16.04的环境来描述如何运行官方Sample。

- 解压

```
tar zxvf linkguard_xxx_xxx.tar.gz
```

- 编译

我们提供了CMakeLists.txt来演示如何使用sdk，若单独使用sdk，请参考CMakeLists.txt来完成库的链接。

```
cd linkguard_xxx_xxx

mkdir build

cd build

cmake ../

make
```

- 运行

```
./linkguard_demo ${Product_Key} ${Device_Name} ${Device_Secret} /data/path /log/path

${Product_Key} ${Device_Name} ${Device_Secret} 分别替换成在物联网平台上申请的三元组即可。
```

1.3.5.3. Android SDK v1.1接口说明

LinkFace SDK封装了设备与云的通讯协议，实现了人脸信息增删查和人脸检测/识别事件上云等功能。SDK支持Android Studio开发环境。

SDK集成

LinkFace v1.1需要通过LinkKit SDK与服务端完成物模型相关通信，支持以下两种方式集成LinkKit SDK:

1) App独立集成LinkKit SDK能力，建议在LinkKit的onInitDone回调中进行LinkFace SDK初始化。

LinkFace同时会使用LinkKit创建的Http2通道进行文件传输，需要LinkKit初始化时支持Http2参数配置，完整示例如下:

```
    private void initLinkKit() {
        LinkKitInitParams params = new LinkKitInitParams();

        DeviceInfo deviceInfo = new DeviceInfo();
        deviceInfo.productKey = productKey;
        deviceInfo.deviceName = deviceName;
        deviceInfo.deviceSecret = deviceSecret;
        params.deviceInfo = deviceInfo;

        Map<String, ValueWrapper> propertyValues = new HashMap<>();
        params.propertyValues = propertyValues;

        IoTApiClientConfig userData = new IoTApiClientConfig();
        params.connectConfig = userData;

        IoTMqttClientConfig mqttClientConfig = new IoTMqttClientConfig(deviceInfo.productKe
y,
                deviceInfo.deviceName, deviceInfo.deviceSecret);
        mqttClientConfig.receiveOfflineMsg = true;
        mqttClientConfig.isCheckChannelRootCrt = true;
        params.mqttClientConfig = mqttClientConfig;

        // Http2通道配置
        IoTH2Config ioTH2Config = new IoTH2Config();
        ioTH2Config.clientId = "client-id";
        ioTH2Config.endPoint = "https://" + deviceInfo.productKey + ioTH2Config.endPoint;
        params.iotH2InitParams = ioTH2Config;
        Id2ItlsSdk.init(this);

        IoTDMConfig ioTDMConfig = new IoTDMConfig();
        ioTDMConfig.enableLocalCommunication = false;
        // 是否启用物模型功能，如果不开启，本地通信功能也不支持
        ioTDMConfig.enableThingModel = true;
        ioTDMConfig.enableGateway = false;
        ioTDMConfig.enableLogPush = false;
        params.ioTDMConfig = ioTDMConfig;

        LinkKit.getInstance().init(this, params, linkkitConnectListener);
    }
```

2) **注意**：如果根据LinkKit demo由App自己实现对接，建议只处理除LinkFace外的明确的物模型服务调用，其他服务调用不要发送响应（demo代码默认会给所有服务调用发送空数据响应），否则可能导致LinkFace的服务响应无法被物联网平台收到，功能无法正常使用。

3) LinkFace SDK自动初始化LinkKit，通过设置init函数的needInitLinkKit参数为true实现。App如要处理自有物模型消息，则需要对接ILinkKitListener接口，参见接口说明部分。

依赖引入

本SDK以jar包形式提供，需要拷贝到项目的libs目录中，在build.gradle中添加相关引用。

```
// 在App build.gradle的dependencies中添加依赖
    implementation files('libs/LinkFace-release-1.1.jar')
```

混淆配置

```
-keep class com.aliyun.iotx.linkface.sdk.LinkFaceErrorCode { public *; }
-keep class com.aliyun.iotx.linkface.sdk.LinkFaceImageInfo { public *; }
-keep class com.aliyun.iotx.linkface.sdk.LinkFaceMatchResult { public *; }
-keep class com.aliyun.iotx.linkface.sdk.ILinkFaceCallback { *; }
-keep class com.aliyun.iotx.linkface.sdk.ILinkKitListener { *; }
-keep class com.aliyun.iotx.linkface.sdk.DeviceInfoEx { public *; }
-keep class com.aliyun.iotx.linkface.sdk.LinkFace { public *; }

-keep class com.aliyun.iotx.linkface.manager.FaceTsl { *; }
-keep class com.aliyun.iotx.linkface.manager.FaceDoorProperty { *; }
-keep class com.aliyun.iotx.linkface.manager.FaceGroupDTO { *; }
-keep class com.aliyun.iotx.linkface.manager.FaceGroupDTO$* { *; }
```

权限说明

```
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.INTERNET" />
```

SDK jar包

LinkFace-release-1.1.jar(43 kB)

接口说明

LinkFace通过单例方式进行接口调用，通过如下方式获取实例，再调用相关接口

```
LinkFace.getInstance()
```

1. 初始化

完成LinkKit接入和人脸信息管理准备工作。

```
void init(Context context, DeviceInfoEx deviceInfo, boolean needInitLinkKit, ILinkFaceCallback linkfaceCallback)
```

参数说明：

参数	类型	说明
context	Context	应用上下文，用于调用上下文环境相关的系统接口
deviceInfo	DeviceInfoEx	设备信息，用于LinkKit接入及初始属性上报，参见DeviceInfoEx定义
needInitLinkKit	boolean	指定SDK是否需要初始化LinkKit，若为false，则需要调用方提前完成LinkKit初始化，并在LinkKit的onInitDone回调中调用LinkFace的初始化接口。已接入LinkKit SDK的请直接设置为false。
linkfaceCallback	ILinkFaceCallback	LinkFace回调接口，需要调用方实现，并按要求完成相应人脸数据处理，参见LinkFace回调接口处理部分

DeviceInfoEx类定义：

```
public class DeviceInfoEx {
    // productKey、deviceName、deviceSecret为三元组必传信息
    public String productKey = null;
    public String deviceName = null;

    public String productSecret = null;
    public String deviceSecret = null;

    // 是否支持人脸特征、算法版本、固件版本必传信息
    public boolean supportFaceFeature = false;
    public String algorithmVersion = "";
    public String firewareVersion = "";

    public DeviceInfoEx() {
    }
}
```

示例代码：

```
    @Override
    public void onInitDone(Object o) {
        Log.d(TAG, "onInitDone");
        DeviceInfoEx deviceInfo = new DeviceInfoEx();
        deviceInfo.productKey = EnvConfig.CURRENT_DEV.productKey;
        deviceInfo.deviceName = EnvConfig.CURRENT_DEV.deviceName;
        deviceInfo.deviceSecret = EnvConfig.CURRENT_DEV.deviceSecret;
        deviceInfo.supportFaceFeature = true;
        deviceInfo.algoVersion = "1.0";
        deviceInfo.firewareVersion = "firmware-1.0";
        LinkFace.getInstance().init(this, deviceInfo, false,this);
    }
```

2. 上报人脸检测结果

无论人脸比对是否成功，均需要调用本接口上报检测结果，比对失败时，person id可以填空.

```
    /**
     * 当人脸门禁机检测到人脸后，无论是否能正确识别出该人脸，均需要调用本接口将识别结果上报到云端。
     *
     * @param matchResult 识别结果，如果识别成功，需要包含已识别的人的基本信息。
     * @return 成功返回0，失败返回-1
     * <p>
     * 注意：
     * 1. 该接口为同步接口，请勿并发调用。
     * 2. 上报结果需要一定耗时，建议不要在UI线程中调用本接口。
     * 3. 若接口返回失败，建议最多重试3次。
     */
    public int uploadMatchResult(LinkFaceMatchResult matchResult) {
```

参数说明：

参数	类型	说明
matchResult	LinkFaceMatchResult	人脸比对结果，参见LinkFaceMatchResult定义

LinkFaceMatchResult定义：

```
public class LinkFaceMatchResult {

    /**
     * 比对失败
     */
    public static final int LINKFACE_MATCH_STATUS_FAILED = -1;

    /**
     * 未进行比对
     */
    public static final int LINKFACE_MATCH_STATUS_NULL = 0;
```

```
    /**
     * 比对成功
     */
    public static final int LINKFACE_MATCH_STATUS_SUCCESS = 1;

    /**
     * 比对状态<br>
     * 对比成功才会有人脸(即person_id, match_score)的值才有意义
     */
    public int linkFaceMatchStatus;

    /**
     * 人脸比对到的人员ID, 抓拍成功时, 必要字段<br>
     * 长度不超过32个字符
     */
    public String personId;

    /**
     * 对比相似度, 抓拍成功时，必要字段, 范围为[1, 100]
     */
    public int matchScore;

    /**
     * 人脸全景图原始数据<br>
     * 不能超过10MByte<br>
     * 大小大于0时全景图的数据才有意义，否则全景图所有数据全部置空
     */
    public byte[] panoramaImgRawData;

    /**
     * 人脸全景图名称
     */
    public String panoramaImgName;

    /**
     * 比对事件发生时的时间戳, 单位ms
     */
    public long timestampMs;

    public LinkFaceMatchResult() {
    }
}
```

返回值：

类型	说明
int	成功返回0，失败返回-1。该接口为同步接口，请勿并发调用，失败时建议最多重试3次。

示例代码：

```
    public void uploadMatchResultClicked(View view) {
        LinkFaceMatchResult linkFaceMatchResult = new LinkFaceMatchResult();
        linkFaceMatchResult.linkFaceMatchStatus = LinkFaceMatchResult.LINKFACE_MATCH_STATUS
_SUCCESS;
        linkFaceMatchResult.personId = "007";
        linkFaceMatchResult.panoramaImgRawData = mockFacePictureData;
        linkFaceMatchResult.timestampMs = System.currentTimeMillis();
        linkFaceMatchResult.panoramaImgName = System.currentTimeMillis() + "_taylor-swift.p
ng";
        linkFaceMatchResult.matchScore = 78;
        int ret = LinkFace.getInstance().uploadMatchResult(linkFaceMatchResult);
        appendLog("上报人脸比对结果 ret="+ret);
    }
```

3. 注册/注销LinkKit监听者回调

若由LinkFace完成LinkKit初始化，调用方仍希望处理自有物模型属性或者服务调用，可注册监听者回调进行处理。已接入LinkKit SDK的无需关注。

```
void registerLinkKitListener(ILinkKitListener linkkitListener)
void unregisterLinkKitListener(ILinkKitListener linkkitListener)
```

参数说明：

参数	类型	说明
linkkitListener	ILinkKitListener	LinkKit监听者回调，参见ILinkKitListener定义

ILinkKitListener定义：

```
public interface ILinkKitListener {
    public static final int LKS_INIT_ERROR = -1;
    public static final int LKS_CONNECT_OFFLINE = 0;
    public static final int LKS_CONNECT_ONLINE = 1;

    void onInitDone();
    void onStatusChange(int status, Object data);

    void onSetProperty(JSONObject properties);

    // 执行完需要发送响应消息，可以调用LinkFace.getInstance().sendServiceResponse(int code, JSO
NObject data)
    void onInvokeService(String serviceId, String serviceIdentifier, JSONObject params);

    void onBroadcast(String identifier, JSONObject data);
}
```

4. 释放资源

退出时调用释放LinkFace使用的资源。

```
void destroy()
```

LinkFace回调接口处理

调用者需要实现ILinkFaceCallback接口来完成人脸相关操作处理

1.SDK初始化结果异步通知

```
    /**
     * SDK初始化结果
     * @param code
     */
    void onInit(int code);
```

2.添加人脸底库

```
    /**
     * 云端下发底库同步请求时，sdk先缓存人脸图片，再调用该接口将图片入库。
     *
     * @param imageInfo 注册人脸图片信息
     * @return 图片入库结果. 参考：{@link com.aliyun.iotx.linkface.sdk.LinkFaceErrorCode}
     * <p>
     * 注意:<br>
     * 1. 该接口为同步调用，接口不宜耗时过久<br>
     * 2. 该接口不会在多线程中并发调用<br>
     * 3. 错误码请严格按照 {@link com.aliyun.iotx.linkface.sdk.LinkFaceErrorCode} 中的定义<br>
     */
    int onAddImageInfo(LinkFaceImageInfo imageInfo);
```

3.删除人脸底库

```
    /**
     * 云端需要删除某个人脸底库时，sdk会调用该接口将人脸信息逐个传递给厂商，厂商需要将该人脸从数据库中删除
     * @param   imageInfo 待删除图片信息
     * @ret     如果删除人脸成功，则返回0，否者返回其他值. 参考：{@link com.aliyun.iotx.linkface.sdk.LinkFaceErrorCode}
     *
     * 注意:<br>
     * 1. 只有人脸数据真实存在时(personId/faceImageMd5)，才返回成功。<br>
     * 2. personId 不存在返回-1。<br>
     * 3. personId 存在而 faceImageMd5 不存在，返回-2。<br>
     */
    int onDeleteImageInfo(LinkFaceImageInfo imageInfo);
```

4.查询人脸底库总数

```
    /**
     * 查询端侧人脸底库总数
     *
     * @return 人脸底库总数
     */
    int onQueryImageCount();
```

5.清空所有人脸底库记录

```
    /**
     * 将算法特征库的内容完全清空并删除数据库记录
     * @return 如果清空成功，则返回0，否者返回具体错误码．参考：{@link com.aliyun.iotx.linkface.sdk.LinkFaceErrorCode}
     */
    int onClearAll();
```

1.3.5.4. 发布记录

版本说明

正式环境、预发环境版本为设备接入Link SDK不同环境区分版本。预发版本用于厂商给阿里同学提供测试固件使用。Link SDK适配版需要厂商外部初始化Link SDKhandle指针给LinkGuard C SDK处理相关逻辑，Link SDK非适配版无需外部初始化Link SDK，默认提供适配版。

v4.1.0 (2021/10/29)

- 新特性：
  - 更改门禁产品名称为LinkGuard C SDK；
  - 升级依赖Link SDK版本为v4.1.0；
  - 优化代码，完善注释；
- 下载地址：
  - 正式环境：
    - 【海清】arm-linux-gnueabihf(20211109)
  - 预发环境：
    - 【海清】arm-linux-gnueabihf(20211109)

v4.0.1 (2021/09/02)

- 新特性：
  - 增加更新数据模式update_mode，0:重复不更新，1:强制更新；
  - 清除指定分组及所有分组数据时，由删除数据文件改为写入空数据的方式；
  - 本地JSON数据文件无法解析时，删除损坏的数据文件；
  - 统一环境宏定义到头文件；
- 问题修复：

- 修复清除指定分组数据情况下，进度计算问题；

- 下载地址：
  - 正式环境：
    - 【华安】arm-himix100-linux(20211125)
    - 【华安】arm-himix200-linux(20211125)
  - 预发环境：
    - 【华安】arm-himix100-linux(20211125)
    - 【华安】arm-himix200-linux(20211125)

v4.0.0 (2021/03/19)

- 新特性：
  - 统一版本不再区分v1、v2版本，以之前v1版本作为演进基线版本；
  - 升级依赖Link SDK4.x版本，不再依赖Link SDK2.x，内部MQTT、HTTP连云通道切换到Link SDK4.x；
  - 支持企业实例接入点；
  - 隐藏Linkface符号表；
  - 优化目录结构，优化代码，规范命名规则；
  - 部分日志打印优化；
  - 删除iotx_matche_result_t结构体中time_sec、time_usec字段，统一为time_msec；
  - 非实时抓拍结果，通过物模型历史数据topic进行上报；
  - 调整字段long_card_id、time_msec的类型为uint64_t；
- 问题修复：
  - 修复下载图片过大时，释放内存引起的崩溃问题；
  - 修改布控服务进度计算逻辑；
- 下载地址：
  - 正式环境：
    - arm-himix100-linux(20210803)
    - arm-himix200-linux(20210803)
    - sigmastar-arm-linux-gnueabihf-9.1.0(20210803)
  - 预发环境：
    - arm-himix100-linux(20210803)
    - arm-himix200-linux(20210803)

v1.3.3 (2022/02/15)

- 新特性：
  - 更改门禁产品名称为LinkGuard C SDK；
  - 清除指定分组及所有分组数据时，由删除数据文件改为写入空数据的方式；

- 本地JSON数据文件无法解析时，删除损坏的数据文件；
- 优化代码，完善注释；

v1.3.2 (2021/07/07)

- 新特性：
  - 删除AliMatcheResult结构体中time_sec、time_usec字段，统一为time_msec；
  - 非实时抓拍结果，通过物模型历史数据topic进行上报；
  - 调整字段long_card_id、time_msec的类型为uint64_t；
- 问题修复：
  - 修改人脸布控进度计算逻辑；
- 下载地址：
  - 正式环境：
    - 【华安】arm-himix100-linux(20211015)
    - 【华安】arm-himix200-linux(20211015)
    - 【宇视】arm-himix200-linux(20211015)
  - 预发环境：
    - 【华安】arm-himix100-linux(20211015)
    - 【华安】arm-himix200-linux(20211015)
    - 【宇视】arm-himix200-linux(20211015)

v1.3.1 (2021/06/11)

- 新特性：
  - HTTP/2上传代码优化；
- 问题修复：
  - 修正部分设备下，通行事件上报和人脸下发死锁的问题；
- 下载地址：
  - 正式环境：
    - arm-himix100-linux(20210611)
    - arm-himix200-linux(20210611)
  - 预发环境：
    - arm-himix100-linux(20210611)
    - arm-himix200-linux(20210611)

v1.3.0 (2021/03/12)

- 新特性：
  - v1版本写入DIFF文件增加备份机制；

- v1版本增加回调函数Ali_AddPersonDataCallback、Ali_DeletePersonDataCallback、Ali_DeleteAllDataCallback执行时间打点INFO日志；

- 问题修复：
  - v1版本修复DIFF文件为空，SDK初始化失败问题；
- 下载地址：
  - 正式环境：
    - arm-himix100-linux(20210312)
    - arm-himix200-linux(20210312)
  - 预发环境
    - arm-himix100-linux(20210312)
    - arm-himix200-linux(20210312)

v1.2.0 (2020/12/28)

- 新特性：
  - v1版本人脸新增回调接口Ali_AddPersonDataCallback入参添加特征数据字段；
  - v1版本新增人脸特征下发服务处理逻辑；
- 下载地址：
  - 正式环境：
    - arm-himix100-linux(20201228)
    - arm-himix200-linux(20201228)
    - arm-hisiv600-linux-gnueabi(20201229)
    - arm-linux-gnueabihf(20210104)
    - arm-ca53-linux-gnueabihf(20210203)
    - sigmastar-arm-linux-gnueabihf-9.1.0(20210207)
  - 预发环境：
    - arm-himix100-linux(20201228)
    - arm-himix200-linux(20201228)
    - arm-hisiv600-linux-gnueabi(20201229)
    - arm-linux-gnueabihf(20210104)
    - arm-ca53-linux-gnueabihf(20210203)
    - sigmastar-arm-linux-gnueabihf-9.1.0(20210207)

v1.1.0 (2020/12/28)

- 新特性：
  - v1版本增加日志记录功能，新增日志配置结构体LinkfaceLogCfg；
  - v1版本调整初始化方法Ali_Init入参为结构体LinkfaceConfig；

- 问题修复：
  - v1版本修复人脸下发失败无法重复下发问题；
  - v1版本修复调用sdk销毁方法Ali_DeInit阻塞问题；
  - v2版本调整宏定义MAX_COUNT_GROUP_ID的指为1024，兼容服务端用户组超长问题；
  - v1、v2版本修复日志组件多次初始化会失败的问题；
- 下载地址：
  - 正式环境：
    - arm-himix100-linux(20201228)
    - arm-himix200-linux(20201228)
    - arm-hisiv600-linux-gnueabi(20201229)
    - arm-linux-gnueabihf(20210104)
    - arm-ca53-linux-gnueabihf(20210203)
    - sigmastar-arm-linux-gnueabihf-9.1.0(20210207)
  - 预发环境：
    - arm-himix100-linux(20201228)
    - arm-himix200-linux(20201228)
    - arm-hisiv600-linux-gnueabi(20201229)
    - arm-linux-gnueabihf(20210104)
    - arm-ca53-linux-gnueabihf(20210203)
    - sigmastar-arm-linux-gnueabihf-9.1.0(20210207)

1.4. 服务API 1.0

1.4.1. 人脸权限服务

文档已迁移至人脸权限服务

1.4.2. 二维码/卡权限服务

接口以阿里云IoT统一网关HTTP接口的形式提供（使用方法参考API网关客户端）。

访问域名：api.link.aliyun.com，接口协议：HTTPS。

通用对象

人员信息UserDTO

字段	类型	备注
identityId	String	统一身份ID。

字段	类型	备注
name	String	人员称谓。
extInfo	String	自定义信息。

门禁卡权限CardDTO

字段	类型	备注
iotId	String	门禁设备的iotId
expiryTime	String	密码有效期，时间格式 yyyy-MM-dd HH:mm:ss"
cardId	String	门禁卡id
identityId	String	统一身份ID

接口列表

模块	接口名称	接口说明
人员信息管理	添加或更新人员信息	添加或更新人员信息
	删除人员信息	删除某人员信息
	查询人员信息	查询人员信息
	查询人员列表	查询人员列表
刷卡门禁	批量设置门禁卡权限	批量设置门禁卡权限
	删除刷卡门禁上某用户的权限	删除刷卡门禁上某用户的权限
	查询用户的门禁卡权限信息	查询人在指定门禁卡设备上的门禁卡权限信息

	查询门禁设备上的权限信息	查询门禁设备上的权限信息
	查询门禁设备的常开门状态	查询门禁设备的常开门状态
	设置门禁设备的常开门状态	设置门禁设备的常开门状态
二维码门禁	生成二维码	分配二维码
	二维码权限配置	为指定门禁设备配置二维码权限
	二维码权限删除	删除刷卡门禁上某用户的权限
	生成二维码并配置权限（旧）	生成二维码并为指定门禁设备配置权限
密码门禁	设置门禁密码	为密码门禁通行设置门禁密码
远程控制开门	远程控制开门	远程控制开门

添加或更新人员信息

path	版本	是否需要登录
/entrance/paas/user/modify	1.0.0	否

入参

字段	类型	是否必传	备注
identityId	String	是	统一身份ID。若identityId在库里存在走更新流程，若identityId在库里不存在走创建流程
name	String	否	人员称谓。在更新的时候，如不传不会对原有值做修改。在创建的时候必须传。
extInfo	String	否	自定义信息。在更新的时候，如不传不会对原有值做修改。在创建时候可以不填。

出参

返回结果使用通用结果类型，data域为空。

请求示例

```
{
  "identityId":"50acopfd3a94900494085ed2699e052432c34452",
  "name":"王先生",
  "extInfo":{}
}
```

返回示例

```
{
  "code": 200,
  "id":"4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "message": null,
  "localizedMsg": null
}
```

删除人员信息

path	版本	是否需要登录
/entrance/paas/user/delete	1.0.0	否

入参

字段	类型	是否必传	备注
identityId	String	是	统一身份ID。

出参

返回结果使用通用结果类型，data域为空。

请求示例

```
{
  "identityId":"50acopfd3a94900494085ed2699e052432c34452"
}
```

返回示例

```
{
  "code": 200,
  "id":"4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "message": null,
  "localizedMsg": null
}
```

查询人员信息

path	版本	是否需要登录
/entrance/paas/user/get	1.0.0	否

入参

字段	类型	是否必传	备注
identityId	String	是	统一身份ID。

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	JSON	人员信息

请求示例

```
{
    "identityId":"50acopfd3a94900494085ed2699e052432c34452"
}
```

返回示例

```
{
  "code": 200,
  "data": {
      "identityId":"50acopfd3a94900494085ed2699e052432c34452"
      "name": "馄饨面",
      "extInfo":{}
  },
  "id": "4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "message": null,
  "localizedMsg": null
}
```

查询人员列表

path	版本	是否需要登录
/entrance/paas/user/query	1.0.0	否

入参

字段	类型	是否	备注
pageSize	Integer	是	分页查询一页内的记录数，上限100
pageNo	Integer	是	分页数，从1开始

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	JSONObject	返回的分页数据对象

分页数据对象PageDTO

字段	类型	备注
pageSize	Integer	传入的pageSize
pageNo	Integer	传入的pageNo
total	Long	总记录数
data	JSONArray	人员信息

请求示例

```
{
  "pageSize":10,
  "pageNo":1
}
```

返回示例

```
{
  "code": 200,
  "data": {
    "total": 1,
    "data": [{
      "identityId":"50acopfd3a94900494085ed2699e052432c34452"
      "name": "馄饨面",
      "extInfo":{}
  }]
  },
  "id": "4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "message": null,
  "localizedMsg": null
}
```

批量设备门禁卡权限

接口定义

path	版本	是否需要登录
/entrance/paas/perm/card/batchmodify	1.0.2	否

入参

字段	类型	是否必传	备注
cards	JSONArray	是	待创建对象列表，上限1000

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	JSONArray	处理失败的记录，建议可以依照这个列表重新添加

FailureResultDTO

字段	类型	备注
iotId	String	门禁设备的iotId
cardId	String	门禁卡id
identityId	String	统一身份ID

请求示例

```
{
    "cards": [{
    "identityId": "50acopfd3a94900494085ed2699e052432c34452",
        "iotId": "p2OsgId2N2kXMWThRCQP000101",
        "expiryTime": "2020-03-04 18:00:00",
    "cardId":"3d8deae9402d38adfji38"
    }]
}
```

返回示例

```
{
  "code": 200,
  "id":"4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "data":[{
    "identityId": "50acopfd3a94900494085ed2699e052432c34452",
      "iotId":"p2OsgId2N2kXMWThRCQP000101",
    "cardId":"3d8deae9402d38adfji38"
  }],
  "message": null,
  "localizedMsg": null
}
```

删除刷卡门禁上某用户的权限

删除门禁卡的权限实际是删除这个设备上的人的权限

path	版本	是否需要登录
/entrance/paas/perm/card/batchdelete	1.0.1	否

入参

字段	类型	是否必传	备注
list	JSONArray	是	待删除对象列表

PermDTO

字段	类型	是否必传	备注
identityId	String	是	统一身份ID
iotId	String	是	门禁id
cardId	String	是	门禁卡id，最大长度255字符

出参

字段	类型	备注
code	Integer	返回码

字段	类型	备注
message	String	返回信息
data	JSONArray	删除失败的列表

ModifyResultDTO

字段	类型	备注
iotId	String	门禁设备的iotId
cardId	String	门禁卡id
identityId	String	统一身份ID

请求示例

```
{
    "list": [{
        "iotId": "p2OsgId2N2kXMWThRCQP000101",
        "identityId": "50acopfd3a94900494085ed2699e052432c34452",
    "cardId":"3d8deae9402d38adfji38"
    }]
}
```

返回示例

```
{
  "code": 200,
  "id":"4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "data":[],
  "message": null,
  "localizedMsg": null
}
```

查询用户的门禁卡权限信息

path	版本	是否需要登录
/entrance/paas/user/perm/query	1.0.0	否

字段	类型	是否必传	备注
identityId	String	是	统一身份ID
iotIds	JSONArray	是	门禁设备列表，上限100

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	JSONObject	分页对象

分页数据对象PermDTO

identityId	String	统一身份ID
data	JSONArray	门禁卡权限对象

请求示例

```
{
  "identityId":"50acopfd3a94900494085ed2699e052432c34452",
  "iotIds":[
      "p2OsgId2N2kXMWThRCQP000101"
  ]
}
```

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data":{
    "identityId": "p2OsgId2N2kXMWThRCQP000101",
      "data":[{
        "iotId": "50acopfd3a94900494085ed2699e052432c34452",
        "identityId": "p2OsgId2N2kXMWThRCQP000101",
        "expiryTime": "2020-03-04 18:00:00",
        "cardId":"3d8deae9402d38adfji38"
        }]
    }
}
```

查询门禁设备上的权限信息

接口定义

path	版本	是否需要登录
/entrance/paas/device/perm/query	1.0.0	否

入参

字段	类型	是否必传	备注
iotId	String	是	刷卡门禁设备的iotId
pageSize	Integer	是	分页查询一页内的记录数，上限100
pageNo	Integer	是	分页数，从1开始

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	JSONObject	分页对象

分页数据对象PageDTO

pageSize	Integer	传入的pageSize
pageNo	Integer	传入的pageNo
total	Long	总记录数
data	JSONArray	门禁卡权限对象列表

请求示例

```
{
   "iotId":"p2OsgId2N2kXMWThRCQP000101",
   "pageSize":10,
   "pageNo":1
}
```

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null
  "data":{
  "total":1,
  "pageSize":10,
  "pageNo":1,
  "data":[{
        "identityId": "50acopfd3a94900494085ed2699e052432c34452",
            "iotId": "p2OsgId2N2kXMWThRCQP000101",
            "expiryTime": "2020-03-04 18:00:00",
        "cardId":"3d8deae9402d38adfji38"
        }],
    }
}
```

查询门禁设备的常开门状态

接口定义

path	版本	是否需要登录
/entrance/paas/keepopen/get	1.0.0	否

入参

字段	类型	是否必传	备注
iotId	String	是	刷卡门禁设备的iotId

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	Boolean	设备是否处于常开状态

请求示例

```
{
   "iotId":"p2OsgId2N2kXMWThRCQP000101"
}
```

返回示例

```
{
  "code": 200,
  "message": null,
  "data": false
}
```

设置门禁设备的常开门状态

接口定义

path	版本	是否需要登录
/entrance/paas/keepopen/set	1.0.0	否

入参

字段	类型	是否必传	备注
iotId	String	是	刷卡门禁设备的iotId

字段	类型	是否必传	备注
keepOpen	Boolean	是	是否常开

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息

请求示例

```
{
   "iotId":"p2OsgId2N2kXMWThRCQP000101",
  "keepOpen":true
}
```

返回示例

```
{
  "code": 200,
  "message": null
}
```

生成二维码

path	版本	是否需要登录
/homelink/common/qrcode/generate	1.0.1	否

入参

字段	类型	是否必传	备注
identityId	String	是	申请二维码人员的统一身份ID
codeLength	Integer	否	二维码字符串长度，默认是16位

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	String	二维码

请求示例

```
{
  "identityId":"50acopfd3a94900494085ed2699e052432c34452"
}
```

返回示例

```
{
  "code": 200,
  "id":"4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "data":"24DD9END8CD3ABCD",
  "message": null,
  "localizedMsg": null
}
```

二维码权限配置

path	版本	是否需要登录
/entrance/paas/perm/qrcode/config	1.0.1	否

入参

字段	类型	是否必传	备注
qrCode	String	是	二维码
iotIds	JSONArray	是	待配置权限的门禁设备iotId
identityId	String	是	申请二维码人员的统一身份ID

字段	类型	是否必传	备注
effectiveTime	String	否	二维码生效期，时间格式 yyyy-MM-dd HH:mm:ss 默认为当前时间
expiryTime	String	是	二维码失效期，时间格式 yyyy-MM-dd HH:mm:ss
maxScanTimes	Integer	否	最大刷卡次数，默认为-1不做次数限制
maxScanScope	String	否	最大刷码次数作用范围，默认为每个设备独立计数。SHARE - 共享; DEVICE - 设备

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	JSONArray	权限配置结果
->iotId	String	门禁设备ID
->code	Integer	成功为200
->message	String	错误信息

请求示例

```
{
  "qrCode": "24DD9END8CD3",
  "identityId":"50acopfd3a94900494085ed2699e052432c34452",
  "iotIds": [
      "p2OsgId2N2kXMWThRCQP000101"
  ],
  "effectiveTime":"2019-03-04 18:00:00",
  "expiryTime":"2020-03-04 18:00:00"
}
```

返回示例

```
{
  "code": 200,
  "id":"4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "message": null,
  "localizedMsg": null,
  "data": [
    {
      "iotId": "p2OsgId2N2kXMWThRCQP000101",
      "code": "200"
    }
  ]
}
```

二维码权限删除

path	版本	是否需要登录
/entrance/paas/perm/qrcode/remove	1.0.1	否

入参

字段	类型	是否必传	备注
qrCode	String	是	二维码
identityId	String	是	申请二维码人员的统一身份ID
iotIds	JSONArray	是	门禁设备iotId

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	JSONArray	权限配置结果

字段	类型	备注
->iotId	String	门禁设备ID
->code	Integer	成功为200
->message	String	错误信息

请求示例

```
{
  "qrCode": "24DD9END8CD3",
  "iotIds": [
      "p2OsgId2N2kXMWThRCQP000101"
  ],
  "identityId":"50acopfd3a94900494085ed2699e052432c34452"
}
```

返回示例

```
{
  "code": 200,
  "id":"4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "message": null,
  "localizedMsg": null,
    "data": [
    {
      "iotId": "p2OsgId2N2kXMWThRCQP000101",
      "code": "200"
    }
  ]
}
```

生成二维码并配置权限（旧）

（不推荐使用）该接口有限流，每个租户每秒最多设置2000次门禁二维码

path	版本	是否需要登录
/entrance/paas/perm/qrcode/generate	1.0.1	否

入参

字段	类型	是否必传	备注
iotIds	JSONArray	是	待配置权限的门禁设备iotId，最多支持500个设备
identityId	String	是	申请二维码人员的统一身份ID
expiryTime	String	是	二维码有效期，时间格式 yyyy-MM-dd HH:mm:ss 不能超过生效期时间+24h
effectiveTime	String	否	二维码生效期，时间格式 yyyy-MM-dd HH:mm:ss 不能小于当前时间，如为空则取当前时间

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	String	二维码

请求示例

```
{
  "identityId":"50acopfd3a94900494085ed2699e052432c34452",
  "iotIds":[
      "p2OsgId2N2kXMWThRCQP000101"
  ],
  "expiryTime":"2020-03-04 18:00:00",
  "effectiveTime":"2020-03-03 18:00:01"
}
```

返回示例

```
{
  "code": 200,
  "id":"4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "data":"24DD9END8CD3",
  "message": null,
  "localizedMsg": null
}
```

设置门禁密码

path	版本	是否需要登录
/entrance/paas/perm/passwd/modify	1.0.0	否

入参

字段	类型	是否必传	备注
passwords	JSONArray	是	密码信息。最长100条记录

PasswordDTO对象

字段	类型	是否必传	备注
identityId	String	是	统一身份ID。若identityId+iotId在数据库中存在走更新流程，若identityId+iotId不存在走创建流程
iotId	String	是	门禁设备的iotId
password	String	是	密码信息。最大长度255字符
number	Integer	是	密码编号，一个密码门禁有可能有多个密码。
expiredDate	String	是	密码有效期，时间格式 yyyy-MM-dd HH:mm:ss

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	JSONArray	设置失败的记录列表

ModifyResultDTO

字段	类型	备注
iotId	String	门禁设备的iotId

请求示例

```
{
    "identityId": "50acopfd3a94900494085ed2699e052432c34452",
    "passwords": [{
        "iotId": "p2OsgId2N2kXMWThRCQP000101",
        "password": "a2d34f84dfafe",
        "number": "030",
        "expiryDate": "2020-03-04 18:00:00"
    }]
}
```

返回示例

```
{
  "code": 200,
  "id":"4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "data":[
      "iotId":"p2OsgId2N2kXMWThRCQP000101"
  ],
  "message": null,
  "localizedMsg": null
}
```

远程控制开门

接口定义

path	版本	是否需要登录
/cloud/thing/service/invoke	1.0.1	否

入参

字段	类型	是否必传	备注
iotId	String	是	门禁设备的iotId
identifier	String	是	固定值“remoteOpen”

字段	类型	是否必传	备注
args	JSNOObject	是	固定值{}

出参

返回结果使用通用结果类型，data域为空。

请求示例

```
{
    "identifier": "remoteOpen",
    "iotId": "p2OsgId2N2kXMWThRCQP000101",
    "args": {}
}
```

返回示例

```
{
  "code": 200,
  "id":"4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "message": null,
  "localizedMsg": null
}
```

1.5. 服务API 2.0

1.5.1. 关系服务

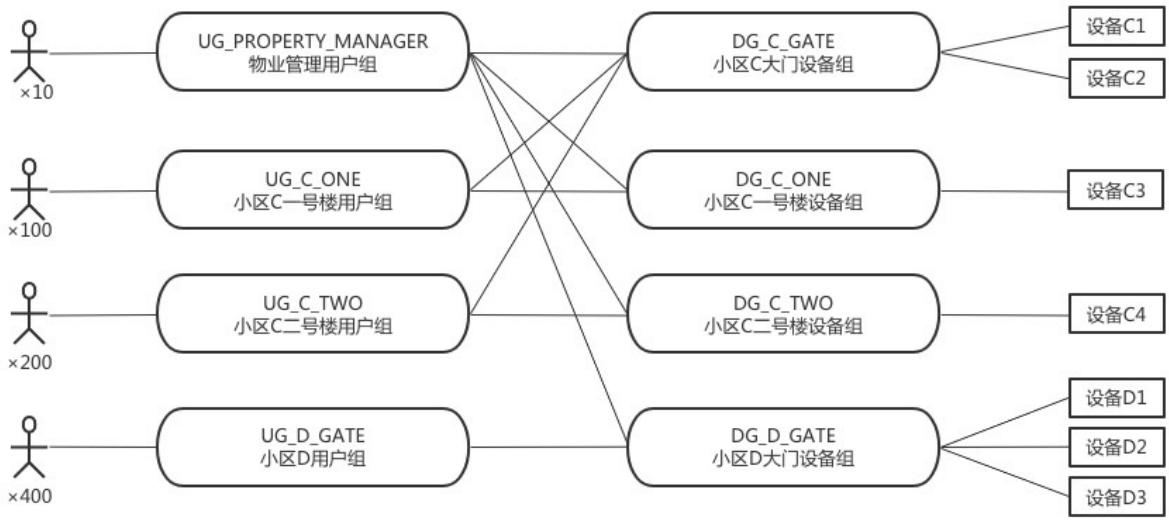
1 概述

本文档是设备与用户基础关系服务接口定义。

基础关系提供：设备 -> 设备组 -> 用户组 -> 用户的关系模型的管理能力。默认情况下，云端会自动处理用户的人脸权限到关联设备的权限部署。每个项目下面的实体数量和关系数量不能超过一定上限，详见第3节。

基础关系可以简化业务平台对下发逻辑的处理。

- 在没有关系管理的情况下，下发人员权限要查询到关联的设备，再一一下发。复杂度是O(n);
- 有关系管理的情况下，只要正常更新人员权限即可。云端会管理所有关联设备的权限下发。用户复杂度是O(1)。云端会通过数据模型把同步结果推送给ISV。



2 接口列表

模块	API名称	API展示名称	API路径	API版本
用户管理	saveDomainUser	保存用户	/domain/user/save	1.0.0
	deleteDomainUser	删除用户	/domain/user/delete	1.0.0
	getDomainUser	获取用户	/domain/user/get	1.0.0
	queryDomainUser	查询用户	/domain/user/query	1.0.0
	saveDomainUserGroup	保存用户组	/domain/user/group/save	1.0.0
	deleteDomainUserGroup	删除用户组	/domain/user/group/delete	1.0.0
	queryDomainUserGroup	查询用户组	/domain/user/group/query	1.0.0
	setUserGroupRelation	绑定/解绑用户与用户组	/domain/user/group/relation/set	1.0.0
	queryUserByGroup	查询用户组中的用户	/domain/group/users/query	1.0.0

	queryGroupByUser	查询用户所属的用户组	/domain/user/groups/query	1.0.0
设备管理	saveDomainDeviceGroup	保存设备组	/domain/device/group/save	1.0.0
	deleteDomainDeviceGroup	删除设备组	/domain/device/group/delete	1.0.0
	queryDomainDeviceGroup	查询设备组	/domain/device/group/query	1.0.0
	setDeviceGroupRelation	绑定/解绑设备与设备组	/domain/device/group/relation/set	1.0.0
	queryDeviceByGroup	查询设备组中的设备	/domain/group/devices/query	1.0.0
	queryGroupByDevice	查询设备所属的设备组	/domain/device/groups/query	1.0.0
组关系管理	setGroupRelation	绑定/解绑用户组与设备组	/domain/group/relation/set	1.0.0
	queryRelationByUserGroup	查询用户组绑定的设备组	/domain/user/group/relations/query	1.0.0
	queryRelationByDeviceGroup	查询设备组绑定的用户组	/domain/device/group/relations/query	1.0.0
	queryGroupRelation	查询用户组和设备组的绑定关系	/domain/group/relation/query	1.0.0
权限管理	saveFaceImageV2	保存用户人脸	/domain/face/image/save	1.0.0
	deleteFaceImageV2	删除用户人脸	/domain/face/image/delete	1.0.0
	queryPermStatusByUser	查询用户的权限下发状态	/domain/perm/status/user/query	1.0.0

状态管理	queryPermStatusByDevice	查询设备的权限下发状态	/domain/perm/status/device/query	1.0.1

保存用户

路径：/domain/user/save

版本：1.0.0

入参：

名称	类型	是否必须	说明
userType	String	是	用户类型，支持IDENTITY/OPEN
userId	String	是	用户id，支持identityId/openId，最大支持64位
userName	String	否	用户名，最大支持128位
mobile	String	否	手机号，最大支持64位

出参：（无）

删除用户

路径：/domain/user/delete

版本：1.0.0

入参：

名称	类型	是否必须	说明
userType	String	是	用户类型，支持IDENTITY/OPEN
userId	String	是	用户id，支持identityId/openId，最大支持64位

出参：（无）

获取用户

路径：/domain/user/get

版本：1.0.0

入参：

名称	类型	是否必须	说明
userType	String	是	用户类型，支持IDENTITY/OPEN
userId	String	是	用户id，支持identityId/openId，最大支持64位

出参：

名称	类型	说明
userType	String	用户类型
userId	String	用户id
userName	String	用户名
mobile	String	手机号
userFace	JSON	用户人脸结构体
faceId	String	人脸id
imageUrl	String	人脸图URL
expiryTime	String	人脸有效期，格式yyyy-MM-dd HH:mm:ss
userQrCodes	List	用户二维码列表（本期暂不实现）
codeType	String	二维码类型
qrCode	String	二维码
startTime	String	生效时间，格式YYYY-MM-DD hh:mm:ss
endTime	String	失效时间，格式YYYY-MM-DD hh:mm:ss

maxScanTimes	int	最大刷码次数
maxScanScope	String	最大刷码次数作用范围，默认为每个设备独立计数（边缘专用）
userCards	List	用户卡列表（本期暂不实现）
cardType	String	卡类型
cardId	String	卡号
startTime	String	生效时间，格式YYYY-MM-DD hh:mm:ss
endTime	String	失效时间，格式YYYY-MM-DD hh:mm:ss

查询用户

路径：/domain/user/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数, 最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数

data	List	数据列表
userType	String	用户类型
userId	String	用户id
userName	String	用户名
mobile	String	手机号

保存用户组

路径：/domain/user/group/save

版本：1.0.0

入参：

名称	类型	是否必须	说明
userGroupId	String	否	用户组id，可由调用方指定或随机生成，最大支持64位，由字母、数字和横线、下划线以及点号组成
userGroupName	String	是	用户组名称，最大支持32位
description	String	否	描述信息，最大支持256位

出参：

名称	类型	说明
userGroupId	String	用户组id

删除用户组

路径：/domain/user/group/delete

版本：1.0.0

入参：

名称	类型	是否必须	说明

userGroupId	String	是	用户组id，可由调用方指定或随机生成，最大支持64位，由字母、数字和横线、下划线以及点号组成

出参：（无）

查询用户组

路径：/domain/user/group/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数，最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数
data	List	数据列表
userGroupId	String	用户组id
userGroupName	String	用户组名称
description	String	描述信息

绑定/解绑用户与用户组

路径：/domain/user/group/relation/set

版本：1.0.0

入参：

名称	类型	是否必须	说明
userGroupId	String	是	用户组id
userType	String	是	用户类型，支持IDENTITY/OPEN
userId	String	是	用户id，支持identityId/openId，最大支持64位
operation	String	是	绑定操作，支持BIND/UNBIND，即绑定与解绑

出参：（无）

查询用户组中的用户

路径：/domain/group/users/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
userGroupId	String	是	用户组id
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数，最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数

data	List	数据列表
userType	String	用户类型
userId	String	用户id
userName	String	用户名
mobile	String	手机号

查询用户所属的用户组

路径：/domain/user/groups/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
userType	String	是	用户类型，支持IDENTITY/OPEN
userId	String	是	用户id，支持identityId/openId，最大支持64位
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数，最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数

data	List	数据列表
userGroupId	String	用户组id
userGroupName	String	用户组名称
description	String	描述信息

保存设备组

路径：/domain/device/group/save

版本：1.0.0

入参：

名称	类型	是否必须	说明
deviceGroupId	String	否	设备组id，可由调用方指定或随机生成，最大支持64位，由字母、数字和横线、下划线以及点号组成
deviceGroupName	String	是	设备组名称，最大支持32位
description	String	否	描述信息，最大支持256位

出参：

名称	类型	说明
deviceGroupId	String	设备组id

删除设备组

路径：/domain/device/group/delete

版本：1.0.0

入参：

名称	类型	是否必须	说明

deviceGroupId	String	否	设备组id，可由调用方指定或随机生成，最大支持64位，由字母、数字和横线、下划线以及点号组成

出参：（无）

查询设备组

路径：/domain/device/group/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数，最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数
data	List	数据列表
deviceGroupId	String	设备组id
deviceGroupName	String	设备组名称
description	String	描述信息

绑定/解绑设备与设备组

路径：/domain/device/group/relation/set

版本：1.0.0

入参：

名称	类型	是否必须	说明
deviceGroupId	String	是	设备组id，如该id未保存过，则系统会默认创建一条名称和描述信息为空的记录
iotId	String	是	设备id
operation	String	是	绑定操作，支持BIND/UNBIND，即绑定与解绑

出参：（无）

查询设备组中的设备

路径：/domain/group/devices/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
deviceGroupId	String	是	设备组id
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数，最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数
data	List	数据列表

iotId	String	设备id

查询设备所属的设备组

路径：/domain/device/groups/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
iotId	String	是	设备id
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数, 最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数
data	List	数据列表
deviceGroupId	String	设备组id
deviceGroupName	String	设备组名称
description	String	描述信息

绑定/解绑用户组与设备组

路径：/domain/group/relation/set

版本：1.0.0

入参：

名称	类型	是否必须	说明
userGroupId	String	是	用户组id，如该id未保存过，则系统会默认创建一条名称和描述信息为空的记录
deviceGroupId	String	是	设备组id，如该id未保存过，则系统会默认创建一条名称和描述信息为空的记录
operation	String	是	绑定操作，支持BIND/UNBIND，即绑定与解绑

出参：（无）

查询用户组绑定的设备组

路径：/domain/user/group/relations/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
userGroupId	String	是	用户组id
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数，最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数
data	List	数据列表

deviceGroupId	String	设备组id
deviceGroupName	String	设备组名称
description	String	描述信息

查询设备组绑定的用户组

路径：/domain/device/group/relations/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
deviceGroupId	String	是	设备组id
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数, 最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数
data	List	数据列表
userGroupId	String	用户组id
userGroupName	String	用户组名称
description	String	描述信息

查询用户组和设备组的绑定关系

路径：/domain/group/relation/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数, 最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数
data	List	数据列表
userGroupId	String	用户组id
userGroupName	String	用户组名称
userGroupDescription	String	用户组描述信息
deviceGroupId	String	设备组id
deviceGroupName	String	设备组名称
deviceGroupDescription	String	设备组描述信息

保存用户人脸

路径：/domain/face/image/save

版本：1.0.0

入参：

名称	类型	是否必须	说明
userType	String	是	用户类型，支持IDENTITY/OPEN
userId	String	是	用户id，支持identityId/openId，最大支持64位
userName	String	否	用户名，最大支持128位
mobile	String	否	手机号，最大支持64位
faceId	String	否	人脸id，最大支持64位
imageUrl	String	否	人脸图URL，与imageBase64二选一
imageBase64	String	否	人脸图base64，与imageUrl二选一
faceExtInfo	String	否	扩展字段
expiryTime	String	否	过期时间，格式YYYY-MM-DD hh:mm:ss
policy	String	否	人脸匹配后人员通行策略：PERMISSION（默认）-允许；DENY-禁止通行

出参：（无）

删除用户人脸

路径：/domain/face/image/delete

版本：1.0.0

入参：

名称	类型	是否必须	说明
userType	String	是	用户类型，支持IDENTITY/OPEN
userId	String	是	用户id，支持identityId/openId，最大支持64位

出参：（无）

查询用户的权限下发状态

路径：/domain/perm/status/user/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
userType	String	是	用户类型，支持IDENTITY/OPEN
userId	String	是	用户id，支持identityId/openId，最大支持64位
permType	String	是	查询类型，当前只支持FACE
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数, 最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数

total	int	总条数
data	List	数据列表
iotId	String	设备id
facePerm	JSON	人脸权限
- faceId	String	人脸id
- syncTime	String	下发时间
- status	String	下发状态

查询设备的权限下发状态

路径：/domain/perm/status/device/query

版本：1.0.1

入参：

名称	类型	是否必须	说明
iotId	String	是	设备id
permType	String	是	查询类型，当前只支持FACE
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数, 最大100条

出参：

名称	类型	说明
pageNo	int	当前页码

pageSize	int	每页条数
total	int	总条数
data	List	数据列表
userType	String	用户类型
userId	String	用户id
userName	String	用户名
mobile	String	手机号
facePerm	JSON	人脸权限
- faceId	String	人脸id
- syncTime	String	下发时间
- status	String	下发状态

3 使用限制

单个项目下，各类实体的数量与关系绑定的数量存在以下限制：

1. 用户数上限：50000
2. 设备数上限：500
3. 用户组数上限：20
4. 设备组数上限：20
5. 单用户绑定的用户组数上限：10
6. 单设备绑定的设备组数上限：10
7. 单用户组绑定的设备组数上限：10
8. 单设备组绑定的用户组数上限：10

如有特殊需求，请与阿里小二联系。人如有特殊需求，请与联阿里处理

1.5.2. 人脸权限管理

本文档是人脸权限相关的管理接口。

模块	API展示名称	API路径	API版本
权限管理	保存用户人脸	/domain/face/image/save	1.0.0
	删除用户人脸	/domain/face/image/delete	1.0.0
状态管理	查询用户的权限下发状态	/domain/perm/status/user/query	1.0.0
	查询设备的权限下发状态	/domain/perm/status/device/query	1.0.0
维护管理	人脸门禁设备数据同步	/face/paas/permission/sync	1.0.0
	重部署设备人脸权限	/face/paas/permissions/redeploy	1.0.0
	查询设备端人脸权限总数	/face/paas/permission/total/get	1.0.0
	重部署needManual状态的人脸信息	/face/paas/permission/needmanual/redeploy	1.0.1

保存用户人脸

路径：/domain/face/image/save

版本：1.0.0

入参：

名称	类型	是否必须	说明
userType	String	是	用户类型，支持IDENTITY/OPEN
userId	String	是	用户id，支持identityId/openId，最大支持64位
userName	String	否	用户名，最大支持128位
mobile	String	否	手机号，最大支持64位

faceId	String	否	人脸id，最大支持64位
imageUrl	String	否	人脸图URL，与imageBase64二选一
imageBase64	String	否	人脸图base64，与imageUrl二选一
faceExtInfo	String	否	扩展字段
expiryTime	String	否	过期时间，格式YYYY-MM-DD hh:mm:ss
policy	String	否	人脸匹配后人员通行策略：PERMISSION（默认）-允许；DENY-禁止通行

出参：（无）

删除用户人脸

路径：/domain/face/image/delete

版本：1.0.0

入参：

名称	类型	是否必须	说明
userType	String	是	用户类型，支持IDENTITY/OPEN
userId	String	是	用户id，支持identityId/openId，最大支持64位

出参：（无）

查询用户的权限下发状态

路径：/domain/perm/status/user/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
userType	String	是	用户类型，支持IDENTITY/OPEN
userId	String	是	用户id，支持identityId/openId，最大支持64位
permType	String	是	查询类型，当前只支持FACE
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数, 最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数
data	List	数据列表
iotId	String	设备id
facePerm	JSON	人脸权限
- faceId	String	人脸id
- syncTime	String	下发时间
- status	String	下发状态

查询设备的权限下发状态

路径：/domain/perm/status/device/query

版本：1.0.0

入参：

名称	类型	是否必须	说明
iotId	String	是	设备id
permType	String	是	查询类型，当前只支持FACE
pageNo	int	是	当前页码，从1开始
pageSize	int	是	每页条数，最大100条

出参：

名称	类型	说明
pageNo	int	当前页码
pageSize	int	每页条数
total	int	总条数
data	List	数据列表
userType	String	用户类型
userId	String	用户id
userName	String	用户名
mobile	String	手机号

facePerm	JSON	人脸权限
- faceId	String	人脸id
- syncTime	String	下发时间
- status	String	下发状态

人脸门禁设备数据同步

触发云端检查人脸设备是否完成权限数据同步并补发未完成的数据。

接口路径：/face/paas/permission/sync

接口版本：1.0.0

参数列表

参数名称	类型	说明
iotId	String	设备id

参数示例

入参示例：

```
{"iotId": "uulrbimJwUzbvTYdxHZd000000"}
```

出参示例：

```
{"code": 200,"message": "success"}
```

错误码

code	message	说明
200	success	触发成功
460	device project relation mismatch	无权访问该设备（项目鉴权失败）
2005	无设备权限	无权访问该设备（租户鉴权失败）
2015	设备的产品类型不支持	设备类型不支持下发人脸

2028	sync task is in progress	设备正在同步人脸，无需再次触发
2030	设备不在线	设备不在线

重部署设备人脸权限

当设备端需要清空数据（包括本机数据和SDK数据）或更换设备时，可以调用重部署接口重新把云端数据推送到设备端。接口会检查设备端：拉取SDK上报数据为空查询设备端权限计数为0（不支持的设备会忽略）注意：清空的设备，或新换三元组的设备上线后，一定要先执行重部署，让云端和设备端状态达到一致。再做添加或删除人脸的操作。如果不执行重部署就下发权限，会触发一次全量人脸下发，设备端会长时间无响应（下发可能成功，也可能失败。

接口路径：/face/paas/permissions/redeploy

接口版本：1.0.0

入参

字段名	类型	必须	默认值	参数说明
iotId	String	是	无	设备

出参

字段名	类型	非空	参数说明
code	Integer	是	成功：200
message	String		消息
data	JSONObject	是	数据部
total	Integer	是	设备待下发人脸总数

入参示例：

```
{"iotId": "uulrbimJwUzbvTYdxHZd000000"}
```

出参示例：

```
{"code": 200,"message": "success","data": {"total": 3000}}
```

错误码

code	message	说明
200	success	触发成功
460	device project relation mismatch	无权访问该设备（项目鉴权失败）
2005	无设备权限	无权访问该设备（租户鉴权失败）
2015	设备的产品类型不支持	设备类型不支持下发人脸
2028	服务端繁忙，请稍后重试	设备正在同步人脸，无需再次触发
2030	设备不在线	检查设备在网络，如果一切正常，启动后需要等待几分钟
80002	无设备权限	
80003	不支持的设备品类	
80004	设备上有残留的权限	权限计数查询非0，不支持查询的设备不会报这个错误
80005	查询设备权限错误	
80006	设备权限非空	设备SDK上的人脸权限未清空
80007	只支持V1版本门禁设备	不支持2.0门禁设备
80008	设备被加锁，等待调度重部署逻辑	如果云端发现设备锁，说明有处理进程在运行，不支持重部署。

查询设备端人脸权限总数

直接查询设备端计数要求设备端实现QueryFacePermTotal服务，如果没有实现则返回-1

接口路径：/face/paas/permission/total/get

接口版本：1.0.0

入参：无

出参：

字段名	类型	非空	参数说明
code	Integer	是	成功：200
message	String		消息
data	JSONObject	是	数据部
total	Integer	是	设备端人脸权限总数 >= 0，正常查询-1，设备端不支持查询

入参示例：

```
{"iotId": "uulrbimJwUzbvTYdxHZd000000"}
```

出参示例：

```
{"code": 200,"message": "success","data": {"total": 128}}
```

设备端未实现：

```
{"code": 200,"message": "success","data": {"total": -1}}
```

错误码

code	message	说明
200	success	触发成功
460	device project relation mismatch	无权访问该设备（项目鉴权失败）
2005	无设备权限	无权访问该设备（租户鉴权失败）
2015	设备的产品类型不支持	设备类型不支持下发人脸
2028	sync task is in progress	设备正在同步人脸，无需再次触发
2030	设备不在线	设备不在线

80001	查询设备计数失败	如超时等导致的查询失败

重部署needManual状态的人脸信息

重置needManual状态权限
接口路径：/face/paas/permission/needmanual/redeploy
接口版本：1.0.1

入参

字段名	类型	必须	默认值	参数说明
iotId	String	是	无	设备

出参

字段名	类型	非空	参数说明
code	Integer	是	成功：200
message	String		消息
data	JOSNObject	是	数据
--total	Integer	是	addList的元素数量
--addList	JSONArray		恢复的待添加权限
--delList	JSONArray		待删除的权限

入参示例：

```
{"iotId": "uulrbimJwUzbvTYdxHZd000000"}
```

出参示例：

```
{"code": 200,"message": "success","data": {"total": 1},"addList":[],"delList":[]}
```

错误码

code	message	说明
200	success	触发成功
460	device project relation mismatch	无权访问该设备（项目鉴权失败）
2005	无设备权限	无权访问该设备（租户鉴权失败）
2015	设备的产品类型不支持	设备类型不支持下发人脸
2028	sync task is in progress	设备正在同步人脸，无需再次触发
2030	设备不在线	设备不在线

1.5.3. 功能API

列表

- 设置常开门 /entrance/paas/keepopen/set
- 查询常开门状态/entrance/paas/keepopen/get
- 查询AIBOX上报图片URL/vision/picture/querybyids

设置常开门

接口路径：/entrance/paas/keepopen/set

接口版本：1.0.0

入参

参数	类型	说明
iotId	String	设备id
keepOpen	Boolean	是否常开

出参

使用通用结果类型。

入参示例

```
{"iotId": "p2OsgId2N2kXMWThRCQP000101","keepOpen": true}
```

出参示例

```
{"code": 200,"message": null}
```

查询常开门状态

接口路径：/entrance/paas/keepopen/get

接口版本：1.0.0

入参

参数	类型	说明
iotId	String	设备id

出参

使用通用结果类型。data为Boolean类型，表示是否常开。

入参示例

```
{"iotId": "p2OsgId2N2kXMWThRCQP000101"}
```

出参示例

```
{"code": 200,"message": null,"data": true}
```

查询AIBOX上报图片URL

接口路径：/vision/picture/querybyids

接口版本：2.0.0

入参

参数	类型	必填	说明
iotId	String	是	设备id
pictureIdList	Array<String>	是	图片ID列表
pictureType	Integer		图片类型：0全部，1原图，2缩率图

出参

使用通用结果类型，data为对象：

参数	类型	非空	说明
pictureList	Array<Object>	是	图片列表
iotId	String	是	设备ID
pictureId	String		图片ID
pictureTime	String		图片创建时间
pictureUrl	String		图片地址
thumbUrl	String		缩略图地址

入参示例

```
{
    "iotId": "p2OsgId2N2kXMWThRCQP000101",
    "pictureIdList": ["xyUjyangedwUke","adfafiYkangleI"],
    "pictureType": 0
}
```

出参示例

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
        "pictureList":[
            {
                "iotId":"9322443e-0711-427b-9dd2-8035d9fed1c7",
                "pictureId":"9322443e8035d9fed1c",
                "pictureTime":"2019-08-23 14:43:45",
                "endTime":"2019-08-23 14:45:44",
                "pictureUrl":"https://xxxx",
                "thumbUrl":"https://xxxx"
            }
        ]
    }
}
```

1.6. 数据订阅

门禁服务提供数能力向ISV推送通行事件，设备报警以及部署过程数据。

包括：

数据能力	说明
门禁通行	支持二维码、卡/蓝牙、密码
刷脸通行	支持人脸1.0和2.0
陌生人脸通行	支持人脸1.0 OnDetect事件
业务数据推送	人脸部署状态变更、设备全量权限、设备同步错误
设备报警事件	设备上下线，以及其它设备报警事件推送
健康码验证记录	健康码门禁的校验结果推送

数据授权方式

通过开放API可以实现数据订阅授权，授权完成后，利用授权appKey可以实时订阅刷脸记录，并通过开放API查询刷脸记录。

数据订阅与查询参考：数据总线

1.6.1. 门禁通行事件

门禁通行事件，支持门禁卡、密码、二维码类型

模型ID：iot_entrance_event_model 模型版本：1.2

属性名称	属性标识符	数据类型	数据描述	必须
租户	tenantId	String	租户	是
IoT体系用户标示	identityId	String	IoT体系用户标示	是
用户姓名	name	String	用户姓名	是
事件时间	time	Date	事件时间	是
IoT设备标示	iotId	String	IoT设备标示	是

属性名称	属性标识符	数据类型	数据描述	必须
通行类型	type	String	通行类型 CARD - 门禁卡 PASSWORD - 密码 QRCODE - 二维码	是
密码号码	number	String	密码号码	
门禁卡号	cardId	String	门禁卡号，只有type是CARD的时候有效	
通行方向	direction	String	通行方向，ENTRANCE 进, EXIT 出	
用户扩展信息	userExtInfo	String	用户扩展信息	
设备扩展信息	devExtInfo	String	设备扩展信息	

1.6.2. 人脸通行事件

人脸通行事件推送

模型ID：iotx_face_record_model 模型版本：1.6

属性名称	属性标识符	数据类型	非空	数据描述
记录标示	eventId	String		记录标示
创建时间	gmtCreate	Date		创建时间
上报时间	eventTime	Date		上报时间
用户Id	userId	String		用户Id，和人脸图片的userId对应
用户类型	userType	String		用户类型，支持IDENTITY、OPEN等
设备扩展信息	deviceExtInfo	String		设备扩展信息

属性名称	属性标识符	数据类型	非空	数据描述
通行方式	mediaType	String		开门方式，例如face
设备id	iotId	String		设备id
空间id	spaceId	String		如果设备绑到空间下，表示设备所在空间Id
用户信息扩展信息	userExtInfo	String		用户扩展信息
数据来源行业	source	String		数据来源行业（人居、园区等）
人脸抓拍图片	pictureUrl	String		人脸抓拍图片URL
根空间ID	rootSpaceId	String		设备所在根空间ID，用于实现业务隔离
通行策略	policy	String		通行策略

1.6.3. 陌生人通行事件

当门禁支持人脸检测，又没有匹配到具体人员时，会上报陌生人事件。

模型ID：StrangerFaceModel 模型版本：1.0

属性名称	属性标识符	数据类型	长度限制	非空	数据描述
记录标示	eventId	String	256	是	记录标示
事件时间	eventTime	Date		是	事件时间
创建时间	gmtCreate	Date		是	创建时间
设备id	iotId	String	256	是	设备id
用户信息扩展信息	userExtInfo	String	256		用户扩展信息

属性名称	属性标识符	数据类型	长度限制	非空	数据描述
数据来源行业	source	String	256		数据来源行业（人居、园区等）
人脸抓拍图片	pictureUrl	String	1024	是	人脸抓拍图片URL，有效期7天
人脸图片ID	pictureKey	String	1024	是	人脸抓拍图片Key
通行策略	policy	String	256		通行策略
项目	projectId	String	64	是	项目ID
事件扩展信息	eventExtInfo	String	1024		事件扩展信息

1.6.4. 设备报警事件

设备上报的报警事件推送。IoT云端订阅设备alarmEvent事件和上下线的MQ消息，将其转换为DOP数据模型，由上层应用订阅获取设备消息。

模型ID：DeviceAlarmEventBasic 模型版本：1.0

属性名称	属性标识符	数据类型	数据描述
租户id	tenantId	String	租户id
事件时间	time	Date	事件时间
设备id	iotId	String	IoT设备标示
报警类型	alarmType	String	字符串类型的报警枚举
扩展信息	extInfo	String	扩展信息

报警类型alarmType取值范围：

- 设备上线

- 设备离线
- 防拆报警解除
- 强制开门报警
- 强制开门报警解除
- 开门超时报警
- 开门超时报警解除

1.6.5. 健康码验证记录

健康码验证记录。当业务侧开通健康码能力并实现云端对接后，人员通行会校验健康码状态。校验结果通过数据推送到业务侧。

模型ID：HealthCodeVerifyRecord 模型版本：1.0

属性名称	属性标识符	数据类型	数据描述
记录ID	eventId	String	记录标示，业务自定义
记录时间	eventTime	Date	事件时间
健康码数据源	healthDataSource	String	健康码数据源
项目ID	projectId	String	项目ID
设备ID	iotId	String	发起验证请求的设备ID
认证类型	certType	String	请求的认证类型，取值范围：QR_CODE（健康码二维码），ID_CARD（身份证号）
认证数据	certValue	String	请求的认证数据
健康状态	healthStatus	Integer	健康码状态枚举值：1-绿码，2-黄码，3-红码，4-已过期，5-不存在，6-核验失败，7-核验超时
状态更新时间	healthStatusUpdateTime	Date	健康码最近更新时间
证件类型	idCardType	String	证件类型：ID_CARD（身份证号）

属性名称	属性标识符	数据类型	数据描述
证件号码	idCardNo	String	证件号码（脱敏）
用户姓名	name	String	姓名（脱敏）
手机号	phoneNo	String	手机号（脱敏）
设备IC卡号	iccId	String	设备物联网卡卡号

1.6.6. 无感通行记录

当使用AIBOX和摄像头接入无感通行时，通过该模型向业务侧推送通行数据。

通行支持三种类型，通过passType表示：

- 1 - 识别可通行
- 2 - 识别禁卡通行
- 3 - 陌生人

ModelId: FaceDetectPassageModel

version: 1.6

名称	标识符	类型	非空	备注
事件ID	eventId	String	是	事件ID，唯一
事件时间	eventTime	Date	是	事件上报时间
上报事件码	eventCode	String	是	AIBOX上报的事件码，恒为IntelligentAlarm
报警图片ID	alarmPicId	String	是	报警图片ID[1]
报警图片URL	alarmPicUrl	String	是	报警图片下载URL[2]
人脸坐标	faceLeftTop	String	是	人脸的轮廓外接矩形在图画中的左上角位置[3]

人脸坐标	faceRightBottom	String	是	人脸的轮廓外接矩形在图画中的右下角位置[3]
性别代码	gender	Integer		0 - 未知 1 - 男 2 - 女
年龄	age	Integer		年龄，0为未知年龄
人脸置信度	faceReliability	Float	是	人脸置信度 [0-1]
口罩置信度	respiratorReliability	Float		口罩置信度 [0-1]
温度	temperature	Float		人脸温度
用户自定义属性	userDefineProps	String		用户自定义信息,JSON字符串，建议 "[{\"key\":value},{key2:vaule2}]"
用户开放标识	openId	String		识别用户的三方账号ID[4]
用户标识	identityId	String		识别用户的内部ID[4]
人脸ID	faceId	String		识别用户的人脸ID[4]
人脸相似度	similarityDegree	Float		人脸相似度[4]
用户信息	userInfo	String		用户信息[4]
名字	name	String		需要业务侧提交用户信息时提供姓名[4]
自定义账号	privateAccount	String		自定义账号，保留字段
允许通行	allowPass	Boolean	是	是否允许通行，只在passType=1时为true。

通行方式	passType	Integer	是	1 - 识别可通行 2 - 识别禁卡通行 3 - 陌生人
设备ID	iotId	String	是	AIBOX的设备id，与gatewayProductKey和gatewayDeviceName唯一映射。
分组ID	groupId	String	是	分组ID，为设备ID
摄像头pk	cameraProductKey	String	是	采集摄像头设备PK
摄像头dn	cameraDeviceName	String	是	采集摄像头设备DN
AIBOXpk	gatewayProductKey	String	是	AIBOX上报时有值
AIBOX dn	gatewayDeviceName	String	是	AIBOX上报时有值
项目ID	projectId	String	是	当前人脸图对应的项目ID

说明：

- [1] 当有问题需要排查时提供图片ID，一般情况下可以忽略；
- [2] 报警图片URL的有效期是1小时；
- [3] 坐标数值部分范围是[0-1]，由两个数值表示一个坐标点，如：“0.6299,0.4700”
- [4] 当passType=1和2时有值

1.6.7. 区域闯入事件

区域入侵事件

ModelId: ZoneBreakInDetectModel

Version: 1.4

名称	标识符	类型	非空	备注
事件ID	eventId	String(64)	是	事件ID，唯一

事件时间	eventTime	Date	是	事件上报时间
上报事件码	eventCode	String(64)	是	AIBOX上报的事件码，恒为IntelligentAlarm
报警图片ID	alarmPicId	String	是	报警图片ID[1]
报警图片URL	alarmPicUrl	String	是	报警图片下载URL[2]
左上角坐标	faceLeftTop	String(64)	是	人脸的轮廓外接矩形在图画中的左上角位置[3]
右下角坐标	faceRightBottom	String(64)	是	人脸的轮廓外接矩形在图画中的右下角位置[3]
目标类型	targetType	Integer	是	目标的类型： 1：人员 3：机动车
设备ID	iotId	String	是	AIBOX的设备id，与gatewayProductKey和gatewayDeviceName唯一映射。
摄像头设备ID	cameraIotId	String	是	采集摄像头设备ID
摄像头pk	cameraProductKey	String	是	采集摄像头设备PK
摄像头dn	cameraDeviceName	String	是	采集摄像头设备DN
AIBOXpk	gatewayProductKey	String	是	AIBOX上报时有值
AIBOX dn	gatewayDeviceName	String	是	AIBOX上报时有值
项目ID	projectId	String	是	当前人脸图对应的项目ID

说明：

- [1] 当有问题需要排查时提供图片ID，一般情况下可以忽略；
- [2] 报警图片URL的有效期是1小时；
- [3] 坐标数值部分范围是[0-1]，由两个数值表示一个坐标点，如： “0.6299,0.4700”

1.6.8. 业务数据推送

业务数据推送用于云端针对不同业务推送各类通知信息。

截止到2021.2.5，支持以下数据推送

- 人脸状态变更
- 设备同步错误
- 设备上报数据

模型ID：BizDataPushModel 模型版本：1.1

属性名称	属性标识符	类型	长度	非空	数据描述
记录标示	eventId	String	256	是	记录标示
事件时间	eventTime	Date		是	事件时间
业务类型	bizType	String	64	是	当前消息的类型人脸同步状态 FACE_SYNC_STATUS设备同步异常 DEVICE_SYNC_ERROR人脸设备上报权限数据 FACE_DEVICE_PERMS
业务标识	bizId	String	64	是	一般来说bizId是ISV可以索引的业务实体标识。由bizIdType标识这个字段要描述的信息。
业务标识类型	bizIdType	String	128	是	描述bizId表示的内容IOT_ID 设备 FACE_ID 人脸
业务数据	data	String	2048		数据部，可空
业务数据URL	dataUrl	String	1024		如果数据较大，可以通用URL下发数据，
数据URL的有效期	dataUrlExpire	Date			数据URL的有效期

属性名称	属性标识符	类型	长度	非空	数据描述
消息级别	eventLevel	String	64	是	消息的级别INFOWARNERROR
消息描述信息	eventText	String	2048		推送文本，如当前消息的处理建议

1) 人员状态变更

- 当人脸开始下发时，推送toBeTransferred状态
- 当完成下发时，推送transferred或transferDeleted
- 当下发异常时，推送needManual
- preStatus是变更前的状态，可能值有
  - toBeTransferred
  - deviceOffline
  - faceCheckTimeout
  - faceCheckError
  - facePushError
  - unknownError
  - transferTimeout
  - deleteTimeout
  - needManual

data域

属性名称	属性标识符	类型	非空	数据描述
人员ID	identityId	String	是	
人脸ID	faceId	String	是	
状态	status	String	是	
前置状态	preStatus	String		

2) 设备同步错误

当同步设备出现错误时，推送数据。

data域

属性名称	属性标识符	类型	非空	数据描述
设备ID	iotId	String	是	
错误结果	result	String	是	
错误码	code	Integer		
错误信息	message	String		

3) 设备上报数据

当查询设备全量数据时，同步推送数据。

data域是URL，指向设备上报数据的下载地址

1.7. 其他功能

1.7.1. 无感通行

无感通行是指用户在通过鉴权通道时自动校验，如果有权限则直接通过，没有阻拦。无权限用户会通过业务人员拦截。

用户需要采购识别摄像头、AI边缘服务器（请联系商务同学），并和IOT行业平台对接。

无感通行能力基于**服务API 2.0**提供的关系服务，云端关系代表通行权限。

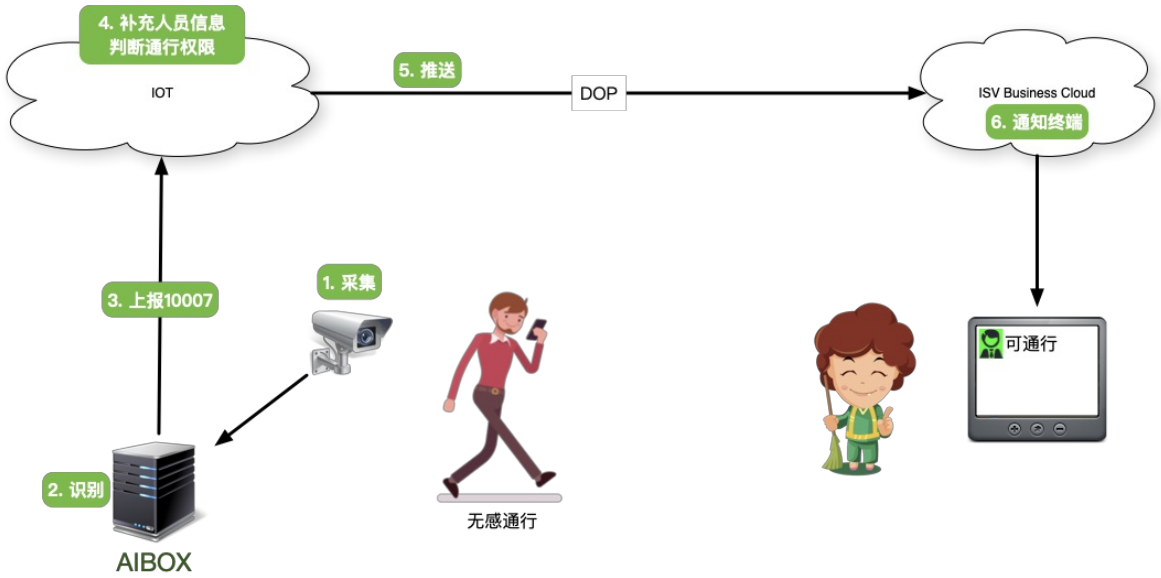
业务流程

在施工完成后，业务流程：

1. 业务方配置用户和设备的关系；
2. 云端向AIBOX下发用户权限；
3. 设备采集到人员通过；
4. 在边缘识别
5. 上报识别结果
6. 云端识别通行权限，判断是否有权限通行
7. 推送给业务平台
8. 业务平台订阅并接收数据
9. 推送给业务终端，通知业务人员执行动作

场景

有权限的无感通行

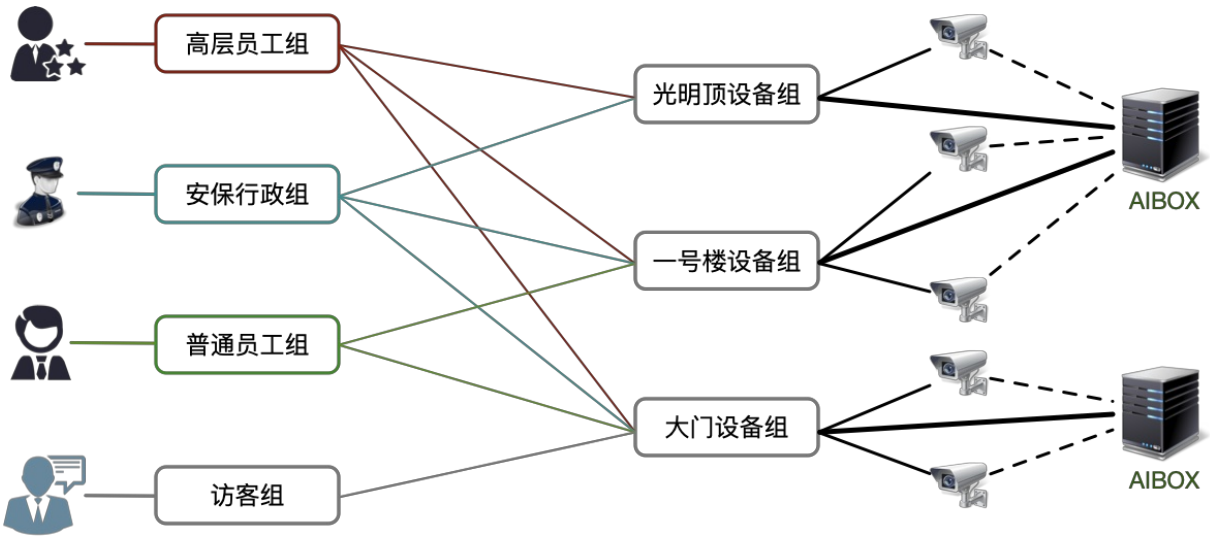


可识别的无权限通行

不可识别的无权限通行

配置

1. 用户和设备之前有实线连接，代表用户在当前设备位置有权限通行；
2. 摄像头绑定到设备组时，会自动根据拓扑关系把关联的AIBOX也绑定到设备组下；
3. 摄像头与组的关系决定云端的判断逻辑，即设备组关联的用户是否有通行权限；
4. AIBOX与组的关系决定权限的下发逻辑，即是否把组相关的用户权限下发给AIBOX；



最佳实践

如果有多个设备组，每个组代表不同的通行权限。又希望所有人都能被识别，可以把AIBOX和所有组关联，目的是把所有人员权限下发给AIBOX。这样，每个摄像头采集的人员都能被识别。

识别后，在云端会根据摄像头与用户的关联判断是否可以通行。

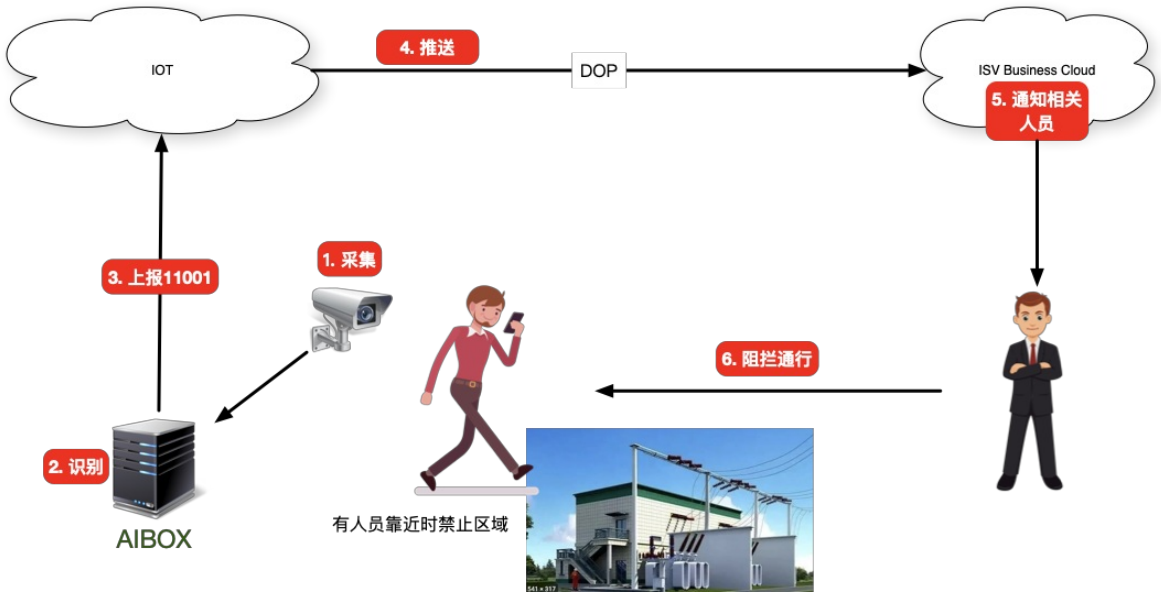
这样，即实现了所有人可识别，又区了分通行权限。

数据

识别到人员通行后，会通过无感通行的数据模型推送数据。见《数据订阅》一节。

1.7.2. 区域闯入

区域闯入是指在摄像头指定区域中，如果有人员进入会向业务方发送通知。业务方可以及时安排人员采集处置措施。



区域闯入不依赖用户配置，现场完成摄像头和AIBOX的安装施工即可。

识别到人员闯入区域后，会通过无感通行的数据模型推送数据。见《数据订阅》一节。

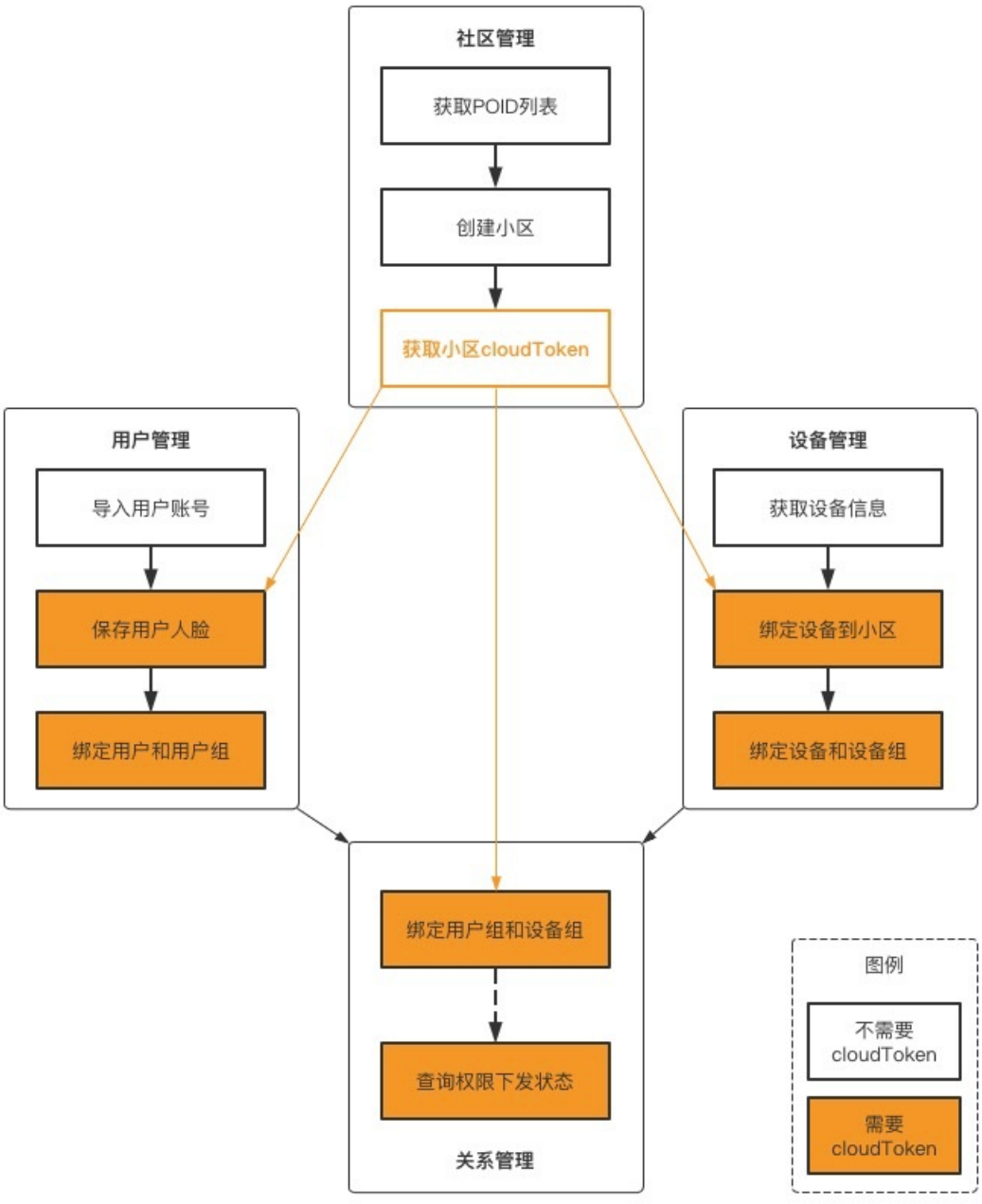
1.8. 最佳实践

1.8.1. 社区人脸门禁

概述

本文以人脸门禁为例，结合阿里云IoT数字社区平台和智能通行服务API 2.0，介绍如何在社区使用场景下，使用一个项目管理多个小区的门禁设备。

接入流程：权限下发



1. 接入准备：请参考数字社区平台接入准备，创建SI项目并获取访问接口所需的appKey和appSecret。

2. 社区管理：创建小区并获取小区的鉴权token，即cloudToken。
   - **cloudToken代表了小区的身份，用于在一个SI项目内实现多个小区的权限认证和数据隔离。**
3. 用户管理：导入用户账号并把用户及其人脸底库图片绑定到小区的用户组内。
   - 用户组是用于管理和组织用户权限的逻辑概念。在本文的使用流程中，用户组是属于小区的。
   - 用户账号没有小区隔离，一个账号可以跨小区使用。
4. 设备管理：把人脸门禁设备绑定到小区的设备组内。
   - 设备组是用于管理和组织设备的逻辑概念。在本文的使用流程中，设备组是属于小区的。
   - 设备有小区隔离，一个设备只能绑定到一个小区。
5. 关系管理：绑定用户组和设备组，实现关联关系的自动下发。

接口说明：社区管理

本节涉及的API，请参考DEMO源代码中的CommunityManager类。参考文档：数字社区平台小区管理

1. 获取POID列表：根据查询关键词搜索高德地图的POI，相当于在高德地图的搜索框内输入文本，搜索地点。
   - 本文的场景下一般输入小区名字作为查询关键词。
2. 创建小区：使用第1步查到的POI相关信息，创建小区，一般情况下取第一条结果即可。
3. 获取cloudToken：小区创建成功并审核通过以后，使用小区ID即可获得代表小区身份的cloudToken，用于后续接口调用。
   - cloudToken有效期一般是2小时，具体以接口返回的expireIn字段为准。
   - cloudToken需要定期刷新或更新，具体请参考API文档。

接口说明：用户管理

本节涉及的API，请参考DEMO源代码中的UserManager类。参考文档：智能通行服务API 2.0

1. 导入账号：把接入方用户账号的唯一标识作为IoT平台的openId，导入到IoT平台。
   - openId的有效范围是SI项目，因此一个openId可以在多个小区中使用，代表了一个账号在多个小区中的权限数据。
2. 保存人脸：保存人脸底库图片，用于下发到设备上，实现人脸识别。
3. 绑定到用户组：绑定用户和用户组。

接口说明：设备管理

本节涉及的API，请参考DEMO源代码中的DeviceManager类。参考文档：智能通行服务API 2.0

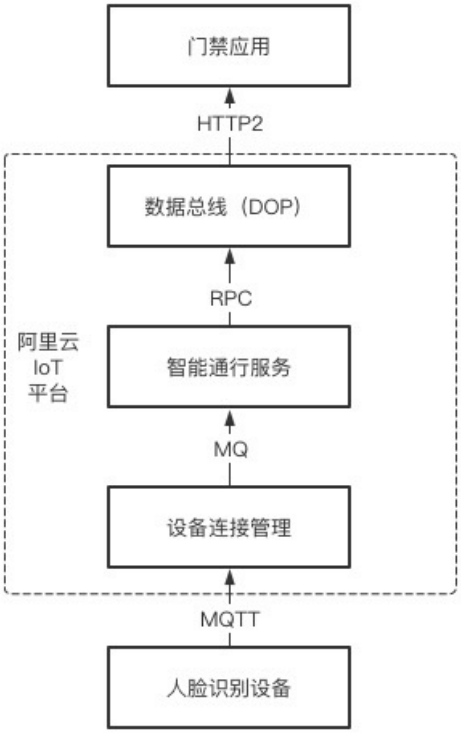
1. 查询设备信息：根据设备三元组中的productKey和deviceName，查询设备的iotId，即设备的唯一标识。
2. 绑定到小区：一个设备只能绑定一个小区。
3. 绑定到设备组：绑定设备和设备组。

接口说明：关系管理

本节涉及的API，请参考DEMO源代码中的RelationManager类。参考文档：智能通行服务API 2.0

1. 绑定用户组和设备组：绑定完成后，人脸和设备就建立了关联关系，人脸会通过云端的下发引擎，自动下发到对应的设备上。
2. 查询下发状态：查询人脸权限的下发状态。

接入流程：数据订阅



当用户通过人脸识别门禁设备刷脸时，设备会通过与云端的连接通道（MQTT）把消息上报到物联网平台，通过平台的数据总线把数据推送给上层应用。关于上层应用如何实时订阅此类消息，请参考DEMO源代码中的DopDataListener类。

DEMO

开源项目建设中，暂请联系阿里技术小二获取DEMO源代码。

如遇接口返回403，请留存返回报文并联系阿里技术小二处理。