ALIBABA CLOUD

阿里云

社区物联网平台

数字社区平台阿里云文档中心合集

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.关于数字社区 13
2.未来社区解决方案 14
2.1. 前提准备 14
2.1.1. 一张图了解准备过程 14
2.1.2. 第一步：钉钉账户注册及开通数字社区 15
2.1.3. 第二步：创建小区，设置小区管理员 17
2.1.4. 第三步：创建支付宝小程序，获取线下张贴二维码 20
2.2. 功能介绍 23
2.2.1. 阿里云账号升级 23
2.2.2. 管理后台 25
2.2.2.1. 小区总览 26
2.2.2.2. 用户信息录入配置 29
2.2.2.3. 居民信息自定义字段 31
2.2.2.4. 居民入驻审批配置 32
2.2.3. 物管服务 33
2.2.3.1. 智能门禁 33
2.2.3.2. 小区管理 36
2.2.3.3. 住户管理 39
2.2.3.4. 权限管理 44
2.2.3.5. 小区公告 47
2.2.3.6. 收款账号 48
2.2.3.7. 访客管理 49
2.2.3.8. 停车管理 52
2.2.3.9. 天猫精灵 57
2.2.3.10. 房屋管理 58
2.2.4. 住户服务 59

2.2.4.1. 我的房屋 60
2.2.4.2. 门禁钥匙 66
2.2.4.3. 小区公告 69
2.2.4.4. 访客邀请 71
2.2.4.5. 天猫精灵 72
2.3. 门禁接入 76
2.3.1. 可视对讲接入 76
2.3.1.1. SIP服务接入规范 76
2.3.1.2. SIP设备端对接说明 80
2.3.2. 智能门禁接入 86
2.3.2.1. 对接说明 86
2.3.2.2. Android设备端开发指南 107
2.4. 车行接入 112
2.4.1. 设备端对接（C语言） 112
2.5. 专题介绍 125
2.5.1. 企业支付宝账号申请 125
2.5.2. 门禁设备SN码管理 125
2.5.3. 社区AI一体机操作手册 127
2.5.4. 小程序不是最新版本的处置方式 157
2.5.5. 房屋认证被驳回后的操作 160
2.5.6. 人员扩展&车位储藏室录入说明 161
2.5.7. 小区快速迁移指引 165
2.5.8. 数字社区体验说明 169
2.5.9. 数字社区小区及设备集成指南 170
2.5.10. 天猫精灵社区服务APK绑定服务 173
2.5.11. 伙伴门禁机集成手册 180
2.6. 相关资料 184
2.6.1. 入驻协议 184

2.6.2. 小区申领凭证 188
2.7. 接口开放 189
2.7.1. 接入准备 189
2.7.2. 通用返回 190
2.7.3. 接口鉴权 191
2.7.4. 小区管理 192
2.7.5. 房屋管理 197
2.7.6. 图片服务 203
2.7.7. 用户管理 207
2.7.8. 居民管理 214
2.7.9. 访客管理 221
2.7.10. 车辆管理 225
2.7.11. 物业员工管理 227
2.7.12. 物业组织管理 233
2.7.13. 设备管理 235
2.7.14. 门禁管理服务 241
2.7.15. AI布防服务 249
2.7.16. 人行管理服务 253
2.7.17. 车行管理服务 255
2.7.18. 公告服务 267
2.7.19. 变更推送及订阅 270
2.7.20. 可视对讲服务 271
2.7.21. 应用权限管理 288
3.物业管理一体机 290
3.1. 什么是物业管理一体机 290
3.2. 功能介绍 290
3.2.1. 人行版一体机 290
3.2.2. 车行版一体机 292

3.2.3. 视频版一体机 294
3.2.4. EBA版一体机 296
3.2.5. 通行版一体机 297
3.3. 实施配置 298
3.3.1. 部署初始化 298
3.3.2. 激活一体机 305
3.3.3. 部署运行组件 312
3.3.3.1. 部署基础组件 312
3.3.3.2. 部署人行设备接入所需组件 317
3.3.3.3. 部署车行设备接入所需组件 320
3.3.3.4. 部署安防设备接入所需组件 322
3.3.3.5. 部署EBA设备接入所需组件 327
3.3.3.6. 部署边缘告警组件 327
3.3.4. 一体机关联项目 329
3.3.5. 设备接入配置 335
3.3.5.1. 接入设备品牌列表 335
3.3.5.2. 通用实施说明 341
3.3.5.3. 人行设备接入 345
3.3.5.3.1. 立林人行设备接入 345
3.3.5.3.2. 宇视人行设备接入 355
3.3.5.3.3. 捷顺人行设备接入 363
3.3.5.3.4. 罗格朗人行设备接入 374
3.3.5.3.5. 立方人行设备接入 378
3.3.5.3.6. 海康人行设备接入 378
3.3.5.3.7. 狄耐克人行设备接入 387
3.3.5.3.8. 大华人行设备接入 397
3.3.5.3.9. 艾特人行设备接入 407
3.3.5.3.10. 冠林人行设备接入 417

3.3.5.3.11. 中证科技人行设备 426
3.3.5.3.12. 太川人行设备 435
3.3.5.3.13. 悉尔人行设备 444
3.3.5.3.14. 通力人行设备 449
3.3.5.3.15. 魔点人行设备 454
3.3.5.3.16. 安居宝人行设备 459
3.3.5.3.17. 霍尼韦尔人行设备 464
3.3.5.3.18. 富士人行设备 469
3.3.5.3.19. 慧锐通人行设备 477
3.3.5.3.20. 京希人行设备 488
3.3.5.3.21. 门禁一体机设备接入 493
3.3.5.4. 车行设备接入 500
3.3.5.4.1. 立方停车系统接入 500
3.3.5.5. 安防设备接入 519
3.3.5.5.1. 安防设备接入 519
3.3.5.5.2. 广拓周界报警主机设备接入 557
3.3.5.5.3. 豪恩周界报警主机设备接入 562
3.3.5.5.4. 精华隆周界报警主机设备接入 567
3.3.5.5.5. 安防设备接入-三实例版 572
3.3.5.6. EBA设备接入 578
3.3.5.6.1. ModbusV2协议接入说明 578
3.3.5.6.2. BACnetV2协议接入说明 587
3.3.5.6.3. ModbusV3协议接入说明 597
3.3.5.6.4. BACnetV3协议接入说明 610
3.3.6. 物业管理一体机账号管理员添加与授权 624
3.3.7. 冷备操作手册 630
3.4. 开放服务 638
3.4.1. 基础对接指引 638

3.4.2. 基础接口 638
3.4.2.1. 应用设备管理服务API 638
3.4.2.2. 项目管理API 673
3.4.2.3. 物模型品类定义 683
3.4.3. 人行接口 689
3.4.3.1. 人员账号服务API 689
3.4.3.2. 传统门禁服务API 697
3.4.3.3. 人脸门禁服务API 721
3.4.3.4. 可视对讲服务API 742
3.4.3.5. 可视对讲移动端SDK 771
3.4.3.5.1. 可视对讲移动端SDK接口说明-[iOS] 771
3.4.3.5.2. 可视对讲移动端SDK接口说明[Andriod] 776
3.4.3.6. 梯控联动服务API 781
3.4.4. 安防接口 796
3.4.4.1. 视频监控服务API 796
3.4.4.2. 智能安防服务API 803
3.4.4.3. 室内安防数据定义 830
3.4.4.4. 周界报警主机数据定义 833
3.4.5. EBA接口 835
3.4.5.1. 设备告警开放服务API 835
3.4.6. 车行接口 858
3.4.6.1. 软硬分离方案 858
3.4.6.1.1. 服务总线开发指引 858
3.4.6.1.2. 边缘文件服务接口 858
3.4.6.1.3. 边缘设备接口 860
3.4.6.2. 系统对接方案 860
3.4.6.2.1. 停车场系统领域服务-服务模型定义 860
3.4.6.2.2. 停车场系统领域服务-数据模型定义 893

3.4.7. 边缘托管应用开发指南 899
3.4.7.1. 应用开发手册 899
3.4.7.2. 边缘应用发布 909
3.4.7.3. 边缘应用运维 916
3.5. 边缘接入 918
3.5.1. 人行标准 918
3.5.1.1. 1 人行方案选型标准 918
3.5.1.2. 2 门禁一体机协议对接方案 922
3.5.1.2.1. 2.1 门禁对讲一体机对接方案介绍 922
3.5.1.2.2. 2.2 门禁一体机标准协议接口定义 936
3.5.1.2.3. 2.3 门禁设备与手机蓝牙对接协议 975
3.5.1.2.4. 2.4 可视对讲设备端SDK接口说明-[Android] 982
3.5.1.2.5. 2.5 可视对讲SIP服务器接入协议 987
3.5.1.2.6. 2.6 人脸算法服务 API 991
3.5.1.3. 3 门禁控制器对接方案 993
3.5.1.3.1. 3.1 门禁控制器对接方案介绍 993
3.5.1.3.2. 3.2 自定义协议驱动开发指导 996
3.5.2. 车行标准 1009
3.5.2.1. 车行道闸对接方案 1009
3.5.2.1.1. 道闸一体机对接方案介绍 1009
3.5.2.1.2. 道闸一体机驱动开发指导 1012
3.5.2.1.3. 边缘文件服务接口说明 1012
3.5.2.2. 停车系统对接方案 1014
3.5.2.2.1. 边缘应用开发手册 1014
3.5.2.2.2. 停车场管理软件对接方案介绍 1028
3.5.2.2.3. 停车场系统领域服务-服务模型定义 1030
3.5.2.2.4. 停车场系统领域模型-数据模型定义 1031
3.5.3. 安防标准 1031

3.5.3.1. 室内安防报警 1031
3.5.3.1.1. 室内安防设备对接边缘服务器接口定义 1031
3.5.3.2. 周界报警 1040
3.5.3.2.1. 周界报警主机对接方案介绍 1040
3.5.3.2.2. 周界报警主机驱动开发指导 1043
3.5.3.3. 监控设备 1043
3.5.3.3.1. 安防设备接入方案 1043
3.5.3.3.2. 安防设备实施手册 1045
3.5.3.3.3. 智能安防服务-边缘事件管理API-Rev0.5 1082
3.5.4. EBA标准 1119
3.5.4.1. EBA设备接入方案介绍 1119
3.6. 常见问题 1122
3.6.1. 网络端常见问题 1122
3.6.1.1. 边缘物业一体机静态IP配置错误 1122
3.6.1.2. 边缘物业一体机网络监测 1123
3.6.1.3. 边缘物业一体机离线排查 1124
3.6.2. 设备端常见问题 1125
3.6.2.1. 单台设备离线排查 1125
3.6.2.2. 设备服务调用错误 1127
3.6.3. 边缘物业一体机常见问题 1127
3.6.3.1. 边缘物业一体机网络配置错误 1128
3.6.3.2. 边缘物业一体机激活失败 1132
3.6.4. 开放服务(API)常见问题 1134
3.6.4.1. 开放API 调用和状态码 1134
3.6.4.2. API 调用返回400 错误码 1134
3.6.4.3. API调用返回403 错误码 1135
3.6.4.4. API调用返回429 错误码 1136
3.6.4.5. API 调用返回460 错误码 1137

3.6.4.6. API 调用返回500 错误码 1137
3.6.4.7. API 调用返回20003 错误码 1138
3.6.4.8. API调用返回20004 错误码 1138
3.6.5. 人行场景常见问题 1139
3.6.5.1. 人脸权限下发状态(NeedManual) 1139
3.6.6. 车行场景常见问题 1140
3.6.6.1. 车场心跳数据缺失 1140
3.7. 发布公告 1143
3.7.1. 2022-01-27 1143
4.边缘智能一体机 1145
4.1. 什么是边缘智能一体机（执行计划版本） 1145
4.2. 激活一体机 1147
4.3. 添加摄像头 1151
4.4. 配置执行任务 1154

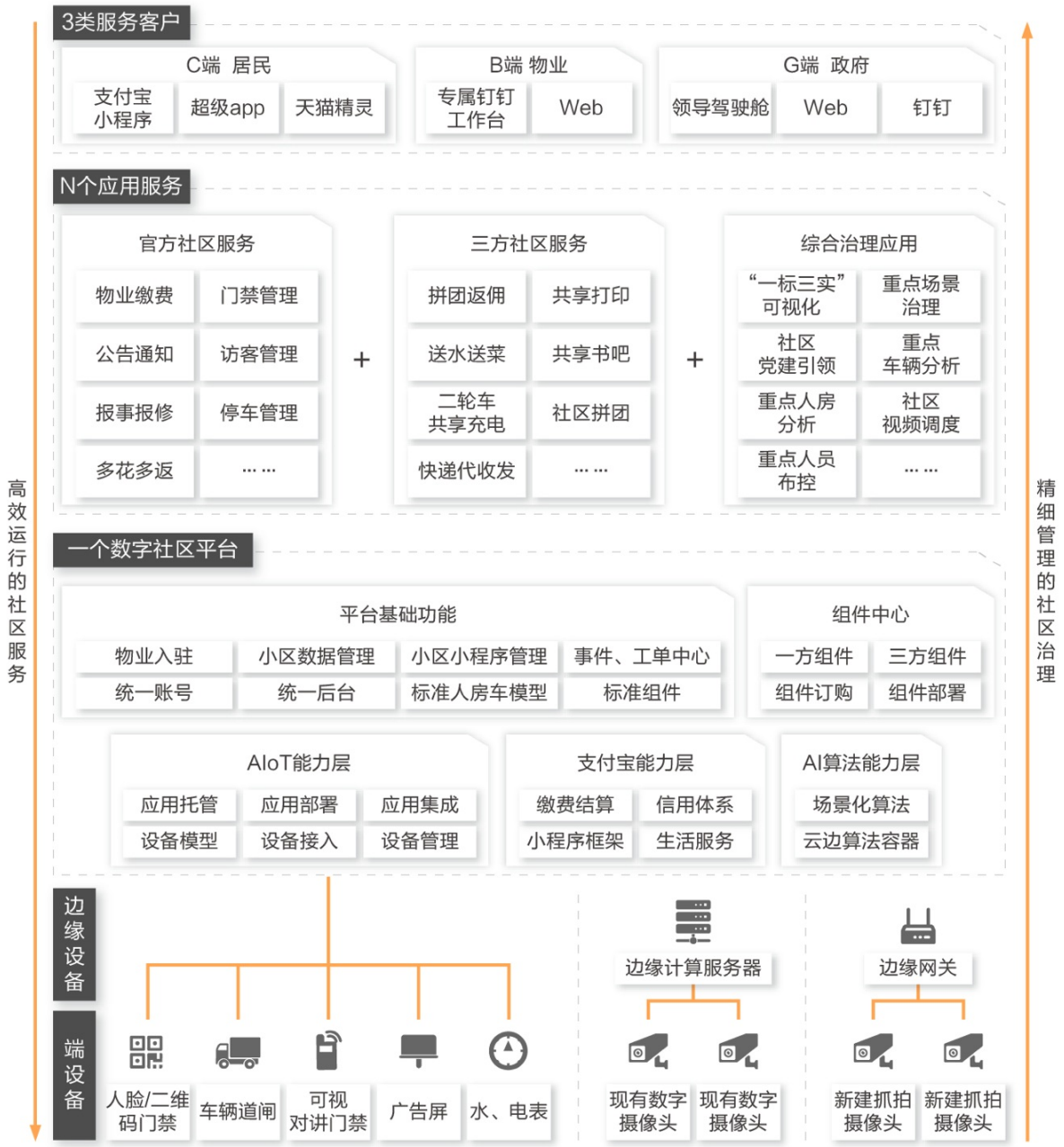
1.关于数字社区

阿里云基于支付宝、钉钉和阿里云AIoT能力搭建了数字社区，为业主提供便民服务，为物业提供持续运营，为伙伴提供生态支持，为城投提供资源整合，为政府提供公共服务和人口管理等全方位解决方案。

快速指引

- 社区SAAS服务平台，端到端社区解决方案，快速进入链接：一张图了解准备过程。
- 边缘应用一体机，社区硬件、应用边缘集成方案，降低边缘集成难度，提升数字化管理能力，快速进入链接：部署初始化。
- 社区智能一体机，社区场景视频AI分析边缘一体机，内部集成7种社区安防、品控场景算法，更加贴合现场的硬件设计，快速进入链接：什么是边缘智能一体机（执行计划版本）。
- 平台开放接口，社区PAAS能力开放服务，聚合硬件能力、算法能力、支撑服务为伙伴提供全面支撑体系，快速进入链接：接入准备。

方案架构



2.未来社区解决方案

2.1. 前提准备

2.1.1. 一张图了解准备过程

> **说明** 本流程作为SAAS服务开通使用，如果您有对SAAS进行二次开发需求，请勿按照此流程开通，可联系您对口的业务同学进行项目化咨询。

您还未安装钉钉APP的一张图准备过程

管理端移动一键领码，让管理更便捷



您已经安装钉钉APP的一张图准备过程

社区疫情防控小程序 - 小区物业手机扫码一键开通，构建社区疫情服务阵地

免费申请零成本一键开通



- 第一步：钉钉账户注册及开通数字社区
- 第二步：创建小区，设置小区管理员
- 第三步：创建支付宝小程序，获取线下张贴二维码

2.1.2. 第一步：钉钉账户注册及开通数字社区

您在数字社区创建出入登记和健康打卡系统的小程序前，需要先下载并注册钉钉，并完成数字社区服务，然后可申请物业入驻使用（疫情期间开放个人或企业/单位注册，建议以个人身份注册，便于更快的使用服务）。

操作步骤

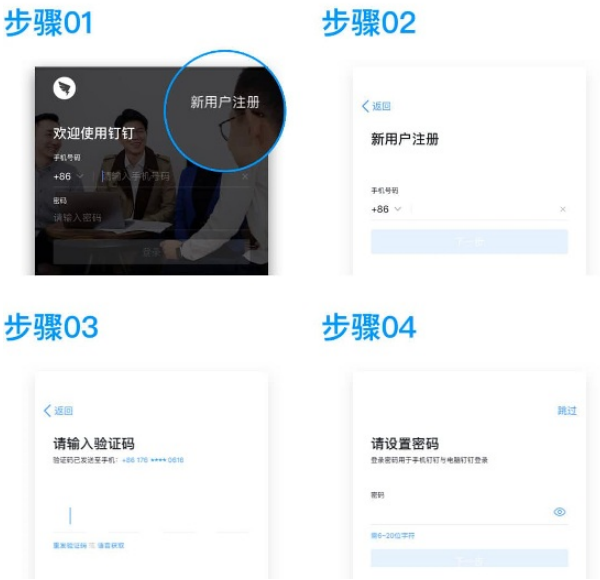
1. 安装钉钉APP。

   面向物业管理的移动端小程序基于钉钉APP，使用前请下载并安装钉钉到手机上，您可以使用手机扫描以下二维码下载钉钉，也可点击以下链接直接下载（下载安卓版，下载iOS版）。



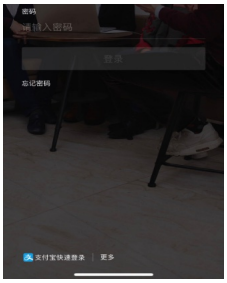
2. 注册账号

   完成下载和安装后，需注册账号，如果您已经拥有一个钉钉账号，可以跳过此步骤，直接进入企业/团队/组织创建流程。



   i. 打开钉钉，点击右上角新用户注册；
   ii. 输入手机号码，点击下一步；
   iii. 输入短信验证码或语音获取；
   iv. 设置登录密码，用于手机和电脑钉钉登录。

   此外，您也可以直接使用支付宝账号授权登录，点击页面下方的“支付宝快速登录”，完成授权后，设置密码，即可完成账号注册。

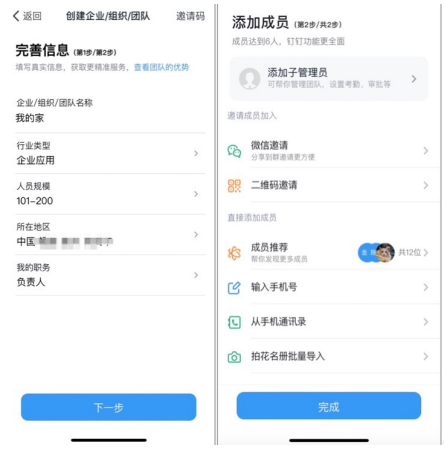


3. 加入/创建团队。

   注册完账号并首次登录后，需要您选择创建或加入团队，您可以为您所在的小区物业公司/小区管理委员会创建一个新的团队，如果您已经有创建好的团队，也可以选择加入团队，直接进入下一步。



   创建团队分为两步：

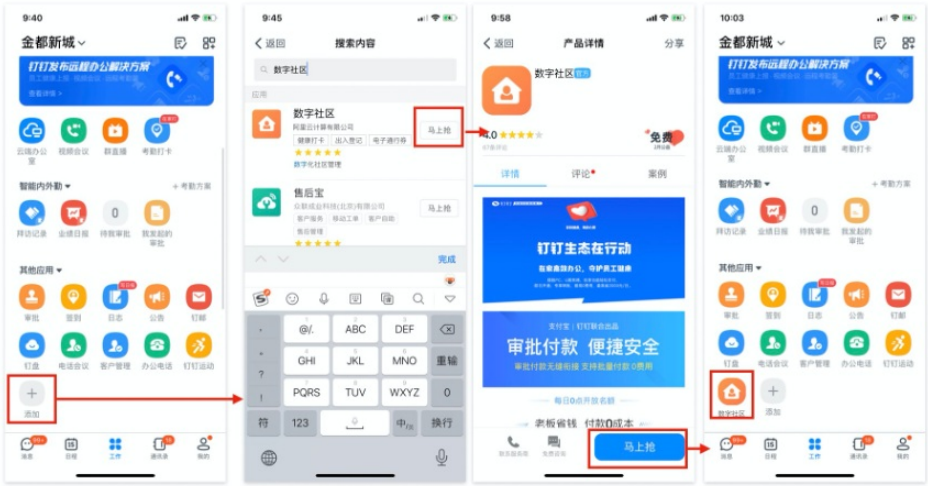


   - 第一步：点击“创建团队”，填写团队信息；
   - 第二步：添加成员，可邀请成员加入，或先跳过添加。

   点击完成后，即可完成团队创建，您可以进入通讯录查看到您所创建的团队。



4. 安装数字社区小程序。



5. 完成数字社区小程序签约。

   首次使用小程序时，如下图，需要录入您的姓名，所在企业名称，企业电话，企业邮箱，并点击提交完成签约操作。

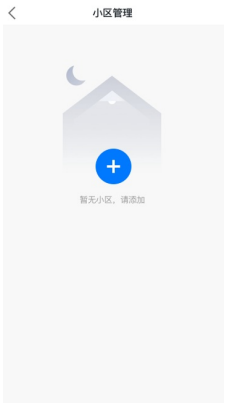


2.1.3. 第二步：创建小区，设置小区管理员

钉钉上安装数字社区后，您需要创建小区及小区基础信息设置。

操作步骤

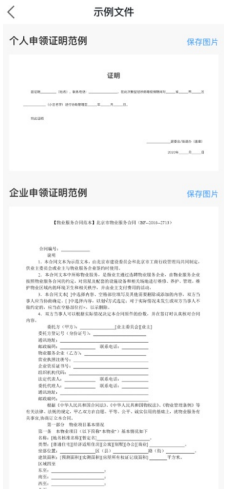
1. 小程序会提示您添加您当前管理的小区，点击添加按钮，进入添加步骤。



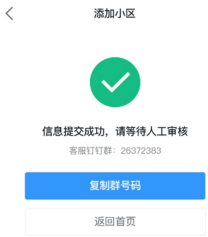
2. 添加小区需要选择城市，和准确的小区名称。并选择物业服务的起始和结束时间，如果是村庄，则需要同样选择城市，搜索准确的村庄名称，如下图所示



3. 此外，您还需要上传物业管理合同，在管证明等凭证的照片，作为人工审核的材料。点击查看范例图片，可以下载模版，打印后，参考模版填写内容并盖章拍照。其他场景的证明材料请查看小区申领凭证



4. 提交信息后，后台人员会对资料进行审核，您可以复制客服群号，添加钉钉群，咨询审核进度，一般24小时内会完成审核。



一家物业公司管理多个小区的，可以返回首页继续添加其他小区。

5. 如果您的小区不在列表中，需要在高德地图APP中添加该小区信息，请点击蓝色的申请小区按钮，在跳转的高德地图界面中，按提示添加小区信息。新添加的小区需要人工审核，还不会立即显示在系统中，请耐心等待。

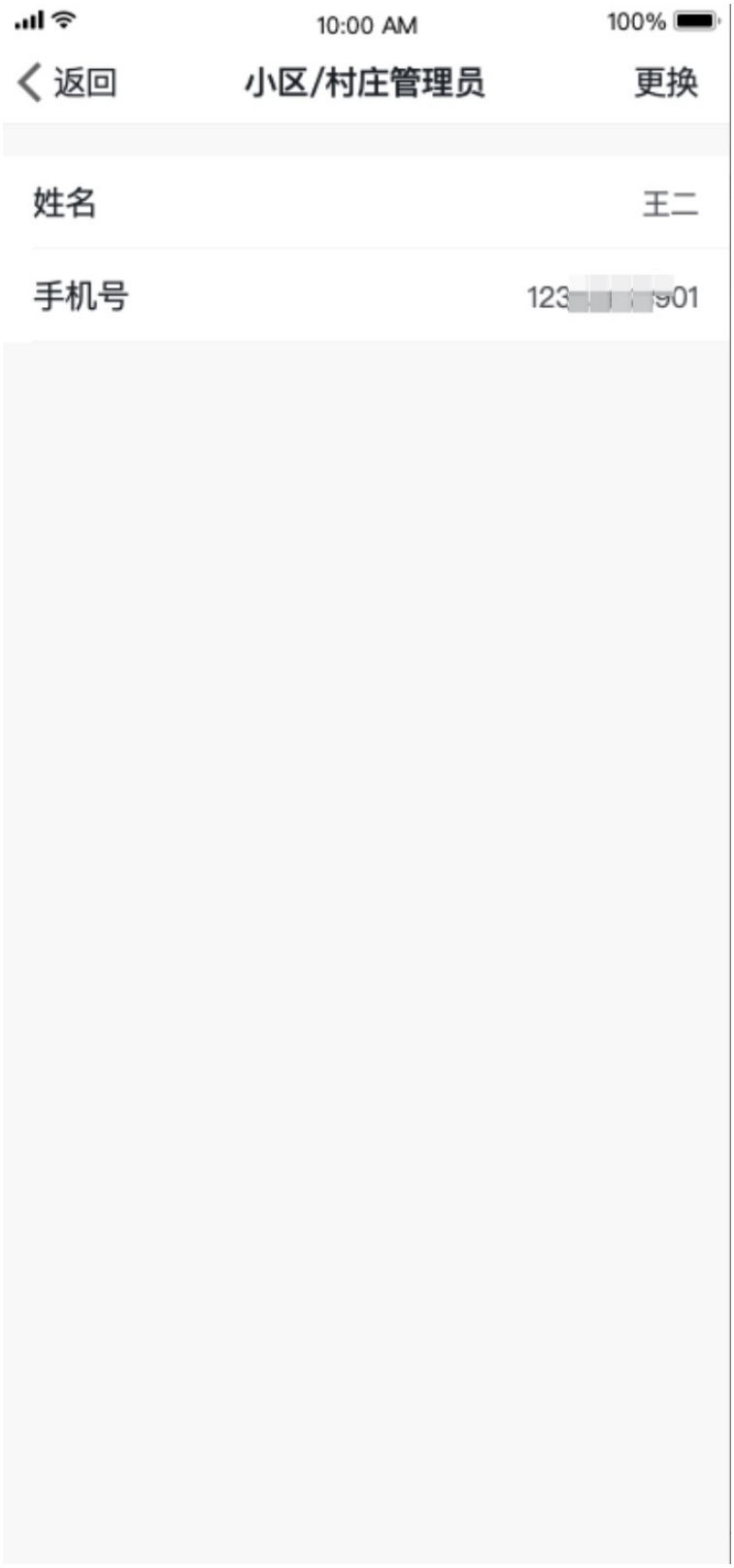




6. 已经添加的小区有三种状态，未通过，审核中，管理中。审核中表示客户正在核验资料，未通过表示资料有问题，您需要点进去重新提交。



7. 如果是管理中的小区，可以点击添加小区管理（即小区物业经理）。



2.1.4. 第三步：创建支付宝小程序，获取线下张贴二维码

您可以通过钉钉账号登录PC管理后台，便于您使用数字社区的功能

操作步骤

1. 打开数字社区应用，找到您已经创建成功的小区，点击“立即创建”支付宝小程序。



2. 如果您当前账号未绑定企业支付宝，需要先完成企业支付宝的绑定工作，点击立即绑定前往支付宝APP完成绑定。如果您当前账号未绑定企业支付宝，需要先完成企业支付宝的绑定工作，点击立即绑定前往支付宝页面登录企业支付宝账号，完成绑定。



3. 绑定支付宝企业账号后，绑定个人淘宝账号进行本小区运营推广负责人备案。



4. 填写小程序的相关信息。注意：小程序的命名、logo图片务必要查看和遵循小程序审核的规范，否则可能导致支付宝审核不通过，建议以小区名称+物业服务/社区服务进行命名。



5. 提交信息后，小程序进入待审核状态，一般在工作日需要2小时左右审核。审核完成后，即可获得小区二维码（每个小区拥有一个独立支付宝小程序）。点击保存二维码打印后，可张贴于小区进出口处，引导业主使用支付宝扫码进入小程序。



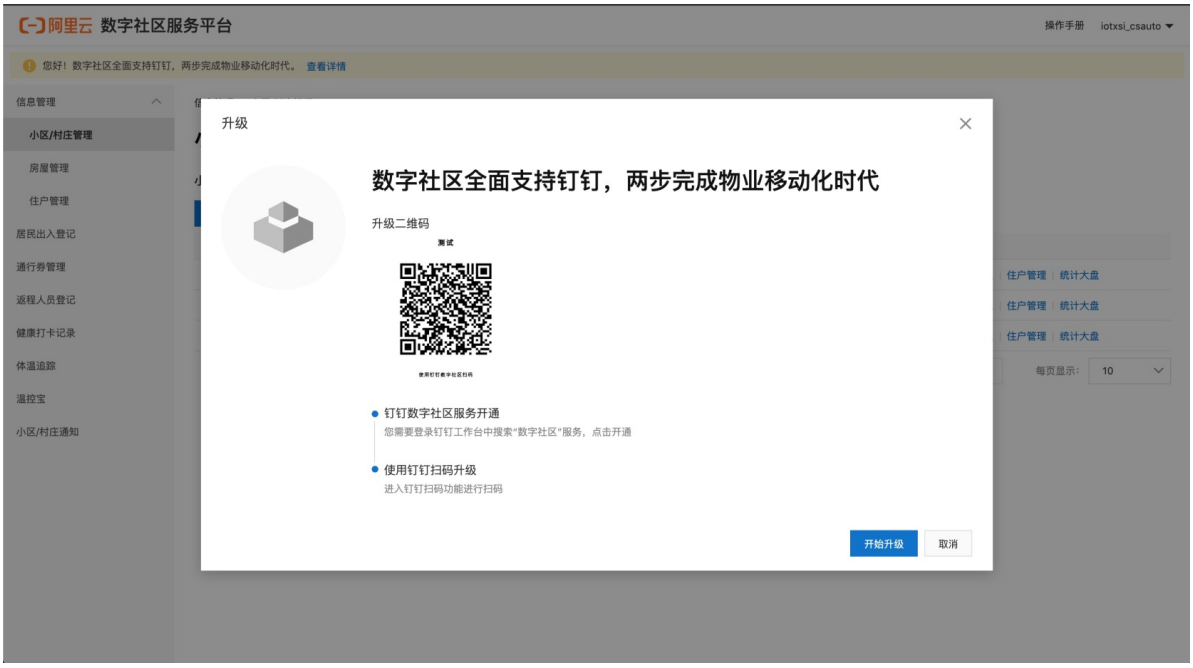
2.2. 功能介绍

2.2.1. 阿里云账号升级

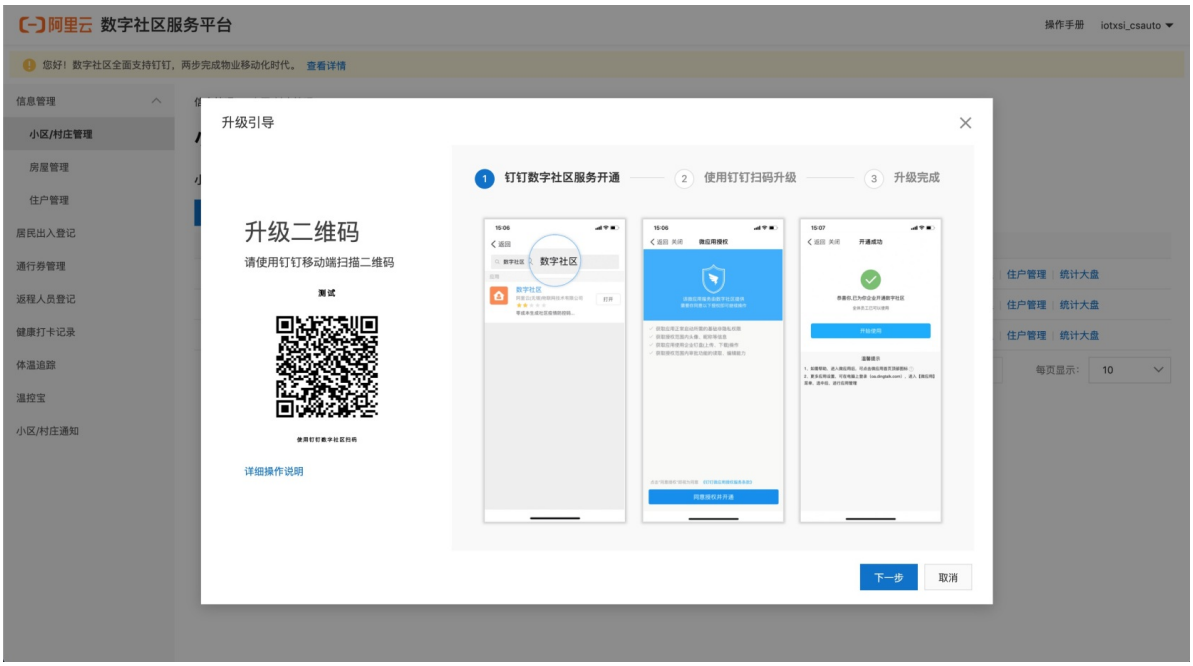
该功能是针对已经在https://community.aliplus.com/introduction数字社区中通过阿里云账号入驻的用户使用，通过该功能可以将原有阿里云账户申领的小区迁移至钉钉工作台

操作步骤

1. 登录数字社区PC控制台，点击“开始升级”进入下一步

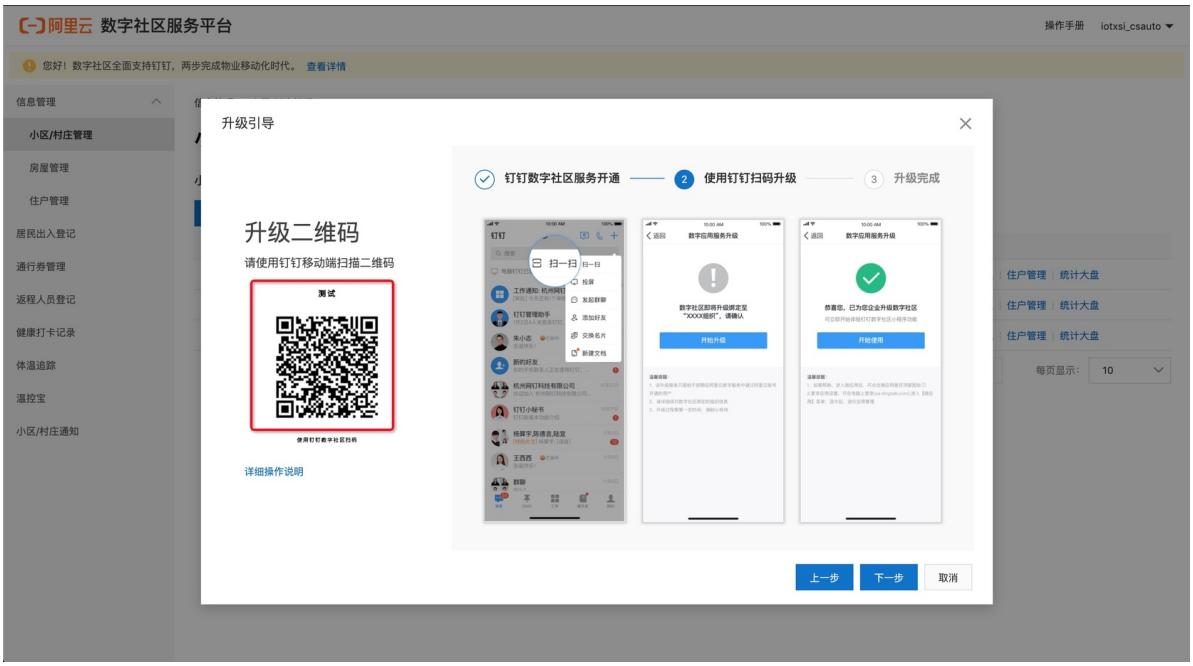


2. 通过钉钉管理员账号在钉钉工作台中搜索“数字社区”，并按照提示流程进行开通

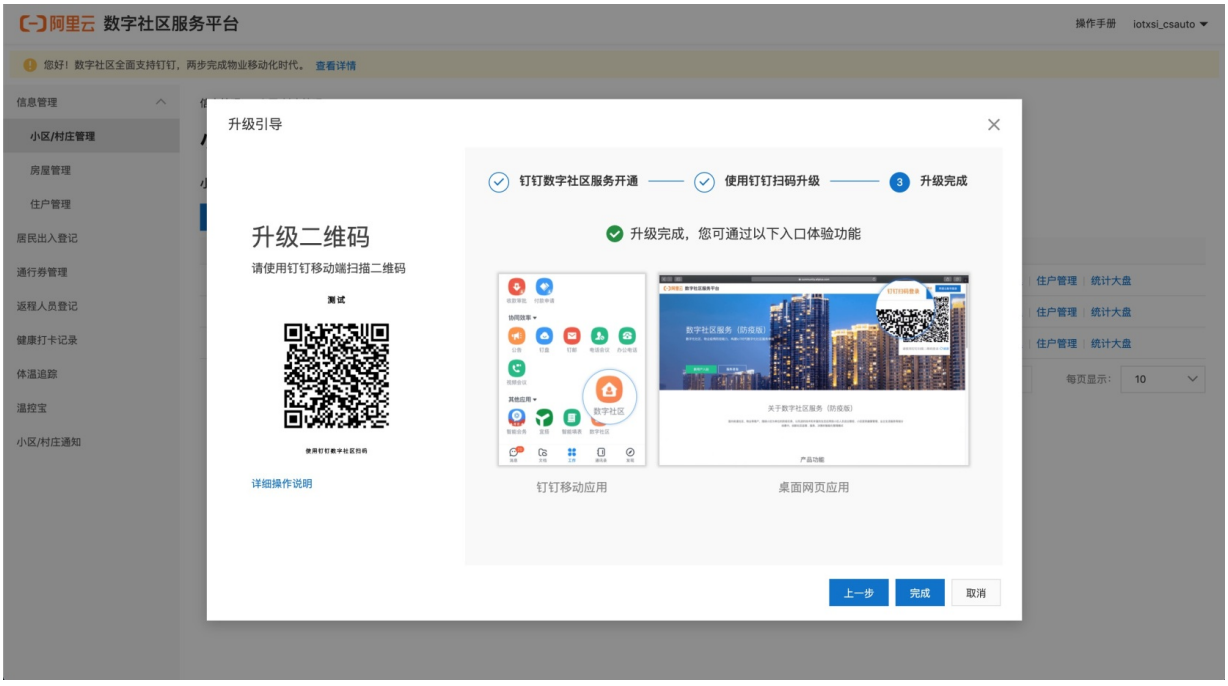




3. 点击数字社区PC控制台中的下一步，钉钉管理员打开钉钉扫码左侧升级二维码进行升级操作，操作时请明确钉钉数字社区服务关联的组织



4. 升级成功后您可以通过钉钉工作台中的“数字社区”和数字社区控制台的钉钉登录进行登录



2.2.2. 管理后台

> 说明
> 数字社区管理后台是专注物业、小区运营方管理物业服务中的配置管理工作，原有物业相关服务已迁移至钉钉数字社区服务中

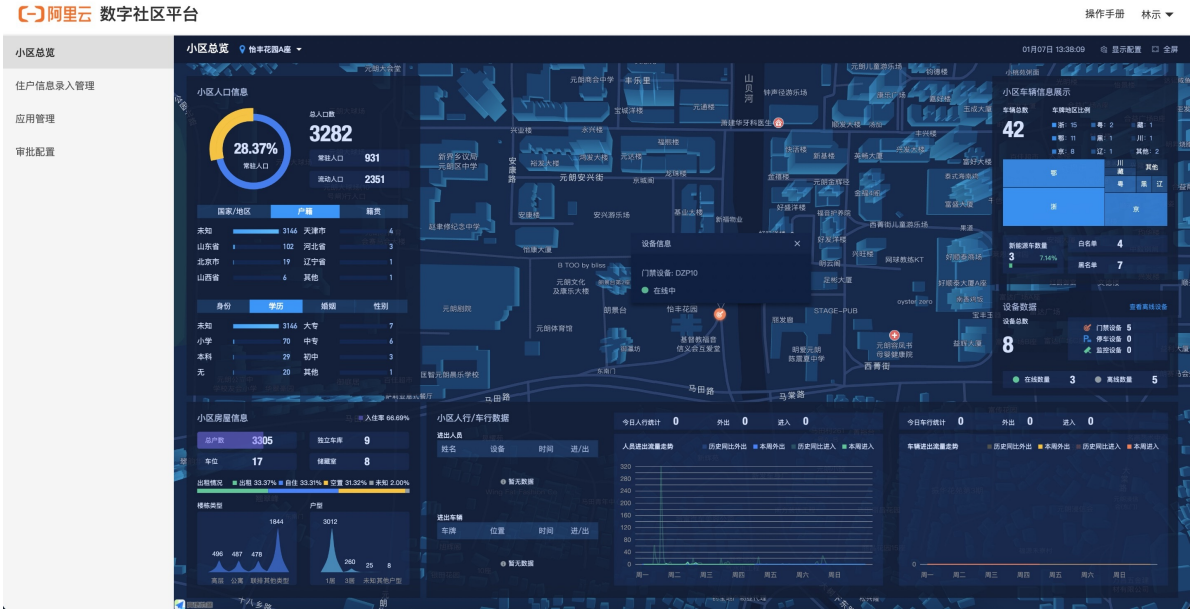
请登录数字社区WEB版首页，在右侧通过钉钉扫码或阿里云账号登录



2.2.2.1. 小区总览

本章将为您介绍小区数字化信息总览展示的数据和功能

1. 登录数字社区web端->小区总览

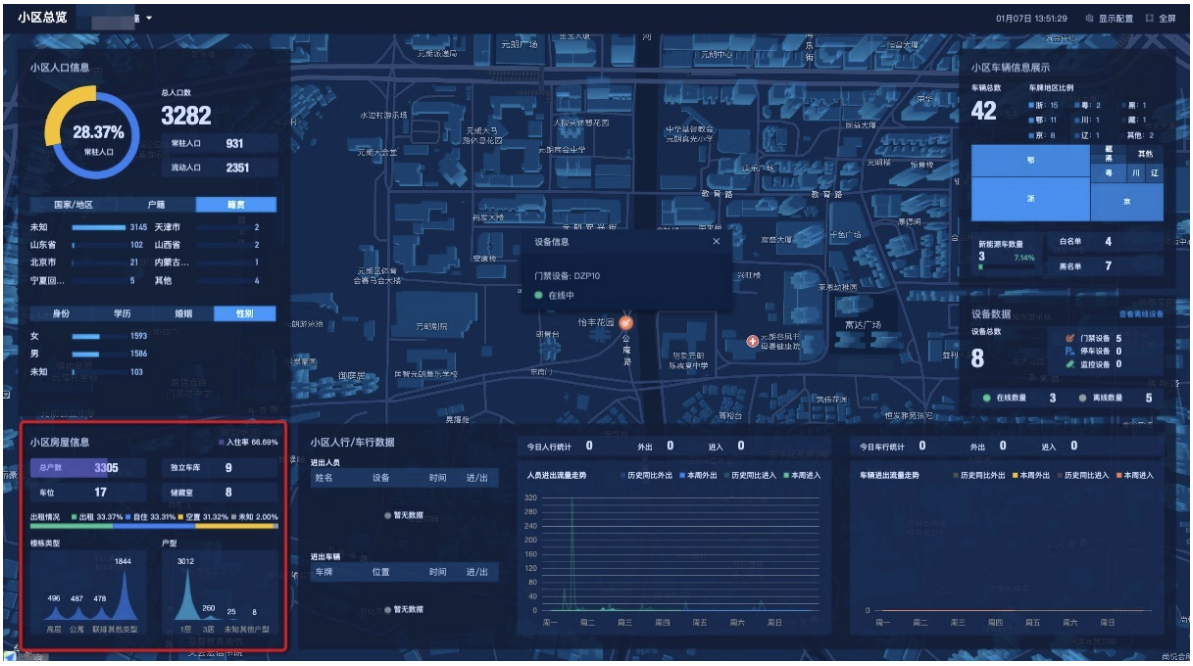


2. 小区人口信息模块，核心展示小区人口信息
   i. 总人口数量，所有通过居民端或物管端登记的用户总量
   ii. 常驻人口数量，房屋中的家庭成员总数
   iii. 流动人口数量，房屋中的租客成员总量

iv. 其他关键字段分类数量



3. 小区房屋信息模块，核心展示房屋分类信息，包含户数、入住率、车位、车库、储藏室的数量，以及房屋类型和户型的分类统计数据



4. 人行车行数据统计，使用此功能前，您需要安装门禁或停车道闸，具体门禁或停车道闸的时候请查看智能门禁、停车管理



i. 进出人员、进出车辆信息显示实时人或车通过设备的记录

ii. 今日人行统计、今日车行统计，统计当天人或车通过设备的记录

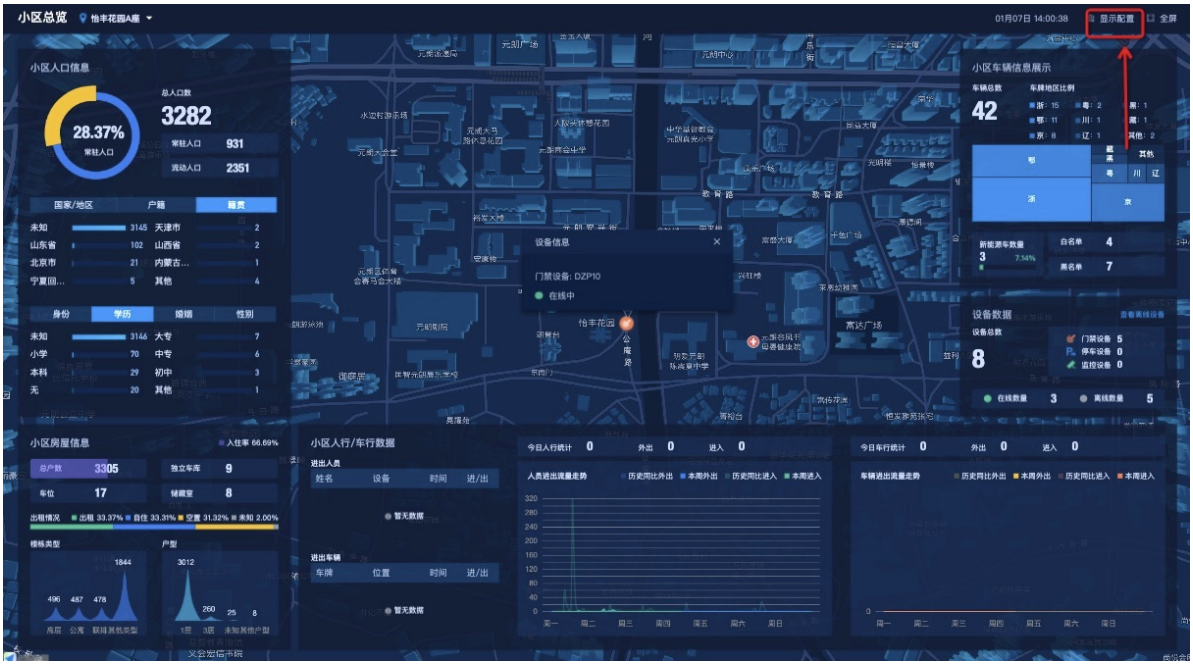
iii. 人员进出流量趋势，本周的进出流量曲线和历史4周流量数据平均值曲线对比

iv. 车辆进出流量趋势，本周的进出流量曲线和历史4周流量数据平均值曲线对比

5. 小区车辆信息展示，通过对系统中录入的车辆信息进行区域分类展示



6. 设置GIS地图，小区总览背后的GIS地图是根据小区申请时确定的地理位置为中心进行展示，您可以在地图上进行小区设备，标注（门禁、停车道闸、摄像头），当设备产生各类告警时可以在GIS地图中进行呈现



i. 选择设备类型并选择具体设备后，会在GIS地图中出现对应图标，随后您可以拖动该图标，进行设备位置的设置



ii. 屏幕最右侧“全屏”按钮后，可以将小区总览覆盖全屏，可以投射到屏幕中做大屏展示

2.2.2.2. 用户信息录入配置

本章将会为您介绍如何配置住户入驻时填写信息的配置能力，您可以根据您所在小区的管理要求选择住户需要录入的信息内容。

1. 登录数字社区web端->住户信息录入管理



选择您需要管理的小区

2. 选择对应字段的是否可见、是否必填以及数据类型
   i. 是否可见，指这个字段是否需要出现在住户录入信息的页面中
   ii. 是否必填，指这个字段是否在住户录入信息时是必选字段
   iii. 数据类型
      a. 主数据，指住户录入信息页面中直接可查看的字段内容



      b. 扩展数据，指住户录入信息页面中点击“”后可查看的字段内容



   iv. 排序，指字段出现在用户入驻页面中的顺序
3. 配置完成后可在小程序中查看住户信息录入的展示



> **说明**
>
> 配置限制
>
> - 姓名、身份、手机号三项为不可配置项
> - 必填字段必须为主数据
> - 扩展字段必须为非必填字段

2.2.2.3. 居民信息自定义字段

本章将为您介绍如果自定义居民信息录入字段，用户可以根据对小区居民信息采集的不同管理诉求进行设置

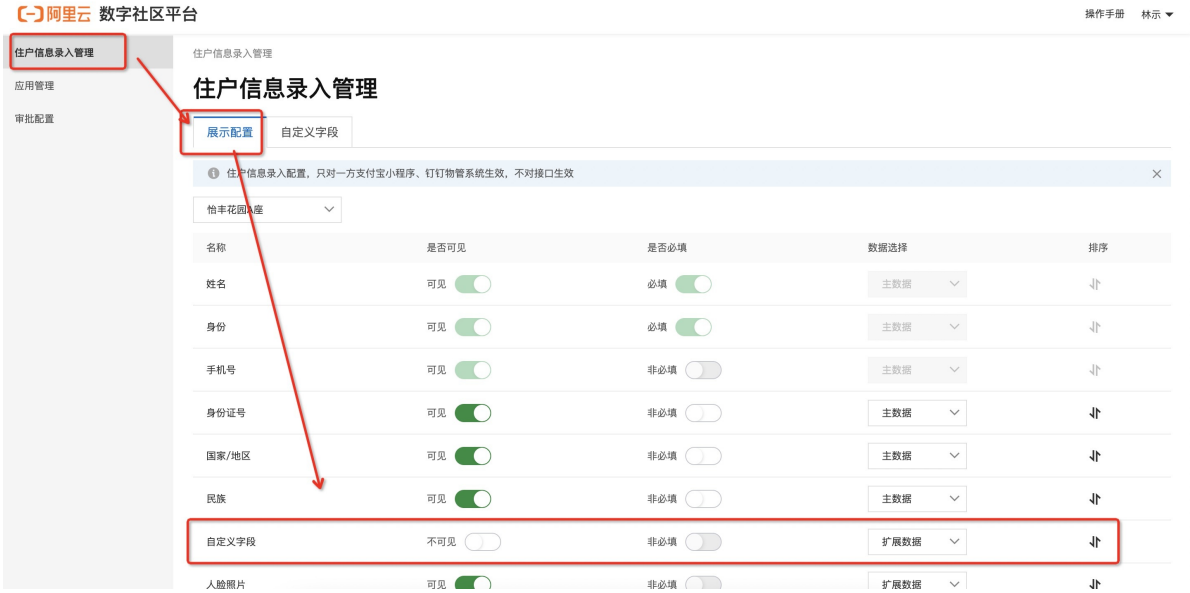
1. 进入数字社区->住户信息录入管理->自定义字段tab



2. 点击新增字段即可新增用户自定义的字段信息
    i. 中文名：该名称将显示在住户入驻信息填写中出现的中文名称
    ii. 英文名：该名称将作为系统存储在数据库中的英文名称
    iii. 类型：目前支持字符型和枚举型两类
        a. 字符型：指该录入内容是无圈定范围的，用户可以根据实际情况自有填写的，当您选择字符时，您需要为这个字段定义一个最大的字符长度
        b. 枚举型：指该录入内容是有圈定范围的，用户只能在提供的选择范围内进行选择填写的，当您选择枚举时，您需要为这个字段定义范围，范围内容包含参数值、参数描述



3. 当您添加完成后，进入数字社区->住户信息录入管理->展示配置，可以在列表中看到刚刚添加的自定义字段，添加的字段默认排列在列表最下方，并且设置成不可见和非必填，您需要根据实际需要对字段进行配置



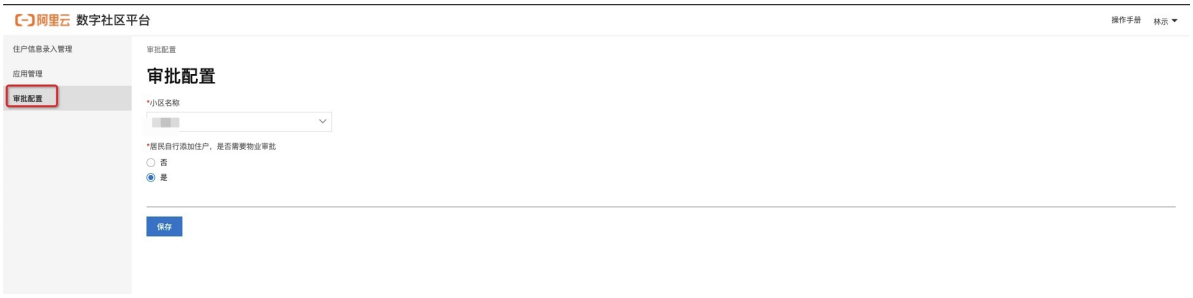
4. 将添加的自定义字段配置为可见、必填、主数据后，可以在住户信息录入页面中看到自定义字段



2.2.2.4. 居民入驻审批配置

本章将会为您介绍，如何让业主自主添加的居民也需要物业审批通过后才允许被添加

1. 登录数字社区管理后台，进入审批配置



2. 选择您需要修改配置的小区
3. 点击 **编辑** 进入编辑状态
4. 选择 **居民自行添加住户，是否需要物业审批** 配置是或否

i. 是，表明由业主在我的房屋中通过手工添加和扫码添加两个路径添加的居民需要被物业审批后才允许添加到房屋



ii. 否，表明由业主在我的房屋中通过手工添加和扫码添加两个路径添加的居民无需审批直接添加到房屋中，系统默认设置为否

2.2.3. 物管服务

2.2.3.1. 智能门禁

您可以通过阅读该篇文章了解数字社区智能门禁的使用方法

门禁配置及管理

1. 进入钉钉数字社区应用，点击 **智能门禁**



2. 选择 **门禁服务商** ，选择门禁品牌和设备配置



3. 选择门禁服务商，目前一个小区只允许一个门禁服务商，不允许设备混用



4. 开启**门禁服务**，再点击**门禁设备**，将设备厂商的设备添加进小区



5. 点击右侧添加门禁，将设备的SN码和别名进行录入
   i. 设备SN码为设备面板或包装上打印的数字
   ii. 设备名称为您对这台设备设置的别名，用于理解设备作用和安装位置
   iii. 通行住户范围为这台门禁机允许哪些楼栋的住户通行，配置完成后，楼栋关联的用户人脸及二维码将下发至设备
   iv. 通话范围为可视对讲设备允许呼叫的单元，
   注意只有设备支持可视对讲时，才允许填写通话范围，目前可视对讲门禁机只支持单元门口机形态，所以只能选择一个单元，当对应单元中用户绑定天猫精灵后即可在门禁设备上输入相应房间号进行对讲呼叫





物业人脸&二维码通行

1. 具备智能门禁使用权限的员工可以在智能门禁服务中进行自己通行的人脸照片管理，或直接使用二维码进行门禁的扫码通行




7:26
返回
关闭
二维码开门
永乐大厦
请将二维码对准门禁，扫码开门

2.2.3.2. 小区管理

您可以通过阅读该篇文章了解数字社区小区管理的使用方法

小区管理使用

1、打开数字社区应用，进入小区管理，钉钉管理员、子管理员点击右上角“添加小区”进行小区的申领。一个企业可以申领多个小区进行管理。




10:00 AM
100%
返回
小区/村庄管理
添加小区
未通过
查看详情
梦想小镇小区
服务期限：2020-12-21 至 2024-7-13
浙江省杭州市西湖区
审核中
查看详情
梦想小镇小区
服务期限：2020-12-21 至 2024-7-13
浙江省杭州市西湖区
管理中
查看详情
梦想小镇小区
服务期限：2020-12-21 至 2024-7-13
浙江省杭州市西湖区
小区管理员
立即添加
支付宝小程序
立即创建
管理中
查看详情
梦想小镇小区
服务期限：2020-12-21 至 2024-7-13
浙江省杭州市西湖区
小区管理员
王二
支付宝小程序
审核中


10:00 AM
100%
返回
添加地点
选择小区/村庄
地点类型
小区
名称
请输入小区名称搜索添加
未找到您服务的小区？可在高德 App 申请新增小区
服务期限
起始日期
结束日期
凭证上传 查看范例图片
*上传照片
请提供相应的证明材料（如管理合同、在管证明等）
提交

2、小区详细信息查询，点击“查看详情”可以查询该小区的详细信息



小区二维码：点击查看小区二维码，点击保存后下载打印粘贴后，小区居民可以使用支付宝扫码进入小程序



1. 小区房屋、住户统计信息
2. 小区管理员、物业电话、服务有效期等信息
3. 支付宝小程序详情查看

2.2.3.3. 住户管理

您可以通过阅读该篇文章了解数字社区住户管理的使用方法

住户管理使用

添加住户

1、进入住户管理点击右上角“添加住户”进入，首先选择居住的房屋，然后再输入住户信息并确认添加。


中国电信
16:25
66%
返回
住户管理
添加住户
第五区兰苑
全部
待审核
楼栋
单元
房号
妮妮测试1
女
业主本人
数字社区1第五区兰苑12101
陈波
男
业主本人
数字社区1第五区兰苑12101
彦晴
女
业主本人
数字社区1第五区兰苑12101



住户详情

1、选择需要查看的住户，点击进入查看住户详细信息或进行编辑


10:00 AM
100%
返回
住户详情
编辑
选择房屋
房号
探海里B区1幢1单元101
住户信息
身份类型
业主
姓名
测试姓名
性别
男
手机号
12345678901
身份证
3333333333333333
人脸照片
人脸照片用户刷脸开门使用


10:00 AM
100%
返回
住户详情
编辑
选择房屋
房号
探海里B区1幢1单元101
住户信息
身份类型
业主
姓名
测试姓名
性别
男
手机号
12345678901
身份证
3333333333333333
人脸照片
人脸照片用户刷脸开门使用
编辑
编辑信息
删除住户
取消

住户认证信息审核

1、小区居民可以通过支付宝小程序来提交身份认证，在待审核列表中可以查看到待审核的住户身份认证申请，点击查看详细信息并进行审核（通过/拒绝）。





2.2.3.4. 权限管理

权限管理是管理物业员工和物管软件功能权限关系，您可以根据不同员工的工作职责为其分配不同功能的权限，权限分配后，员工即可在数字社区中使用具体功能模块，权限管理功能只对钉钉管理员、钉钉子管理员、小区管理员三者可见。

> 注意　您在数字社区主页中查看不到“权限管理”这个功能模块，请通过以下两种方式来获取权限1) 钉钉管理员将您设置为子管理员2) 钉钉管理员或钉钉子管理员将您设置成指定小区管理员

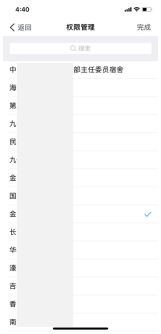


操作说明

1、点击“数字社区”中的“权限管理”进入管理页面



2、点击顶部“>”用于切换管理的不同小区的权限



3、点击顶部“添加权限”添加物业员工权限，添加权限分两部分内容；成员，是您这次添加权限给哪些员工，可以在您所在钉钉组织中选择员工；授权内容，是您给已选择的员工授予功能权限

> 注意　权限说明，使用和管理两个权限是根据功能来决定，不是严格的读写权限，例如小区公告，使用权表明是具备公告查看的权利，管理权表明具备公告编辑发送的权利





4、权限添加后，返回查看小区中已经配置的权限关系

平台提供了两种查看权限的视角，A)人员总览，通过员工维度查看员工具备的权限；B)权限总览，通过功能维度来查看权限授权员工情况。





5、权限修改

- 支持“人员总览”中对员工进行右划操作，进行删除
- 支持“人员总览”中点击员工图标进入功能权限的修改





- 支持“权限总览”中点击功能列进入授权人员列表，对员工权限进行修改



2.2.3.5. 小区公告

您可以通过阅读该篇文章了解钉钉小程序中小区公告的使用方法

在查看该文章时，请保证已经完成账号注册及数字社区服务开通，详情查看一张图了解准备过程

操作说明

1、各级有权限的管理员进入到“数字社区”钉钉小程序中会看到“小区公告”，进入后点击右上角“添加公告”并录入相应内容，点击“保存”即可创建成功。



2、公告创建成功后，点击小区公告列表中“发送”按钮，点击“关联小区”指定发送公告的小区后点击“发布”即可。



2.2.3.6. 收款账号

您可以通过阅读该篇文章了解钉钉小程序中收款账号的使用方法

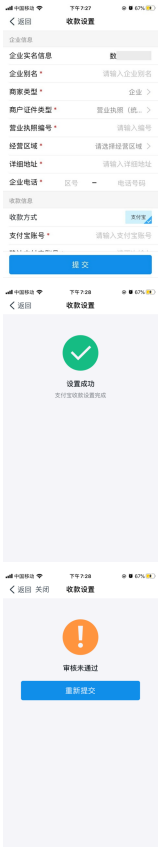
操作说明

1、操作前准备内容

操作前您需要确认，您已经开通企业支付宝账号，如果未开通，请查看右侧文档目录中的专题介绍->企业支付宝账号申请

2、钉钉-收款账号设置

有权限的钉钉管理员进入到“数字社区”钉钉小程序中会看到“收款账号”，点击进入后按要求填写真实信息，全部填写完成后点击提交，一定要按照企业营业执照内容填写真实内容，否则审核会失败，需要重新填写，信息填写完毕后请管理员等待支付宝审核，注意查收短信，收到短信后按提示进行操作，在PC端进行服务签约成功后收款账号才会提示“设置成功”。该账号将可以用于收缴物业费等收款服务。注意：谨慎使用该功能，账号不宜频繁更换。



2.2.3.7. 访客管理

您可以通过阅读该篇文章了解数字社区访客管理的使用方法

背景信息

在查看该文章时，请保证已经完成账号注册及数字社区服务开通，详情查看一张图了解准备过程

访客管理使用

访客登记

访客到访时，通过任一扫码应用扫描访客登记二维码进行访客信息登记。


10:00 AM
100%
返回
访客管理
金地·格林小镇
访客登记
访客记录
请访客扫码登记
刷新
扫码验证
16:26
4G
AA
baidu.com
到访户号
请输入到访户号
来访时间
2020/02/29
来访结束
2020/02/29
访客信息
身份类型
请选择身份
姓名
请输入姓名
手机号
请输入手机号
身份证
请填写身份证号码
添加访客
确认登记

访客记录

查看小区访客到访历史记录，可以按照时间段进行查询。





2.2.3.8. 停车管理

您可以通过阅读该篇文章了解数字社区停车管理的使用方法

停车管理配置

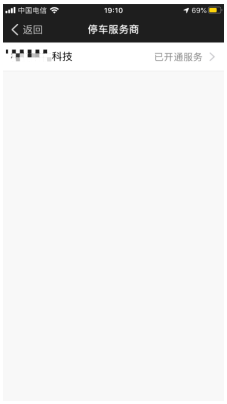
1. 进入钉钉数字社区应用，点击**停车管理**



2. 选择**停车管理**，选择出入口管理添加小区车辆出入口



3. 选择**停车服务商**，目前一个小区只允许一个停车服务商，不允许设备混用



4. 开启停车服务，再点击停车设备，将厂商的设备添加进小区



5. 点击右侧添加设备，将设备SN码、设备别名以及设备位置进行录入

- 设备SN码为设备或包装上打印的数字
- 设备名称为您对这台设备设置的别名，用于理解设备作用

- 设备位置为这台设备所安装的小区具体出入口位置



- 选择车辆权限管理，对小区的车辆进出权限进行配置
  i. 小区通行车辆是对车辆日常进出小区的权限管理，点击右侧添加，选择需要添加的车牌、有效期并添加允许通过小区哪些出入口，配置完成后将车牌下发至设备。



  ii. 白名单车辆为允许通过小区所有的车辆出入口，点击右侧添加，录入车牌号码，配置完成后将白名单车牌下发至设备。



  iii. 黑名单车辆为不允许进入小区的车辆，点击右侧添加，录入车牌号码，配置完成后将黑名单车牌下发至设备。



停车服务

1. 车道监控视频，选择具体需要查看出入口视频监控，点击进入查看。



2. 远程开闸，选择具体需要放行的车道，点击开闸后车辆道闸将会抬杠放行。



通行记录

1. 车辆通行记录，可以查看最多一周的车辆进出记录

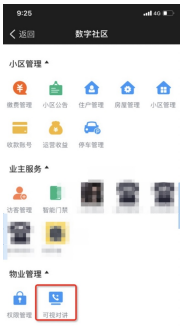


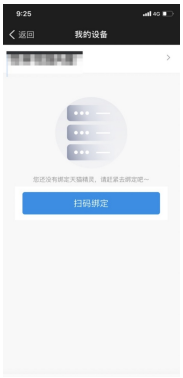
2.2.3.9. 天猫精灵

本章将为您介绍物业如果通过天猫精灵实现物业服务

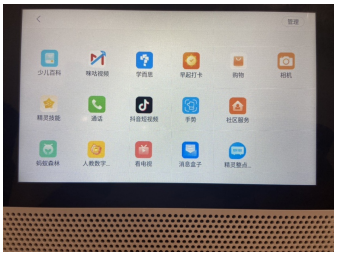
绑定天猫精灵

1. 物业钉钉端进入数字社区应用->可视对讲





2. 在天猫精灵应用列表中找到**社区服务**，点击打开后，首次使用将会展示绑定二维码，使用第一步中打开的**扫码绑定**功能进行扫码



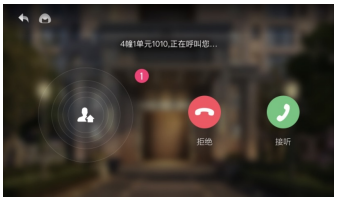
3. 使用钉钉扫码后，如果您有多台猫精设备时，可以点击编辑按钮，对设备进行别名设置



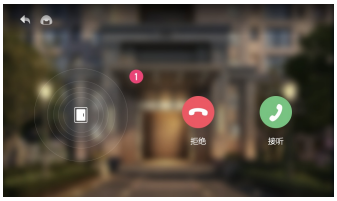
猫灵社区服务

绑定完成后天猫精灵已经成为社区服务终端，您可以在天猫精灵社区服务应用中体验到以下功能

1. 门禁可视对讲，当居民通过室内猫精进行呼叫物业操作时，您的天猫精灵将会通过响铃方式通知您，您可以通过说“天猫精灵，接听”来打开可视对讲，此时您可以和居民进行语音通话



2. 门禁可视对讲，当单元门的门禁点击呼叫物业时，您的天猫精灵将会通过响铃方式通知您，您可以通过说“天猫精灵，接听”来打开可视对讲，并可在通话页面中说“天猫精灵，开门”来打开单元门



3. 进入社区服务应用，您也可以通过更多的语音操作来使用社区服务

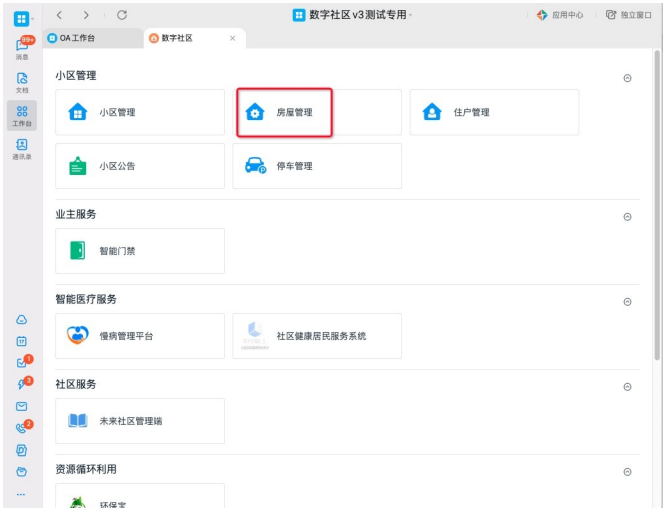


   i. “天猫精灵，开门”，天猫精灵将为您打开小区管理的智能门禁设备，如果小区有多台门禁，那您有一个选择的过程
   ii. “天猫精灵，查看视频”，天猫精灵将为您打智能门禁摄像头视频，如果小区有多台门禁，那您有一个选择的过程
   iii. “天猫精灵，帮我呼叫电梯”，天猫精灵将为您呼叫电梯至指定楼层，如果小区有多套梯控系统，那您有一个选择的过程

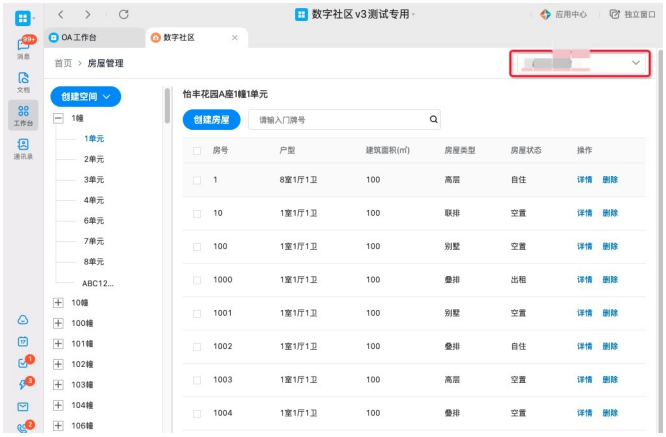
2.2.3.10. 房屋管理

您可以通过阅读该篇文章了解数字社区房屋管理的使用方法

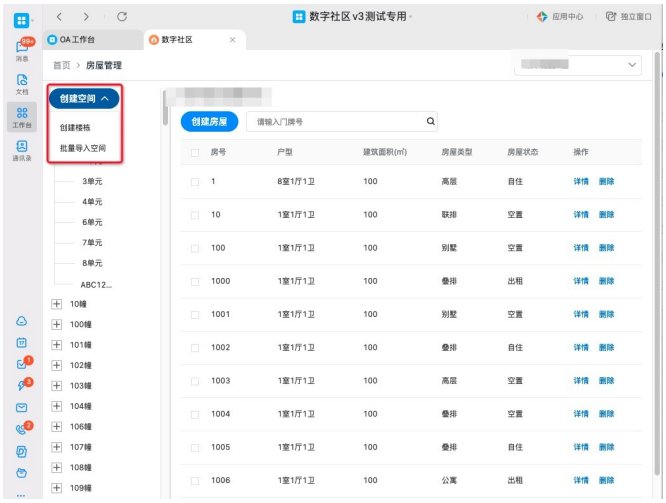
1. 进入钉钉工作台，切换至开通数字社区的组织，并点击**房屋管理**



2. 在房屋管理主界面左上方切换需要管理的小区

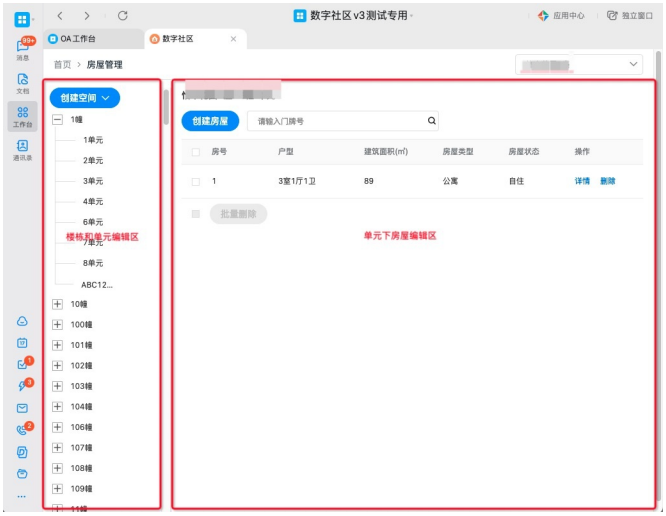


3. 在房屋管理主界面右侧**创建空间**中选择创建方式，小区的空间结构按照**楼栋->单元->房屋**三级进行管理



- 创建楼栋，是将楼栋、单元、房屋三级结构通过操作界面逐一进行填写
- 批量导入空间，是通过房屋模板文件进行批量操作

4. 房屋管理分为两个编辑区域



- 左侧树桩结构编辑区，支持对楼栋和单元的编辑
- 右侧列表结构编辑区，支持对单元下房屋的编辑

2.2.4. 住户服务

2.2.4.1. 我的房屋

您可以通过阅读该篇文章了解支付宝小程序中我的房屋的使用方法

背景信息

在查看该文章时，请保证已经完成账号注册及数字社区服务开通，

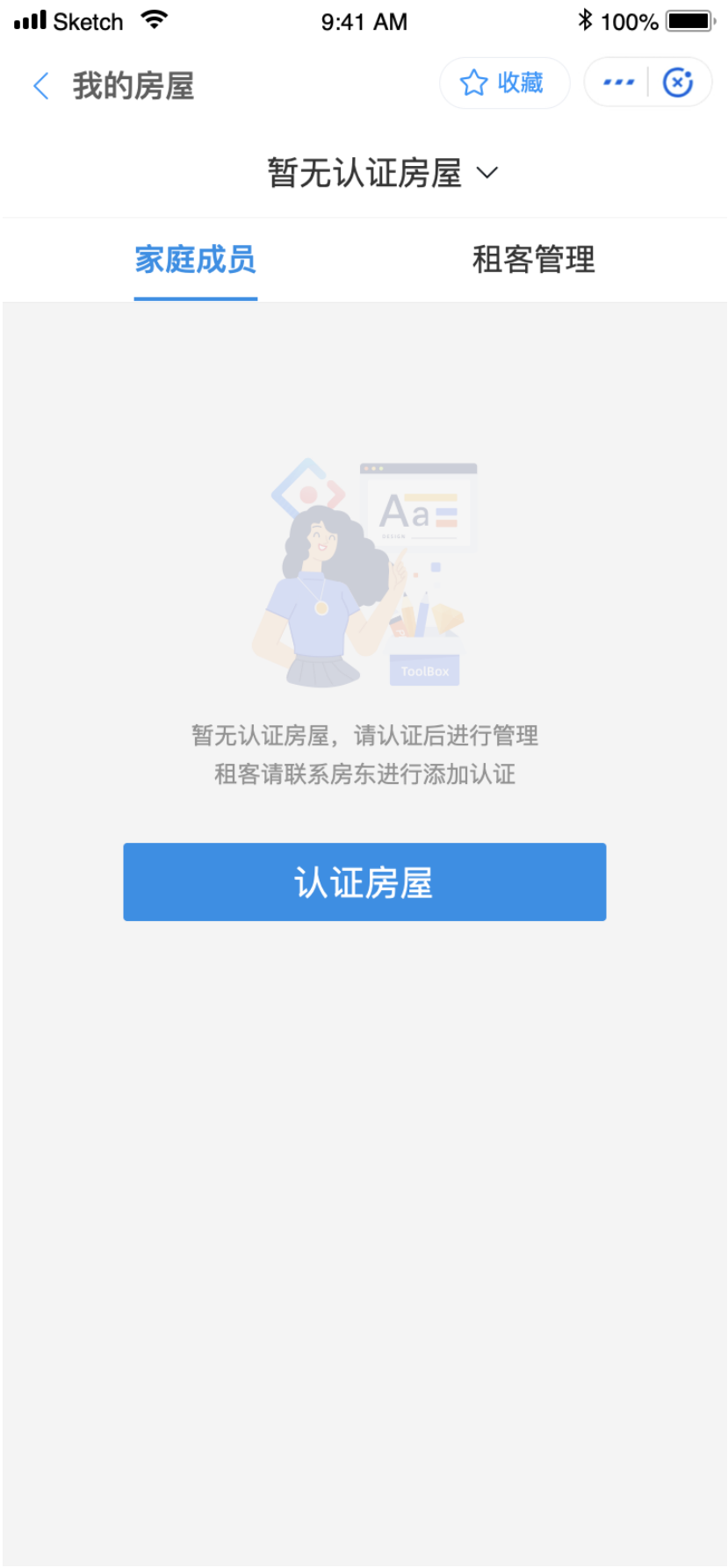
操作指引

- 第一步：钉钉账户注册及开通数字社区
- 第二步：创建小区，设置小区管理员
- 第三步：创建支付宝小程序，获取线下张贴二维码

我的房屋使用

1、小区居民进入本小区支付宝小程序后，点击“我的”>>“我的房屋”进行认证，认证审核通过后即可以使用相关的物业服务。住户认证信息审核


1:20 PM
77%
我的
李明新
134 **** 5703
我的房屋
尚未未认证房屋，点击开始
缴费记录
访客记录
我的订单
购物车
社区生活
我的


Sketch
9:41 AM
100%
我的房屋
收藏
暂无认证房屋
家庭成员
租客管理
暂无认证房屋，请认证后进行管理
租客请联系房东进行添加认证
认证房屋



2、添加家庭成员或租客：可以手动点击“+”添加家庭成员或租客，也可以将二维码保存发送给家庭成员或租客，通过支付宝扫码加入。


Sketch
9:41 AM
100%
我的房屋
收藏
7幢3单元2302
家庭成员
租客管理
李小平
文家强
姜华
顾永宣名...
业主名称的家
小区名称
用支付宝扫二维码，加入家庭
刷新
保存二维码



3、家庭成员或租客信息查询及删除



2.2.4.2. 门禁钥匙

您可以通过阅读该篇文章了解支付宝小程序中门禁钥匙的使用方法

背景信息

在查看该文章时，请保证已经完成账号注册及数字社区服务开通，详情查看一张图了解准备过程

我的门禁卡

根据小区安装的设备类型，使用二维码扫码开门或刷脸开门


Sketch
9:41 AM
100%
门禁钥匙
收藏
林东
天和人家
请将二维码对准门禁，扫码开门
刷新
人脸照片
照片用于刷脸开门，请使用正面照

2.2.4.3. 小区公告

您可以通过阅读该篇文章了解支付宝小程序中小区公告的使用方法

在查看该文章时，请保证已经完成账号注册及数字社区服务开通，详情查看一张图了解准备过程

操作说明

进入某小区支付宝小程序内即可看到中间的小区公告，点击某一条公告直接进入公告详情页，点击右侧箭头进入小区公告列表，可看到所有该小区公告内容。




中国移动
下午 5:39
48%
公告详情
收藏
春季邻里活动
2020-04-09 17:36
万物复苏的季节，欢迎各位邻居一起踏青



2.2.4.4. 访客邀请

您可以通过阅读该篇文章了解支付宝小程序中访客邀请的使用方法

背景信息

在查看该文章时，请保证已经完成账号注册及数字社区服务开通，详情查看一张图了解准备过程

访客邀请使用

1. 新建访客邀请，支付宝小程序中点击“访客邀请”进入填写访客信息，确认提交后访客将收到短信通知。



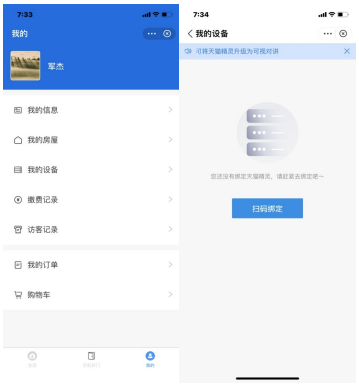
2. 被邀请后将收到短信通知，短信中有访客来访通行凭证，访客二维码。

2.2.4.5. 天猫精灵

本章将介绍如何通过支付宝小程序绑定天猫精灵成为社区家庭室内终端，并使用社区服务功能

绑定天猫精灵

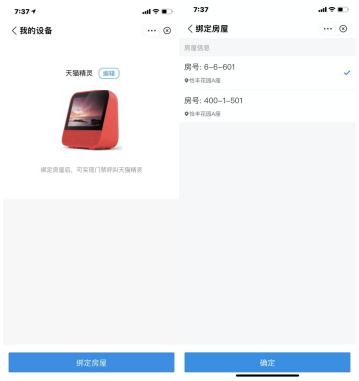
1. 支付宝小程序进入我的->我的设备，您所在的支付宝小程序入口请查看第三步：创建支付宝小程序，获取线下张贴二维码



2. 在天猫精灵应用列表中找到**社区服务**，点击打开后，首次使用将会展示绑定二维码，使用第一步中打开的**扫码绑定**功能进行扫码



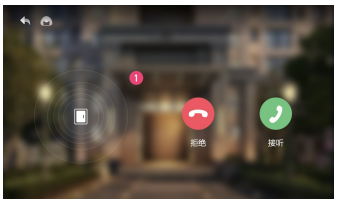
3. 使用小程序扫码后，如果您有多台猫精设备时，可以点击编辑按钮，对设备进行别名设置，再将设备和房屋做关联



猫灵社区服务

绑定完成后天猫精灵已经成为室内社区服务终端，您可以在天猫精灵社区服务应用中体验到一下功能

1. 门禁可视对讲，当单元门的门禁点击您所在房号，您的天猫精灵将会通过响铃方式通知您，您可以通过说“天猫精灵，接听”来打开可视对讲，并可在通话页面中说“天猫精灵，开门”来打开单元门



2. 社区公告推送，当物业在物管服务中通过公告服务推送猫精公告时，天猫精灵将提醒您有新公告，您可以说“天猫精灵，播放”，天猫精灵将为您播放公告内容



3. 进入社区服务应用，您也可以通过更多的语音操作来使用社区服务



i. “天猫精灵，开门”，天猫精灵将为您打开单元门

ii. “天猫精灵，查看视频”，天猫精灵将为您打开门禁摄像头视频



iii. “天猫精灵，呼叫物业”，天猫精灵将为您拨通物业语音



iv. “天猫精灵，帮我呼叫电梯”，天猫精灵将为您呼叫电梯至您的楼层，首次使用呼梯服务，您需要设置您所在楼层可以通过语音收入或者手工输入的方式完成



v. “天猫精灵，播放公告”，天猫精灵将会为您播放当天所有公告内容

vi. 点击设备解绑，可将天猫精灵和支付宝小程序做解绑操作，解绑后设备需要再次绑定后才能使用社区服务

说明

天猫精灵社区服务应用支持的语料

场景	场景说明	支持语料
语音控制单元门禁开门	在可视对讲时或在应用打开的主界面中，语音交互进行打开单元门的语音控制	开门
		请开门
		打开门
		请打开门
		请帮我开门
		接起
		接听
		接听来电

语音进行可视对讲对话	当单元门口机通过对讲呼叫至室内猫精，您可以对猫精进行语音控制，进行可视对讲的接听	接听对讲 接听呼叫 接听电话 接通 接通来电 接通对讲 接通呼叫 接通电话
语音挂断可视对讲对话	当单元门口机通过对讲呼叫至室内或已经在对讲状态下，您可以对猫精进行语音控制，进行可视对讲的挂断	挂断 挂断来电 挂断对讲 挂断呼叫 挂断电话 挂机 结束 结束来电 结束对讲 结束呼叫 结束电话
语音控制开始播报小区公告	应用主界面打开时或者有新消息推送时，可对猫精进行语音对话，让应用开始播报小公告信息	播放 公告 小区公告 播放公告 播放小区公告 读公告 读小区公告 打开公告 打开小区公告

语音控制公告弹窗不播放	当猫精弹出新消息推送页面时，可对猫精进行语音对话，不让应用进行公告的播放	不
		不要
		不听
		不播放
		不要播放
语音控制停止播报小区公告	当猫精开始播放公告时，可以语音控制公告播放停止	停止
		停止播放
		停止公告播放
		退出
		退出播放
		退出公告播放
		关闭
		关闭播放
		关闭公告播放

2.3. 门禁接入

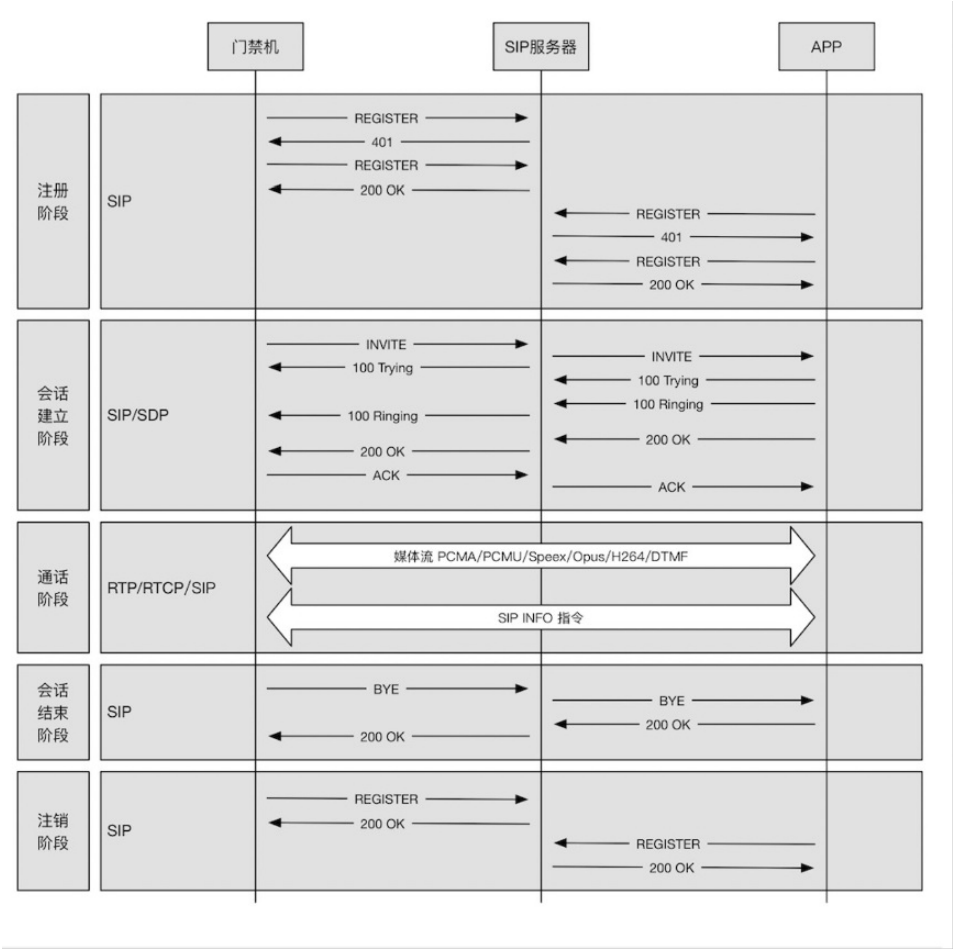
2.3.1. 可视对讲接入

2.3.1.1. SIP服务接入规范

SIP对讲服务接入规范

1. 概述

本文档旨在描述SIP对讲服务接入规范，云对讲采用标准SIP协议，其主要分为注册、会话建立、通话、会话结束、注销等几个阶段。涉及SIP、SDP、RTP、RTCP、PCMA、PCMU、Speex、Opus、H264、DTMF等协议。



2. 协议规范

2.1 SIP

SIP(Session Initiation Protocol)是一个应用层的信令控制协议。用于创建、修改和释放一个或多个参与者的会话。这些会话可以是Internet多媒体会议、IP电话或多媒体分发。会话的参与者可以通过组播（multicast）、网状单播（unicast）或两者的混合体进行通信。SIP协议详见《RFC 3261 SIP: Session InitiationProtocol》。

SIP有很多开源库，如PJSIP、osip2、opensipstack等，可以根据自身平台特性选择。

如下是注册阶段的示例报文：

```
09:57:14.634407 IP 192.168.1.233.62039> 47.112.101.107.6050: UDP, length 605
E..y#...@..<..../pek.W...e..REGISTERsip:47.112.101.107 SIP/2.0
Via: SIP/2.0/UDP192.168.1.233:62039;branch=z9hG4bK.7jiHBl8Sm;rport
From:<sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107>;tag=nPfvQ6pKa
To:sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107
CSeq: 20 REGISTER
Call-ID: 7ay7mqUOym
Max-Forwards: 70
Supported: replaces, outbound
Accept: application/sdp
Accept: text/plain
Accept: application/vnd.gsma.rcs-ft-http+xml
Contact:<sip:TU9e8mJvLJLcSsy9dmDKuc@192.168.1.233:62039;transport=udp>;+sip.instance="<urn:uuid:418ee786-4cf0-435a-b42c-5ab6d06f8b88>"
Expires: 600
User-Agent: Linphone Desktop/4.1.1(belle-sip/1.6.3)


09:57:14.658069 IP 47.112.101.107.6050> 192.168.1.233.62039: UDP, length 525
E..).&..2..2/pek.......W..._SIP/2.0401 Unauthorized
Via: SIP/2.0/UDP192.168.1.233:62039;branch=z9hG4bK.7jiHBl8Sm;received=121.204.13.62;rport=62039
From:<sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107>;tag=nPfvQ6pKa
To:sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107;tag=as4fef1d8e
Call-ID: 7ay7mqUOym
CSeq: 20 REGISTER
Server: JustPBX
Allow: INVITE, ACK, CANCEL, OPTIONS,BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH, MESSAGE
Supported: replaces, timer
WWW-Authenticate: Digest algorithm=MD5,realm="asterisk", nonce="59320d53"
Content-Length: 0


09:57:14.670190 IP 192.168.1.233.62039> 47.112.101.107.6050: UDP, length 786
E....k..@.>...../pek.W......REGISTERsip:47.112.101.107 SIP/2.0
Via: SIP/2.0/UDP192.168.1.233:62039;branch=z9hG4bK.awvGndk3C;rport
From:<sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107>;tag=nPfvQ6pKa
To:sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107
CSeq: 21 REGISTER
Call-ID: 7ay7mqUOym
Max-Forwards: 70
Supported: replaces, outbound
Accept: application/sdp
Accept: text/plain
Accept:application/vnd.gsma.rcs-ft-http+xml
Contact:<sip:TU9e8mJvLJLcSsy9dmDKuc@121.204.13.62:62039;transport=udp>;+sip.instance="<urn:uuid:418ee786-4cf0-435a-b42c-5ab6d06f8b88>"
Expires: 600
User-Agent: Linphone Desktop/4.1.1(belle-sip/1.6.3)
Authorization:Digest realm="asterisk",nonce="59320d53", algorithm=MD5,username="TU9e8mJvLJLcSsy9dmDKuc",uri="sip:47.112.101.107", respon
se="414132ffca95be7131b756d4890b1fc4"


09:57:14.691681 IP 47.112.101.107.6050> 192.168.1.233.62039: UDP, length 575
E..[./..2.../pek.......W.G.gSIP/2.0 200OK
Via: SIP/2.0/UDP192.168.1.233:62039;branch=z9hG4bK.awvGndk3C;received=121.204.13.62;rport=62039
From:<sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107>;tag=nPfvQ6pKa
To: sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107;tag=as4fef1d8e
Call-ID: 7ay7mqUOym
CSeq: 21 REGISTER
Server: JustPBX
Allow: INVITE, ACK, CANCEL, OPTIONS,BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH, MESSAGE
Supported: replaces, timer
Expires: 600
Contact: <sip:TU9e8mJvLJLcSsy9dmDKuc@121.204.13.62:62039;transport=udp>;expires=600
Date: Thu, 16 Jan 2020 01:57:14 GMT
Content-Length: 0
```

2.2 SDP

SDP(Session DescriptionProtocol)是一种会话描述格式 — 它不属于传输协议 — 它只使用不同的适当的传输协议，包括会话通知协议（SAP）、会话初始协议（SIP）、实时流协议（RTSP）、MIME 扩展协议的电子邮件以及超文本传输协议（HTTP）。SDP协议是基于文本的协议，这样就能保证协议的可扩展性比较强，这样就使其具有广泛的应用范围。SDP不支持会话内容或媒体编码的协商，所以在流媒体中只用来描述媒体信息。SDP协议详见《RFC 4566 - SDP: Session Description Protocol》

如下是INVITE携带SDP报文示例：

```
10:04:41.223197 IP 192.168.1.233.62039> 47.112.101.107.6050: UDP, bad length 2114 > 1472
E..... .@.T...../pek.W...J,.INVITEsip:M28PLP9MSvu6w0Vc13HPnC6@47.112.101.107 SIP/2.0
Via: SIP/2.0/UDP192.168.1.233:62039;branch=z9hG4bK.8aOxITrEA;rport
From:<sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107>;tag=7B13LTdJn
To: sip:M28PLP9MSvu6w0Vc13HPnC6@47.112.101.107
CSeq: 20 INVITE
Call-ID: hpgH62gK2G
Max-Forwards: 70
Supported: replaces, outbound
Allow: INVITE, ACK, CANCEL, OPTIONS,BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO, UPDATE
Content-Type: application/sdp
Content-Length: 1449
Contact:<sip:TU9e8mJvLJLcSsy9dmDKuc@121.204.13.62:62039;transport=udp>;+sip.instance="<urn:uuid:418ee786-4cf0-435a-b42c-5ab6d06f8b88>"
User-Agent: Linphone Desktop/4.1.1(belle-sip/1.6.3)

v=0
o=TU9e8mJvLJLcSsy9dmDKuc 2846 1117 INIP4 192.168.1.233
s=Talk
c=IN IP4 192.168.1.233
t=0 0
a=ice-pwd:305b42a8561a2b3f06a158d7
a=ice-ufrag:a60560d1
a=rtcp-xr:rcvr-rtt=all:10000stat-summary=loss,dup,jitt,TTL voip-metrics
m=audio 54480 RTP/AVPF 96 97 98 0 8 10199 100
c=IN IP4 121.204.13.62
a=rtpmap:96 opus/48000/2
a=fmtp:96 useinbandfec=1
a=rtpmap:97 speex/16000
a=fmtp:97 vbr=on
a=rtpmap:98 speex/8000
a=fmtp:98 vbr=on
a=rtpmap:101 telephone-event/48000
a=rtpmap:99 telephone-event/16000
a=rtpmap:100 telephone-event/8000
a=rtcp:51670
a=candidate:1 1 UDP 2130706431192.168.1.233 54480 typ host
a=candidate:1 2 UDP 2130706430192.168.1.233 51670 typ host
a=candidate:2 1 UDP 1694498815121.204.13.62 54480 typ srflx raddr 192.168.1.233 rport 54480
a=candidate:2 2 UDP 169449881
```

2.3 RTP/RTCP

RTP(Real-time TransportProtocol)即实时传输协议，是一个网络传输协议，它是由IETF的多媒体传输工作小组1996年在RFC1889中公布的。RTP协议详细说明了在互联网上传递音频和视频的标准数据包格式。它一开始被设计为一个多播协议，但后来被用在很多单播应用中。RTP协议常用于流媒体系统（配合RTSP协议），视频会议和一键通（Push to Talk）系统（配合H.323或SIP），使它成为IP电话产业的技术基础。RTP协议和RTP控制协议RTCP一起使用，而且它是创建在UDP协议上的。

RTP标准定义了两个子协议，RTP和RTCP。数据传输协议RTP，用于实时传输数据。该协议提供的信息包括：时间戳（用于同步）、序列号（用于丢包和重排序检测）、以及负载格式（用于说明数据的编码格式）。控制协议RTCP，用于QoS反馈和同步媒体流。相对于RTP来说，RTCP所占的带宽非常小，通常只有5%。RTP/RTCP协议详见《RFC3550 - RTP: A Transport Protocol for Real-Time Applications》

2.4 音频

目前支持PCMA、PCMU、Speex 8kHz/16kHz、Opus音频格式

·PCMA/PCMU

PCMA/PCMU协议接入详见《RFC 3551 -RTP Profile for Audio and Video Conferences with Minimal Control》

·Speex

Speex是一个声音编码格式，目标是用于网路电话、线上广播使用的语音编码，基于CELP开发，Speex宣称可以免费使用，以BSD授权条款开放原始码。 Speex的开发者将这个格式视为Vorbis的补充。 Speex是一种有损格式，这意味著使用此格式的音讯，品质将会永久性地降低以减少档案的大小。接入详见《RFC 5574 - RTP Payload Format for the Speex Codec》

Speex支持8kHz和16kHz

·Opus

Opus是一个有损声音编码的格式，由Xiph.Org基金会开发，之后由网际网路工程任务组进行标准化，目标是希望用单一格式包含声音和语音，取代Speex和Vorbis，且适用于网路上低延迟的即时声音传输，标准格式定义于RFC 6716档案。Opus格式是一个开放格式，使用上没有任何专利或限制。接入详见《RFC 7587 - RTP Payload Format for the Opus Speech and Audio Codec》

2.5 视频

目前支持H264视频格式

·H264

H264是国际标准化组织（ISO）和国际电信联盟（ITU）共同提出的继MPEG4之后的新一代数字视频压缩格式。主要目标是：与其它现有的视频编码标准相比，在相同的带宽下提供更加优秀的图象质量。通过该标准，在同等图象质量下的压缩效率比以前的标准（MPEG2）提高了2倍左右。接入详见《RFC3984 - RTP Payload Format for H.264 Video》

2.6 DTMF

DTMF(Dual Tone MultiFrequency)即双音多频，由高频群和低频群组成，高低频群各包含4个频率。一个高频信号和一个低频信号叠加组成一个组合信号，代表一个数字。DTMF信号有16个编码。利用DTMF信令可选择呼叫相应的对讲机。DTMF在对讲中用于发送开门指令（*键）。接入详见《RFC 2833 - RTP Payload for DTMF Digits, Telephony Tones andTelephony Signals》

3. 扩展协议

3.1 开锁协议

为了扩展功能，现在SIP基础上，通过SIP INFO方法（详见RFC 6086 - Session InitiationProtocol (SIP) INFO Method and Package Framework）来发送、接收自定义请求，由于目前SIP采用udp，可能会存在丢包，为了保证成功率，这里采用双包策略，即发送端同时发送2个请求（响应）包，接收端判断3秒内如果是同一个包（根据reqsn），则第二个包不做处理。

请求方发送命令后，在3秒内未收到响应包，则认为请求失败，并返回给上层。请求报文格式如下

·公共头

需设置如下header

Content-Type:application/json

·请求

|参数|类型|必填|描述|

|:--|:--

|req|string|是|请求命令，值为unlock

|reqsn|int|是|请求序列号，1-999范围内循环自增，每次请求加1，双包策略中第二个包reqsn不自增

|sip|string|是|开锁发起方SIP号

示例：

```
{
    "req":"unlock",
    "reqsn":100,
    "sip":"M124123123"
}
```

·响应

|参数|类型|必填|描述|

|:--|:--

|res|string|是|返回请求的命令，值为unlock

|reqsn|int|是|返回请求的reqsn

|status|int|是|状态码，200成功，400以上失败

示例：

```
{
"res":"unlock",
"reqsn":100,
"status":200
}
```

4. 附录

《RFC 3261 - SIP:Session Initiation Protocol》 - https://tools.ietf.org/html/rfc3261

《RFC 4566 - SDP:Session Description Protocol》 - https://tools.ietf.org/html/rfc4566

《RFC 3550 - RTP: ATransport Protocol for Real-Time Applications》 - https://tools.ietf.org/html/rfc3550

《RFC 3551 - RTPProfile for Audio and Video Conferences with Minimal Control》 - https://tools.ietf.org/html/rfc3551

《RFC 5574 - RTPPayload Format for the Speex Codec》 - https://tools.ietf.org/html/rfc5574

《RFC 7587 - RTPPayload Format for the Opus Speech and Audio Codec》 - https://tools.ietf.org/html/rfc7587

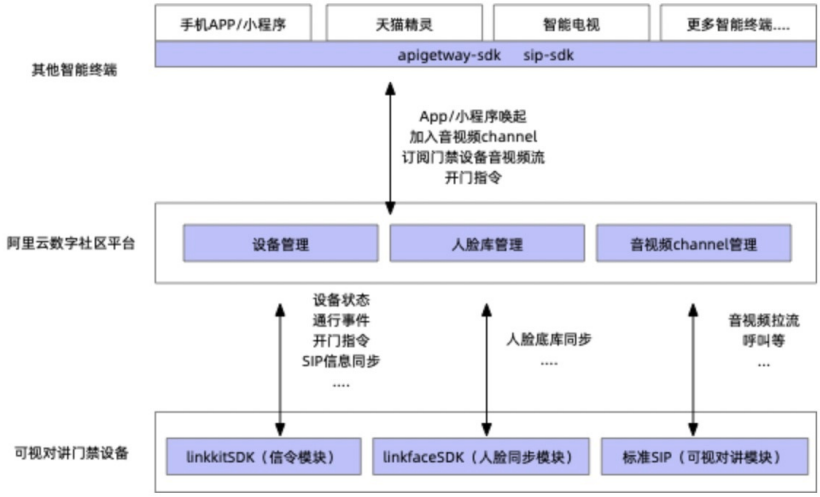
《RFC 3984 - RTPPayload Format for H.264 Video》 - https://tools.ietf.org/html/rfc3984

《RFC 2833 - RTPPayload for DTMF Digits, Telephony Tones and Telephony Signals》 - https://tools.ietf.org/html/rfc2833

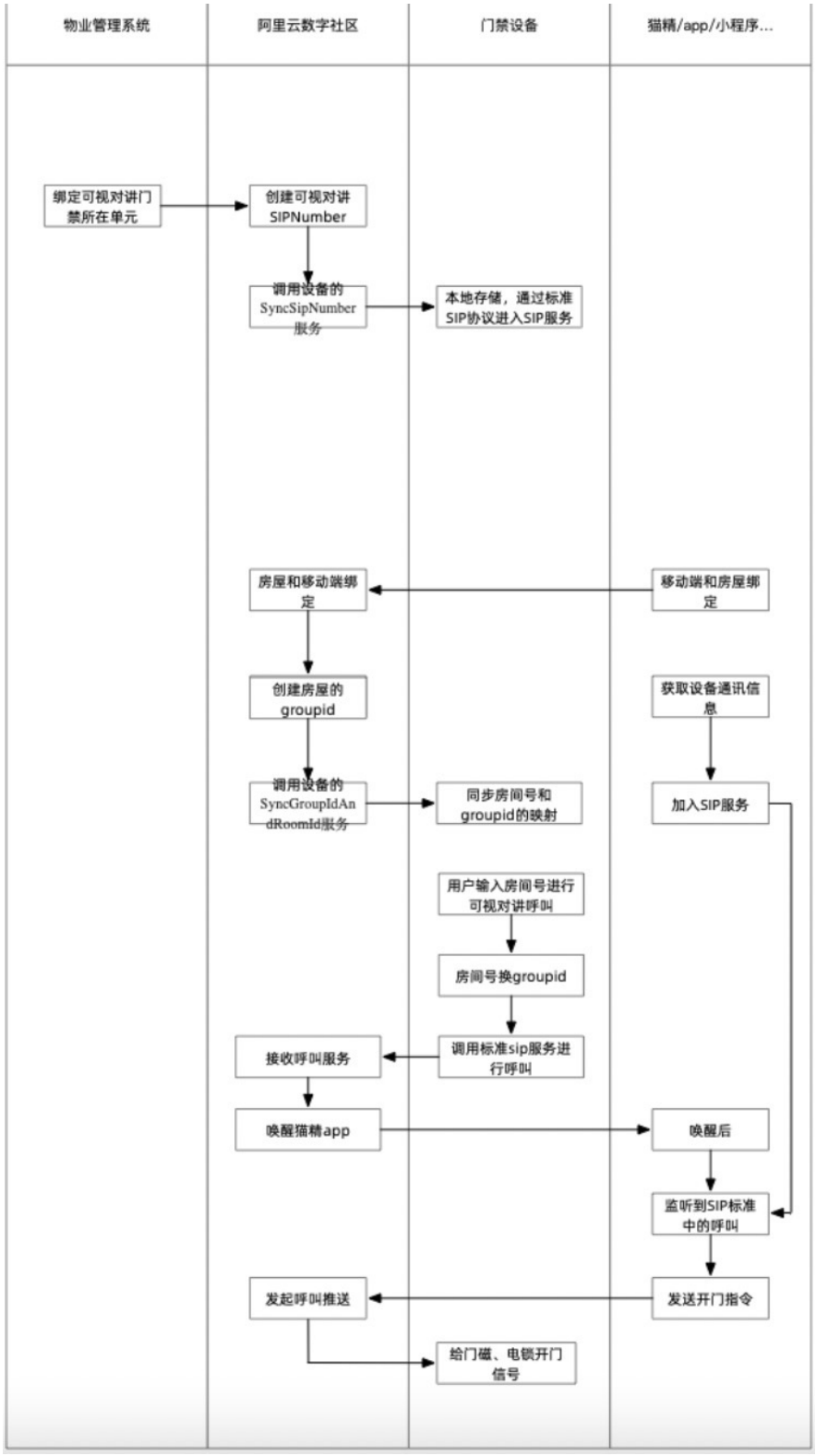
《RFC 6086 - SessionInitiation Protocol (SIP) INFO Method and Package Framework》 - https://tools.ietf.org/html/rfc6086

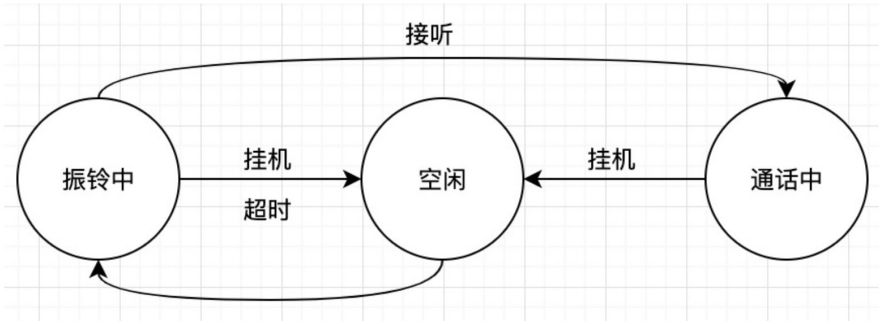
2.3.1.2. SIP设备端对接说明

1. 整体架构


手机APP/小程序
天猫精灵
智能电视
更多智能终端....
apigetway-sdk
sip-sdk
其他智能终端
App/小程序唤起
加入音视频channel
订阅门禁设备音视频流
开门指令
阿里云数字社区平台
设备管理
人脸库管理
音视频channel管理
设备状态
通行事件
开门指令
SIP信息同步
....
人脸底库同步
....
音视频拉流
呼叫等
...
可视对讲门禁设备
linkkitSDK（信令模块）
linkfaceSDK（人脸同步模块）
标准SIP（可视对讲模块）

2. 业务流程


物业管理系统
阿里云数字社区
门禁设备
猫精/app/小程序...
绑定可视对讲门禁所在单元
创建可视对讲SIPNumber
调用设备的SyncSipNumber服务
本地存储，通过标准SIP协议进入SIP服务
房屋和移动端绑定
移动端和房屋绑定
创建房屋的groupid
获取设备通讯信息
调用设备的SyncGroupIdAndRoomId服务
同步房间号和groupid的映射
加入SIP服务
用户输入房间号进行可视对讲呼叫
房间号换groupid
接收呼叫服务
调用标准sip服务进行呼叫
唤醒猫精app
唤醒后
监听到SIP标准中的呼叫
发起呼叫推送
发送开门指令
给门磁、电锁开门信号



3. 可视对讲物模型说明

进行SIP物模型开发前，需要优先对接人脸和二维码门禁服务，具体对接手册请查看https://help.aliyun.com/document_detail/178726.html， 通过这份对接手册，您将明确怎么使用LinkKit SDK中物模型的开发

可视对讲门禁物模型

字段	类型	输入参数	输出参数	参数类型	参数说明
querySipNumber 查询SipNumber	service		sipNumber	String	SIP编号
SyncSipNumber 颁发SipNumber	service	sipNumber		enum	SIP变化
		password		String	密码
		sipServerPort		String	SIP Server端口号
		sipServerAddr		String	SIP Server地址
SyncGroupIdAndRoomId 同步组号和房间号	service	groupRoomMappingFileUrl		String	映射关系文件url详细格式如下：
doorOpenEvent 开门事件	event	sourceType		enum	源类型0-云对讲1-电话开门-1-普通开门在云对讲过程中开门为0，其他为-1
		number		String	呼叫的SIP账号，普通开门时为空
		extInfo		String	扩展信息{ "roomId": "xx", "captureImg":"XXX"，//开门抓图Filekey, 由下文OSS文件上传链路获得}
OssStsTokenRequest StsToken请求上报	event	RequestType		enum	0：图片文件上传1：文本文件上传2：其他类型文件上传
SendOssStsToken 下发OssStsToken	service	Credentials		String	加密的访问凭证加密方式详见开门事件传图一章节
		BucketName		String	OSSbucket名称
		EndPoint		String	OSS的EndPoint名称
		ObjectName		String	OSS存储空间名称

ExtInfo 扩展属性	属性			String	{"intercomDeviceMode":0 //0 单元门口机，1围墙机，默认为单元门口机}

映射关系文件格式示例

```
{"syncGroupRoomList": [{"groupId": "Group-1","roomId": "Room-1"},{"groupId": "Group-2","roomId": "Room-2"},{"groupId": "Group-3","roomId": "Room-3"},{"groupId": "Group-4","roomId": "Room-4"}]}
```

房号下发规则

门禁机存在两种按照模式：一种是安装在单元楼下， 称为**单元门口机**，服务的范围为本单元的房屋。另外一种是安装在小区大门口，称为**围墙机**，服务的范围是全小区，或者小区中的一片区域（多个楼栋）。这两种模式却别由其服务的范围引入：单元门口机服务的是本单元房屋，所以在可视对讲中，输入房号即可定位到目标房屋。围墙机服务的是小区，或者多个楼栋，那么需要输入具体的楼栋-单元-房号，才能定位到房屋。这将引入门禁机上交互的差异，以及在上文中提到下发组号和房间号的差异。

单元门口机下发的是房号，如上文中映射关系文件格式中提到的：groupId->roomId的规则下文重点对围墙机模式进行解释。

围墙机RoomId字段规则定义如下：

${楼幢}-${单元}-${房号}

例如示下：

1-3-1202

10-3A-1709

最长位数：

3位-3位-5位，规则数字+字母

门禁机交互界面：

1.支持输入 ${楼幢}-${单元}-${房号} 的规则

2.设备端需要有“围墙机模式”和“单元门口机模式”

在围墙机模式下，用户输入楼幢、单元、房号的信息

在单元门口机模式模式下，用户只需要输入房号

围墙机模式由物模型中ExtInfo字段进行下发或者设备端上报，具体的格式定义为：

ExtInfo：

```
{"intercomDeviceMode":0 //0 单元门口机，1围墙机，默认为单元门口机}
```

在ExtInfo中增加一个intercomDeviceMode的属性

属性定义如下：

字段名称	类型	说明
intercomDeviceMode	Integer	门禁机工作模式0 单元门口机，1围墙机，默认为单元门口机

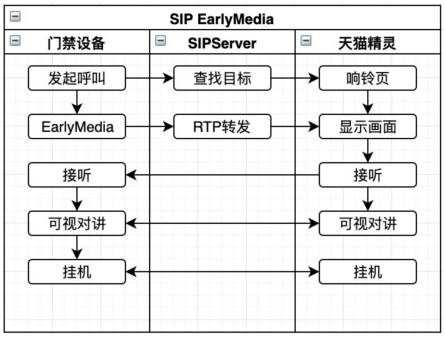
主动查看视频及视频预览

当门禁机发起视频时，室内设备，如猫精，app等需要先行查看门禁视频的内容，再决定是否接听，避免用户与一些无关人员，或者不想会见的人员进行通话。

门禁发起视频门禁

在SIP标准协议中，支持earlymedia（信令：183）模式，在接通前投递音频及视频流。

整体流程如下：

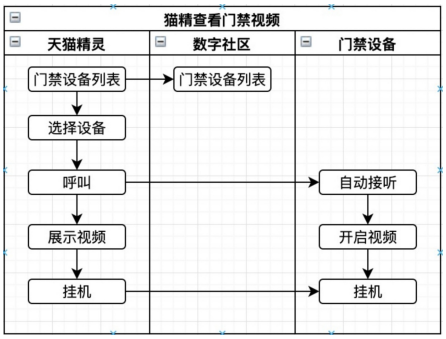


门禁设备端需要支持呼叫时总是发起EarlyMedia呼叫。

- 配置EarlyMedia启用
- 在发起后启动摄像头采集

主动查看门禁视频

用户可以通过猫精端/App主动查看当前楼下门禁机的视频流，应对外卖/快递通过电话呼入的情况，进行二次确认。主动查看门禁视频通过自动接听的方式完成，整体流程如下：



门禁设备端需要支持的点

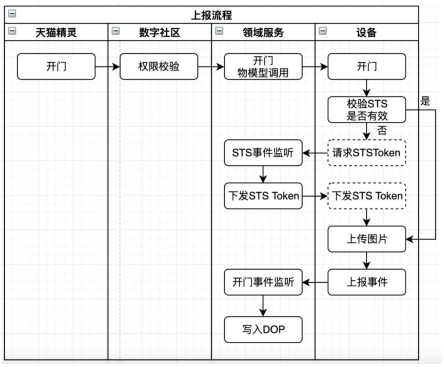
- 配置自动接听
- 静默接听，不会发出响铃提示
- 通话时间60S【支持可配置】

使用限制

- 当门禁设备在呼叫中，无法查看。
- 同时只允许一个猫精进行查看。

开门事件图片上传

当远程开门时，需要进行开门事件图片留档，方便门禁管理方进行回溯，如下是一个典型的开门事件上报以及传图流程：



ststoken使用OssStsTokenRequest来进行请求，请求时指定需要获取的目标类型。

当前支持的目标类型有：

0：图片文件上传

1：文本文件上传

2：其他类型文件上传

下发的SendOssStsToken中Credentials字段是加密后的ststoken

加密方式为 AES加密，使用的秘钥为设备三元组中的DS前16位，加密模式为：AES/ECB/PKCS5Padding

加密前的源文格式：

```
{"securityToken":"XXXX","accessKeySecret":"xxxx","accessKeyId":"xxxxx","expiration":"XXXXX"}
```

4. 对接注意事项

服务端口定义

功能	端口	备注
SIP	6050	SIP协议端口
RTP	10000-20000	多媒体流端口

多媒体格式

多媒体流	格式	备注
音频	PCM	采样率16K 双通道
视频	H.264	480P

SIP服务接入规范

设备的远程开门服务请遵循人脸+二维码门禁对接中的remoteopen服务进行开门。

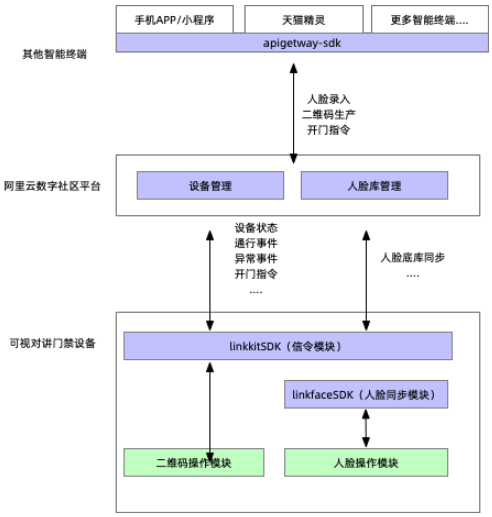
5.使用时长

建议设备端侧控制单次使用时长，1~5分钟，最好UI可配置。

2.3.2. 智能门禁接入

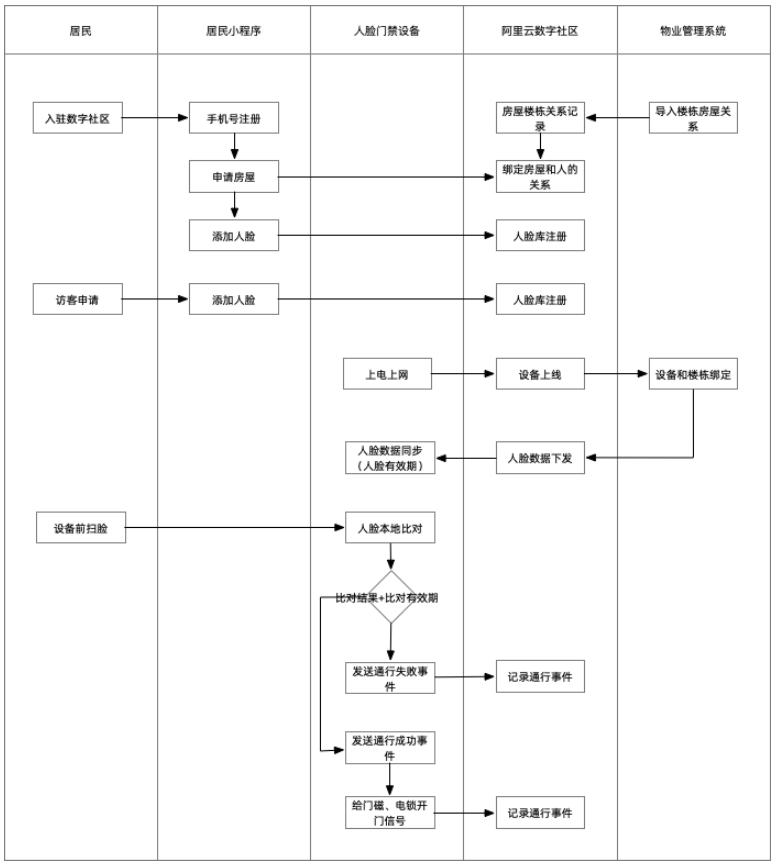
2.3.2.1. 对接说明

1. 整体架构

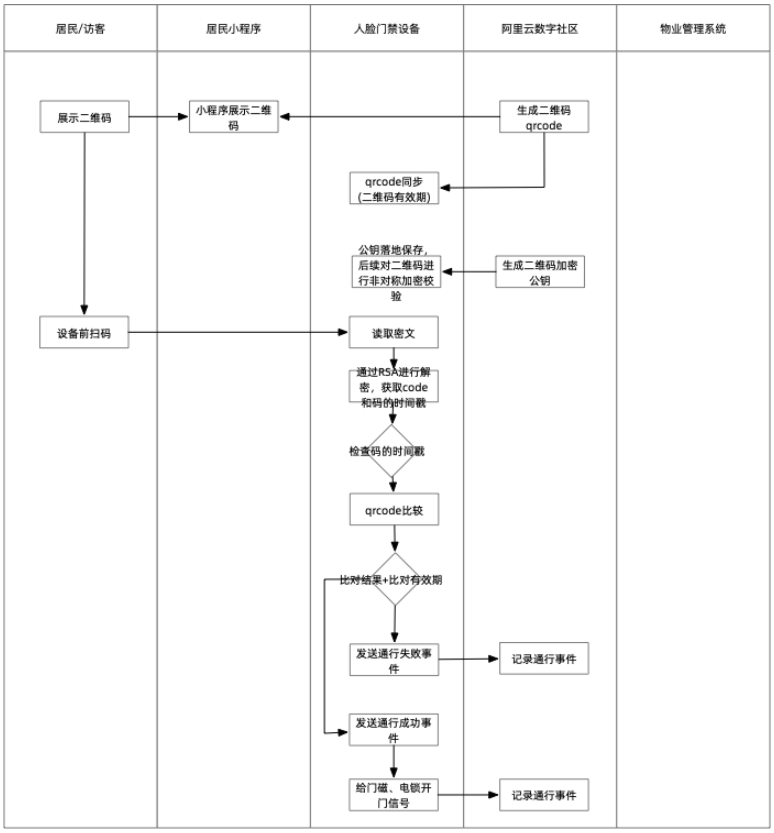


2. 业务流程

1. 智能识别业务链路



2. 二维码业务链路



3. 对接LinkKit-SDK

1. C语言SDK

i. 完整SDK手册：https://code.aliyun.com/edward.yangx/public-docs/wikis/user-guide/linkkit/Archived/V230_Snapshot_Index

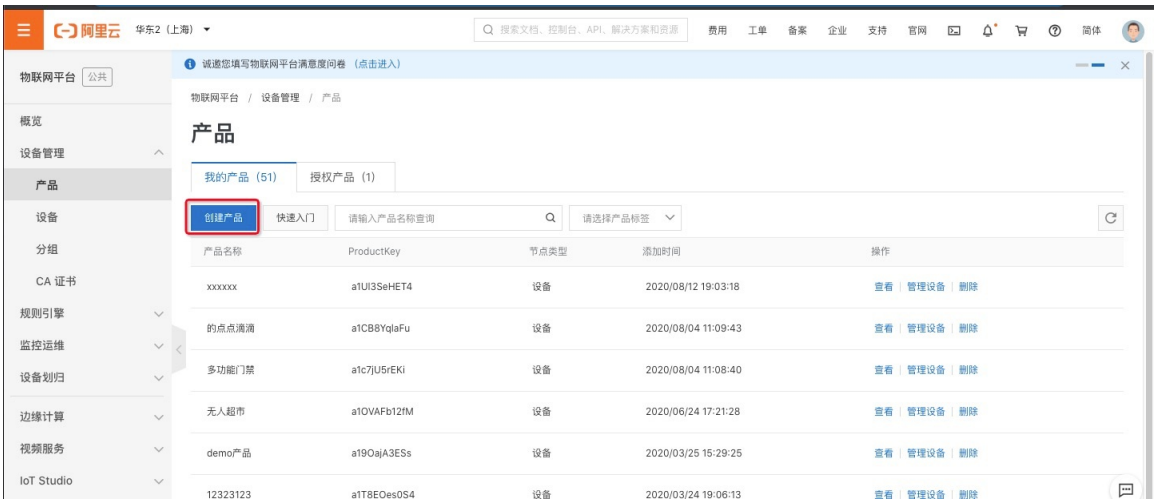
ii. 交叉编译手册：https://code.aliyun.com/edward.yangx/public-docs/wikis/user-guide/linkkit/Archived/V230_Snapshot/porting/Make_Usage

2. Android版SDK手册

i. https://help.aliyun.com/document_detail/96607.html

3. 创建门禁产品

i. 访问地址：https://iot.console.aliyun.com/product，创建产品（创建多功能门禁产品）



ii. **所属品类**选择**多功能门禁**



iii. 前往定义物模型



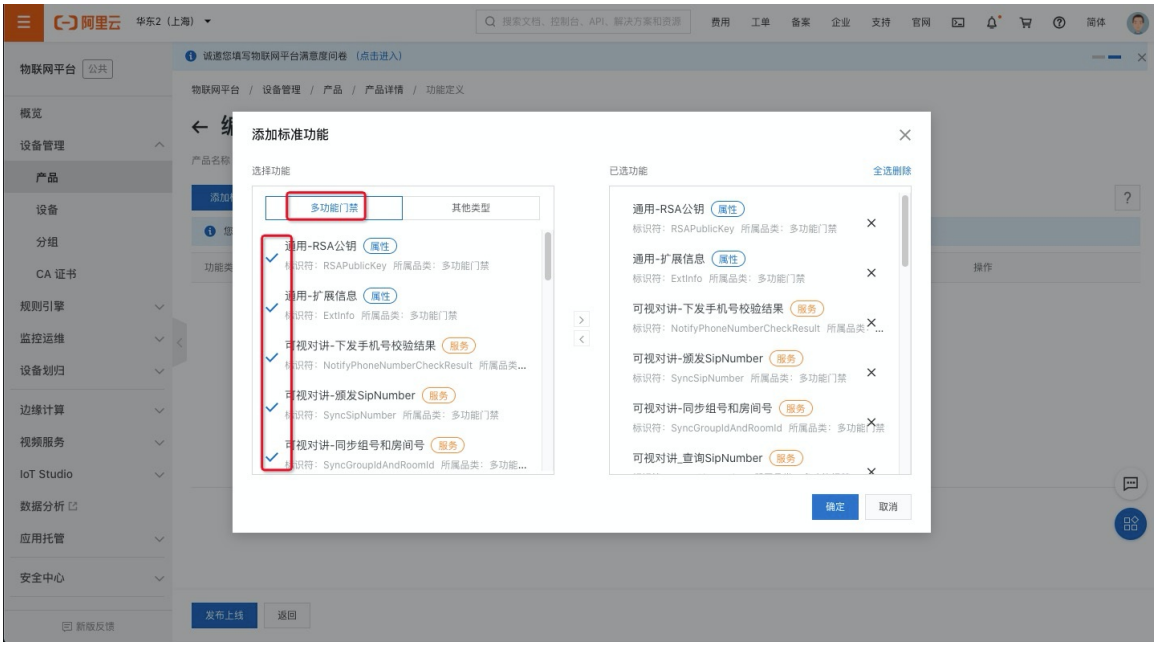
iv. 添加设备的多功能门禁功能，点击**编辑草稿**



v. 点击**添加标准功能**

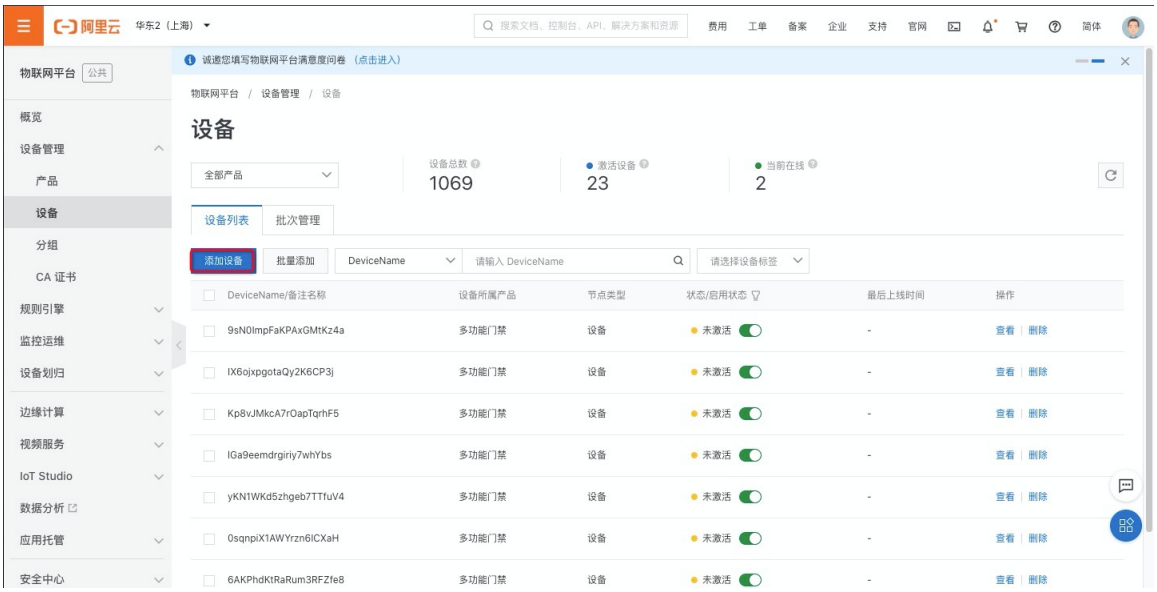


vi. 进入功能页，勾选所有功能项，点击**确认**

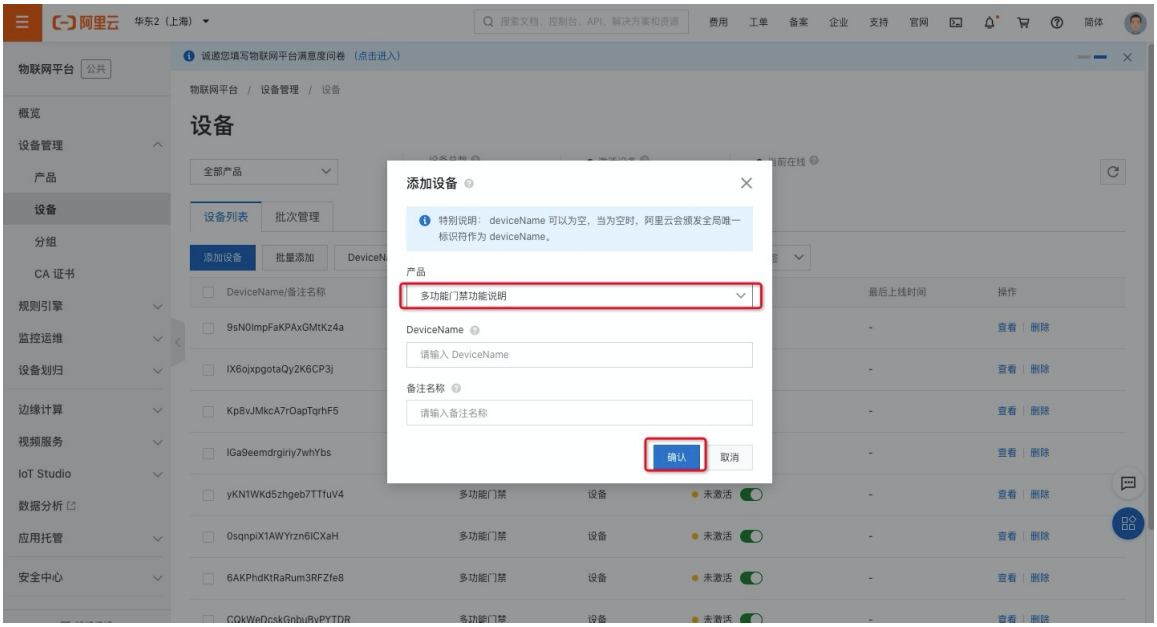


4. 创建门禁设备

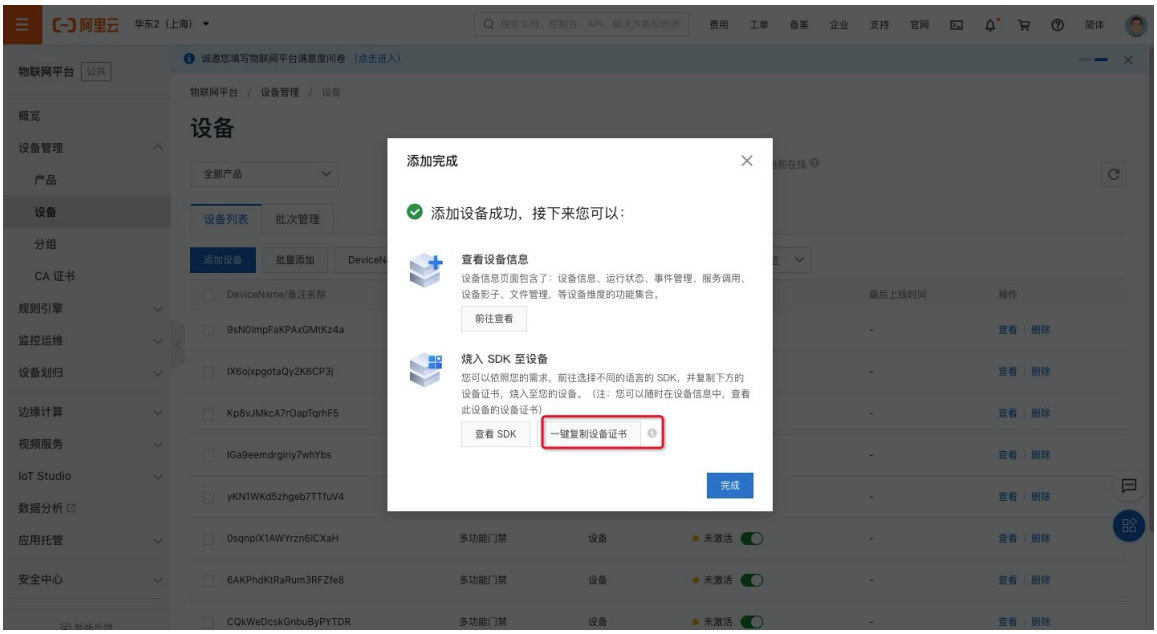
i. 进入创建设备页面，https://iot.console.aliyun.com/devices



ii. 选择多功能门禁产品



iii. 复制多功能门禁证书



5. 下载的证书，请参照一下文档，将设备三元组设置到linkkit sdk中，正常运行后，可以查看到该设备的状态变更为上线，本案例采用一机一密模式进行

   i. C语言设备认证地址：https://code.aliyun.com/edward.yangx/public-docs/wikis/user-guide/linkkit/Archived/V230_Snapshot/programme/Auth_Connect#%E4%B8%80%E6%9C%BA%E4%B8%80%E5%AF%86%E7%BC%96%E7%A8%8B

   ii. Android设备认证地址：https://help.aliyun.com/document_detail/96608.html

6. 设备开发，设备物模型开发手册，您可以通过这个文档内容学习怎么通过sdk完成物模型中的服务和事件的实现，针对门禁对接中您需要实现syncPermissions、remoteOpen两个服务、passEvent一个事件、RSAPublicKey一个属性。

   i. C语言SDK设备开发说明：https://code.aliyun.com/edward.yangx/public-docs/wikis/user-guide/linkkit/Archived/V230_Snapshot/programme/DeviceModel_Prog

   ii. Android版SDK设备开发说明：https://help.aliyun.com/document_detail/96609.html

   iii. RSA公钥（RSAPublicKey）：二维码加密公钥，云端会在设备上电并设备绑定后触发下发的公钥信息，用于后续通过摄像头获取用户二维码之后的解密校验

iv. 同步门禁权限（syncPermissions）：云端下发使用二维码的用户权限配置信息到设备端。参数：权限url（permissionUrl）为权限配置文件下载地址，通过http下载。

字段	类型	备注
qrCodePermissionList	JSONArray	二维码的权限配置内容
tradCardPermissionList	JSONArray	传统卡的权限配置内容
bleCardPermissionList	JSONArray	蓝牙卡的权限配置内容
passwordPermissionList	JSONArray	密码的权限配置内容

二维码的权限配置定义

字段	类型	备注
type	Integer	操作类型。 取值范围：1-增加，2-删除，3-修改。
qrCode	String	用户信息内容。样例：ALI24DD9END8CD3F，表示一个人的唯一id
startTime	String	生效时间（UTC）单位精确到毫秒。
endTime	String	生效时间（UTC）单位精确到毫秒。
maxScanTimes	String	最大刷码限制次数。-1 表示无限制。
maxScanScope	Integer	表明二维码刷码次数限制是多个设备共享还是单个设备维度，默认设备维度。如果不限制刷码次数，无需指定该值。 取值范围：SHARE 或 DEVICE，默认DEVICE。

二维码校验机制，用户在设备前展示二维码，设备获取二维码后进行RSA解密，解密后的数据格式为：code:timestamp（样例：ALI24DD9END8CD3:1589369696）

字段	语义	备注
code	下发二维码权限中的qrCode	需要和本地存储的qrCode列表做对比，对比成功表示当前人是有效的
timestamp	秒级时间戳，表示当前码的有效期	需要和本地设备时间进行对比，如果这个时间戳<当前时间则提示“二维码已过期”

v. 远程开门（remoteOpen）：云端下发对设备的开门命令，设备端执行开门动作

vi. 通行事件（passEvent）：使用二维码、传统卡、蓝牙卡、密码进行通行时，将通行结果及认证信息发到云端

字段	类型	备注
code	Int	错误编码，成功为0，其他值为失败。格式定义见下面code列表。
message	String	错误信息，成功（success），或者失败的原因。格式定义见下面code列表
type	String	认证类型，二维码（qrCode）、传统卡（tradCard）、蓝牙卡（bleCard）、密码（password）、智能（face）
content	String	认证内容（content）：二维码字符串、卡号、蓝牙卡号、密码字符串。
extInfo	String	扩展信息（extInfo）：内容为string格式，先构造JSON对象，再将JSON对象转成string格式放入其中，详见下面示例。

```
{
"code": 0, //
"message": "success", //
"type": "qrCode", //
"content": "ALI7af81cfad5df1", //
"extInfo": "{\"type\":\"qrCode\",\"qrCode\":\"ALI7af81cfad5df1
\",\"userID\":\"508072501fea93ae39fc8737a2fb2a2424f88787\",\"event
Time\":\"1597821579\"}"
7}
```

通行结果（code）	结果说明（message）	
0	成功	success
101	二维码未生效	QR code does not take effect
102	二维码已过期	QR code has expired
103	二维码不存在	QR code does not exist
104	二维码刷码次数用尽	QR code have be used up

vii. 设备SDK下载

a. C语言SDK下载地址：https://product-info.oss-cn-shanghai.aliyuncs.com/linkkit-sdk-c.tar%2830000%29.gz

b. AndroidSDK下载地址：http://gaic.alicdn.com/ztms/android-linkkit-demo/IoTDeviceSDKDemo.zip?spm=a2c4g.11186623.2.14.46642af38h0ZAL&file=IoTDeviceSDKDemo.zip

4. 对接LinkFaceSDK

本章将以C版本SDK为例着重介绍，如过您使用的是Android版本，请查看LinkFace Android设备端开发指南

a. 文件目录介绍

```
tree -L 3
.
├── CMakeLists.txt
├── LinkFace
│   ├── samples
│   │   ├── 1.jpg
│   │   ├── database.c
│   │   ├── database.h
│   │   ├── linkface_demo.c
│   │   ├── linkface_demo_v2.c
│   │   ├── linkkit_adapter.c
│   │   ├── link_visual_face_demo
│   │   ├── link_visual_face_demo_v2  //对接Demo可执行文件
│   │   └── list.h                                  //链表头文件
│   └── sdk
│       ├── liblinkface.a
│       ├── liblinkface.so
│       ├── liblinkface_v2.a                  //linkface静态库
│       ├── liblinkface_v2.so                 //linkface动态库
│       ├── linkface_sdk.h
│       └── linkface_sdk_v2.h                 //linkface头文件
└── third_party                                         //以静态库方式对接时，在此目录下获取需要的三方.a文件
    ├── cJSON
    │   ├── cJSON.h
    │   ├── libcjson.a
    │   ├── libcjson.so
    │   ├── libcjson.so.1
    │   └── libcjson.so.1.7.7
    ├── cJSON-1.7.7.tar.gz
    ├── linkkit-sdk-c
    │   ├── exports
    │   ├── imports
    │   ├── iot_export.h
    │   ├── iot_import.h
    │   ├── libiot_hal.a
    │   ├── libiot_sdk.a
    │   └── libiot_tls.a
    ├── linkkit-sdk-c.tar.gz
    ├── release                                         //以静态库方式对接时，在此目录下获取需要的三方.a文件
    │   ├── apr
    │   ├── apr_util
    │   ├── curl
    │   ├── expat
    │   ├── iconv
    │   ├── mxml
    │   ├── openssl
    │   ├── oss_c_sdk
    │   └── zlib
    ├── sqlite
    │   ├── libsqlite3.a
    │   ├── libsqlite3.so
    │   └── sqlite3.h
    └── sqlite-amalgamation-3310100.zip

    sdk集成方式:
    1.集成动态库:链接liblinkface_v2.so libiot_sdk.a libiot_tls.a libiot_hal.a
    2.集成静态库:链接liblinkface_v2.a libiot_sdk.a libiot_tls.a libiot_hal.a libcjson.a liboss_c_sdk_static.a libaprutil-1.a
libapr-1.a libmxml.a libcurl.a libcrypto.a libssl.a     2.集成静态库:链接liblinkface_v2.a libcjson.a liboss_c_sdk_static.a li
baprutil-1.a libapr-1.a libmxml.a libcurl.a libiconv.a libssl.a libcrypto.a m
```

b. 头文件说明

```
#ifndef __LINKFACE_SDK_V2_H__
#define __LINKFACE_SDK_V2_H__

#include "utils/list.h"

#ifdef __cplusplus
extern "C"
{
#endif

#define LENGTH_PERSON_ID            65
#define LENGTH_PERSON_NAME          128
#define LENGTH_FACE_MD5             33
#define LENGTH_GROUP_ID             64
#define LENGTH_EVENTLIST            1024  //事件队列长度
#define MAX_COUNT_GROUP_ID          16
```

```
#define MAX_COUNT_GROUP_ID          16
#define MAX_LENGTH_EXTRA_INFO       2048

/* 错误码枚举值 */
typedef enum {
    LF_ERR_SUCCESS = 0,                   //成功

    // 添加底库错误码表
    LF_ERR_IMAGE_PARSEPIC_FAIL   = 16001, //图片解析失败
    LF_ERR_IMAGE_PIC_TOOBIG      = 16002, //图片过大
    LF_ERR_IMAGE_PIC_TOOSMALL    = 16003, //图片过小
    LF_ERR_IMAGE_NO_FACE         = 16004, //图片中未检测到人
    LF_ERR_IMAGE_MULTI_FACES     = 16005, //图片中检测到多人
    LF_ERR_IMAGE_FACE_EXIST      = 16006, //已存在
    LF_ERR_IMAGE_PIC_FORMATE_ERR = 16007, //不支持的图片格式
    LF_ERR_IMAGE_PIX_ERR         = 16008, //不支持的像素
    LF_ERR_IMAGE_UNKNOWN_ERR     = 16009, //其他未知错误
    LF_ERR_IMAGE_ALG_ERR         = 16010, //算法异常，无法提特征
    LF_ERR_IMAGE_FACE_DB_FULL    = 16011, //底库达到最大值

    // 文件下载错误码列表
    LF_ERR_URL_DOWNLOAD_FAILED   = 17001  //下载失败
} LINKFACE_ERROR_CODE_E;

/*新增底库需要的字段*/
typedef struct {
  /*! 人员ID：可能是身份证号码，也可能是其他身份唯一标示。 */
  char person_id[LENGTH_PERSON_ID];

  /*! 数据md5值。*/
  char faceimg_md5[LENGTH_FACE_MD5];

  /*! 图片数据大小。单位：字节。仅在注册时有效。 */
  int faceimg_size;

  /*! 图像原始数据。*/
  char* faceimg;

  /*! 新增本次数据时的时间戳，单位为ms。*/
  long long timestamp_ms;
} linkface_image_info;

/*
 * 照片添加队列
 */
typedef struct __tag_add_face_image_list{
    struct  list_head   face_list;
    linkface_image_info face_image_info;
}LINKFACE_ADD_FACE_IMAGE_LIST_ST;

typedef struct {
    /*! 人员ID：可能是身份证号码，也可能是其他身份唯一标示。 */
    char person_id[LENGTH_PERSON_ID];

    /*! 人员姓名，可能为空。*/
    char name[LENGTH_PERSON_NAME];

    /*! 该人员所在的组ID，一个人可能存在于多个组中*/
    char group_ids[MAX_COUNT_GROUP_ID][LENGTH_GROUP_ID];

    int group_id_count;

    /*! 新增该人员信息时的时间戳，单位为ms。*/
    long long timestamp_ms;

    /*! 注册时的其他信息。 */
    char *extra_info;
} linkface_user_info;

/// 比对状态。
typedef enum {
    /*! 比对失败。 */
    LINKFACE_MATCH_STATUS_FAILED  = -1,

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

    /*! 比对成功。 */
    LINKFACE_MATCH_STATUS_SUCCESS    = 1,
} linkface_match_status_t;
```

```
}linkface_match_status_t;

// 增量事件类型
typedef enum {
    //用户信息
    LINKFACE_SYNC_EVENT_USERINFO    = 1,
    //用户图片
    LINKFACE_SYNC_EVENT_USERPIC     = 2,
    //用户特征
    LINKFACE_SYNC_EVENT_USERFEATURE = 3,
    //用户快速删除
    LINKFACE_SYNC_EVENT_USERFASTDEL = 4,
    //删除用户组
    LINKFACE_SYNC_EVENT_GROUPDEL    = 5,
    //清空所有数据
    LINKFACE_SYNC_EVENT_DELTOTAL    = 21
}LINKFACE_SYNC_EVENT_E;

/*对比上报*/
/*注册需要的字段*/
typedef struct {
    /*! 比对状态。 */
  linkface_match_status_t match_status; //对比成功才会有(即person_id，match_score)的值才有意义

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

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

  /*! 全景图大小。单位为Byte，不能超过10MByte. */
  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

  /*! 全景图原始数据。必要字段。 */
  char *panorama_img;

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

  /*! 事件产生utc时间。 必要字段。 */
  long long timestamp_ms;
}linkface_match_result;

/*
 * 事件上报队列
 */
typedef struct __tag_event_list{
    struct  list_head       event_list;
    linkface_match_result   match_result;
}LINKFACE_EVENT_LIST_ST;

/** 云端下发底库同步请求时，sdk先缓存图片，再调用该接口将图片批量入库。
 * @brief        批量注册数据接收回调函数。
 * @param[IN]    pImageInfoList，图片信息列表。
 * @param[IN]    num，该值为pImageInfoList、pErrorCode的个数。
 * @param[OUT]   pErrorCode，每张图片的入库结果
 * @ret          void
 *
 * 注意：
 * 1. 该接口为同步调用，接口不宜耗时过久。
 * 2. 该接口不会在多线程中并发调用。
 * 3. 每张图片的添加结果，都要返回到pErrorCode中
 * 4. 错误码请严格按照 @LINKFACE_ERROR_CODE_IMAGE_E 中的定义.
 */
typedef int (*linkface_add_image_batch_cb)(const struct list_head *pImageInfoList, int num, int *pErrorCode);

/*
 * 添加用户信息，成功返回0，失败返回其他值.
 */
typedef int (*linkface_add_user_cb)(const linkface_user_info *info, void *user);

/** 云端需要删除某个底库时，sdk会调用该接口将信息逐个传递给厂商，厂商需要将该从算法数据库中删除.
 * @brief    删除数据接收回调函数格式定义。
 * @param    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。
 * @ret      如果删除成功，则返回0，否者返回其他值。
 *
 * 注意：
 * 1. 只有数据真实存在时(person_id/faceimg_md5组合校验)，才返回成功。
 * 2. person_id 不存在返回-1
```

```
    2. person_id 不存在返回 1。
 * 3. person_id 存在而 faceimg_md5 不存在，返回-2。
 * */
typedef int (*linkface_del_image_cb)(const linkface_image_info *info, void *user);

/*
 * 删除用户信息，成功返回0，失败返回其他值.
 * */
typedef int (*linkface_del_user_cb)(const linkface_user_info *info, void *user);

/*
 * 基于入库的时间戳，轮询底库信息。
 * */
typedef void (*linkface_polling_image_cb)(long long timestamp_ms, void *user, void (*cb_sdk)(const linkface_image_info *info
, void *user), void *user_sdk);

/*
 * 基于入库的时间戳，轮询人员库信息。
 * */
typedef void (*linkface_polling_user_info_cb)(long long timestamp_ms, void *user, void (*cb_sdk)(const linkface_user_info *i
nfo, void *user), void *user_sdk);

/*
 * 基于人员id，查询底库信息。
 * 厂商实现，结构体内存由sdk释放。
 * */
typedef linkface_image_info* (*linkface_query_image_info_cb)(char *person_id, void *user);

/*
 * 基于人员id，查询人员信息.
 * */
typedef linkface_user_info* (*linkface_query_user_info_cb)(char *person_id, void *user);

/*
 * 查询端侧图片总数
 * */
typedef int (*linkface_querey_image_count_cb)(void *user);

/*
 * @brief       清除用户分组下的所有用户信息
 * @param[IN]   group_id，用户组ID
 * @ret         如果删除成功，则返回0，否者返回具体错误码
 */
typedef int (*linkface_clear_usergroup_cb)(const char *group_id);

/*
 * 清空库回调函数，由设备厂商实现
 *
 * */
typedef int (*linkface_clearall_cb)();


/**  当门禁机检测到后，无论是否能正确识别出该，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，失败返回其他值。
 *
 * 注意：
 * 1. 该接口为同步接口，请勿并发调用。
 * 2. 若接口返回失败，建议最多重试3次。
 */
int linkface_upload_match_result(const linkface_match_result *result);

/* 产品三元组长度限制 */
#define LINKFACE_PRODUCT_KEY_LEN     (20)
#define LINKFACE_DEVICE_NAME_LEN     (32)
#define LINKFACE_DEVICE_SECRET_LEN   (64)

typedef struct {
    char product_key[LINKFACE_PRODUCT_KEY_LEN + 1]; /* 三元组之一，由平台颁发的产品唯一标识，11位长度的英文数字随机组合 */
    char device_name[LINKFACE_DEVICE_NAME_LEN + 1]; /* 三元组之一，在注册设备时，自定义的或自动生成的设备名称，具备产品维度内的唯一性
*/
    char device_secret[LINKFACE_DEVICE_SECRET_LEN + 1];  /* 三元组之一，物联网平台为设备颁发的设备密钥*/
} linkface_device_triple_t; /* 物联网平台上分发的三元组信息。 */

/* SDK日志等级 */
typedef enum {
    LF_LOG_DEBUG = 0,
    LF_LOG_INFO = 1,
    LF_LOG_WARN = 2
```

```
    LF_LOG_ERROR = 3,
} linkface_log_level_e;

typedef enum {
    LF_STDOUT = 0,
    LF_FILE   = 1
} linkface_log_dest_e;

typedef struct {
    linkface_log_level_e lvl;
    linkface_log_dest_e  dst;
} linkface_log_cfg_t;

typedef struct {
    char        *logpath;           //日志文件保存路径
    char        *picpath;           //照片保存路径
    char        *eventpath;         //告警事件文件保存路径
    int         max_size_log_mb;    //日志存储目录大小Mb
    int         max_size_cache_mb;  //照片存储目录大小Mb
    int         max_size_event_mb;  //告警事件存储目录大小Mb
} linkface_cache_cfg_t;

typedef struct {
    int                       addFaceBatchSize; //每次批量添加的人数，范围为1~15
    linkface_device_triple_t  triple;
    linkface_log_cfg_t        cfg_log;
    linkface_cache_cfg_t      cfg_cache;

    /* 批量增加底库图片*/
    struct {
        linkface_add_image_batch_cb cb;
        void                        *user;
    } config_batch_add_image;

    /* 删除一个底库图片*/
    struct {
        linkface_del_image_cb     cb;
        void                      *user;
    } config_del_image;

    /* 基于id查询底库图片信息*/
    struct {
        linkface_query_image_info_cb cb;
        void                      *user;
    } config_query_image_info;

    /* 遍历所有底库图片信息*/
    struct {
        linkface_polling_image_cb cb;
        void                        *user;
    } config_polling_image_info;

    /* 增加人员信息. */
    struct {
        linkface_add_user_cb   cb;
        void                    *user;
    } config_add_user;

    /* 删除人员信息. */
    struct {
        linkface_del_user_cb     cb;
        void                      *user;
    } config_del_user;

    /* 查询人员信息. */
    struct {
        linkface_query_user_info_cb cb;
        void                      *user;
    } config_query_user_info;

    /* 轮询所有人员信息. */
    struct {
        linkface_polling_user_info_cb cb;
        void                        *user;
    } config_polling_user_info;

    /* 聚合接口：查询端侧总数. */
    struct {
        linkface_query_image_count_cb cb;
```

```
        void                                *user;
    } config_query_image_count;

    /* 聚合接口： 清空所有和底库图片信息. */
    struct {
        linkface_clearall_cb cb;
    } config_clearall;

    /* 清空指定分组. */
    struct {
        linkface_clear_usergroup_cb cb;
    } config_clear_usergroup;

} linkface_config_t;

/*
 * SDK 初始化接口.
 */
int linkface_init(linkface_config_t *cfg);

/*
 * 消息路由接口，源码开放.
 */
int linkface_message_router(const char *service_name, const int service_name_len, const char *request, const int request_len
, char **response, int *response_len);

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

#ifdef __cplusplus
}
#endif

#endif
```

c. 对接DEMO源码

```
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
//填写.h文件的实际路径
#include "../src/v2/utils/list.h"
#include "linkface_sdk_v2.h"

// 固定失败一些入库图片
#define  FAILED_SOME_PIC

extern int linkkit_destroy();
extern int linkkit_init(const char *product_key, const char *device_name, const char *device_secret);
int g_flag_is_running = 1;

void sig_handler(int signo)
{
    printf("catch the signal SIGINT %d\n",signo);
    g_flag_is_running = 0;
    exit(0);
}

/* 批量入库回调函数 */
static int _add_user_batch(const struct list_head *pImageInfoList, int num, int *pErrorCode, void *arg)
{
    if (!list_empty(pImageInfoList))
    {
        LINKFACE_ADD_FACE_IMAGE_LIST_ST  *node = NULL;
        LINKFACE_ADD_FACE_IMAGE_LIST_ST  *next = NULL;
        int cur_index = 0;
        list_for_each_entry_safe(node, next, pImageInfoList, face_list, LINKFACE_ADD_FACE_IMAGE_LIST_ST)
        {
        #ifdef FAILED_SOME_PIC
            // 固定第二个错误
            if (++cur_index == 1){
                *(pErrorCode + cur_index) = -1;
                continue;
            }
        #endif
```

```
            add_image_info((const char *)(&(node->face_image_info))); //add_image_info由设备厂商实现
            printf("add_image person_id:{%s}, md5:{%s}\n", node->face_image_info.person_id, node->face_image_info.faceimg_md
5);
        }
    }
    return 0;
}

/*
 * 用户照片删除
 */
static int _del_image(const linkface_image_info *person, void *arg)
{
    printf("del face from local database: %s\n", person->person_id);

    return del_image_info((const char *)person->person_id); //del_image_info由设备厂商实现
}

/*
 * 用户信息添加
 */
static int _add_user(const linkface_user_info *person, void *arg)
{
    printf("add user to local database: %s, name: %s\n", person->person_id, person->name);

    return add_user_info((const linkface_user_info*)person); //add_user_info由设备厂商实现
}

/*
 * 用户信息删除
 */
static int _del_user(const linkface_user_info *person, void *arg)
{
    printf("del user from local database: %s\n", person->person_id);

    return del_user_info((const char *)person->person_id); //del_user_info由设备厂商实现
}

/*
 * 清空库
 */
static int _clear_all()
{
    clear_data(); //clear_data由设备厂商实现
    return 0;
}

/*
 * 清除指定用户分组
 */
static int _clear_gourp(char * groupid)
{
    printf("del group face from local database: %s\n", groupid);
    return clear_gourp_face(groupid);   //clear_gourp_face由设备厂商实现

}

/*
 * 底库照片总数查询回调函数
 */
static int _query_image_base_count(void *arg)
{
    int count = 0;

    count = query_image_count(); //clear_gourp_face由设备厂商实现

    printf("query facebase count : %d\n", count);
    return count;
}

/*
 * 用户信息查询回调函数
 */
static linkface_user_info *_query_user_info(char *person_id, void *arg)
{
    linkface_user_info *tmp = NULL;
    int len = sizeof(linkface_user_info);

    tmp = (linkface_user_info*)malloc(len);
```

```
    if (tmp == NULL) {
        return NULL;
    }
    memset(tmp, 0, len);

    query_user_info((const char *)person_id, &tmp); //query_user_info由设备厂商实现
    printf("query user info: %s, name: %s\n", person_id, tmp->name);

    if(tmp->timestamp_ms == 0) {
        free(tmp);
        tmp = NULL;
    }

    return tmp;
}

/*
 * 用户照片查询回调函数
 */
static linkface_image_info *_query_image_info(char *person_id, void *arg)
{
    linkface_image_info *tmp = NULL;
    int len = sizeof(linkface_image_info);

    tmp = (linkface_image_info *)malloc(len);
    if (tmp == NULL) {
        return NULL;
    }
    memset(tmp, 0, len);

    query_image_info((const char *)person_id, &tmp);    //query_image_info由设备厂商实现

    printf("query face info: %s, md5: %s\n", person_id, tmp->faceimg_md5);
    if(strlen(tmp->faceimg_md5) == 0) {
        free(tmp);
        tmp = NULL;
    }
    return tmp;
}

/*
 * 用户照片遍历回调函数
 */
static void _polling_image_base (long long timestamp, void *arg, void (*cb_sdk)(const linkface_image_info *person, void *use
r), void *user_sdk)
{
    printf("polling_image_base\n");
    polling_image_info(timestamp, cb_sdk, user_sdk);    //polling_image_info由设备厂商实现
}

/*
 * 用户信息遍历回调函数
 */
static void _polling_userinfo_base (long long timestamp, void *arg, void (*cb_sdk)(const linkface_user_info *person, void *u
ser), void *user_sdk)
{
    printf("polling_userinfo_base\n");
    polling_user_info(timestamp, cb_sdk, user_sdk);     //polling_user_info由设备厂商实现
}

#define PRODUCT_KEY ""
#define DEVICE_NAME ""
#define DEVICE_SECRET ""

int main(int argc, char **argv)
{
    int ret = 0;
    const char *pk = NULL;
    const char *dn = NULL;
    const char *ds = NULL;
    const char *dir = NULL;
    /*
      alarm_interval < 0 直接退出
      alarm_interval = 0 不上报事件
      alarm_interval > 0 代表上报事件间隔
     */
    int alarm_interval = 0;
    if(argc < 5) {
        printf("use default triples, you can modify it by: ./xxx_demo $pk $dn $ds $path \n");
```

```
        pk = PRODUCT_KEY;
        dn = DEVICE_NAME;
        ds = DEVICE_SECRET;
        dir = "./";
    } else {
        pk = argv[1];
        dn = argv[2];
        ds = argv[3];
        dir = argv[4];
        alarm_interval = atoi(argv[5]);
    }

    if (signal(SIGINT,sig_handler) == SIG_ERR) {
        perror("signal errror");
        exit(EXIT_FAILURE);
    }

    /* 初始化linkface配置参数。 */
    linkface_config_t cfg;
    memset(&cfg, 0, sizeof(linkface_config_t));

    /* 物联网平台三元组信息. */
    strncpy(cfg.triple.product_key, pk, LINKFACE_PRODUCT_KEY_LEN);
    strncpy(cfg.triple.device_name, dn, LINKFACE_DEVICE_NAME_LEN);
    strncpy(cfg.triple.device_secret, ds, LINKFACE_DEVICE_SECRET_LEN);

    /* 日志输出配置信息. */
    cfg.cfg_log.lvl = LF_LOG_DEBUG;
    cfg.cfg_log.dst = LF_FILE;

    /* 缓存配置信息. */
    cfg.addFaceBatchSize = 10;
    cfg.cfg_cache.logpath = dir;
    cfg.cfg_cache.picpath = dir;
    cfg.cfg_cache.eventpath = dir;
    cfg.cfg_cache.max_size_log_mb = 30;     //日志文件目录100M
    cfg.cfg_cache.max_size_cache_mb = 50;  //缓存图片 500M
    cfg.cfg_cache.max_size_event_mb = 50;

    /* 用户需要实现的接口：增加底库信息. */
    cfg.config_batch_add_image.cb = _add_user_batch;
    /* 用户需要实现的接口：删除底库信息. */
    cfg.config_del_image.cb = _del_image;
    /* 用户需要实现的接口：通过id查询某个底库信息*/
    cfg.config_query_image_info.cb = _query_image_info;
    /* 用户需要实现的接口：轮询底库信息*/
    cfg.config_polling_image_info.cb = _polling_image_base;
    /* 用户需要实现的接口：查询底库总数*/
    cfg.config_query_image_count.cb = _query_image_base_count;

    /* 用户需要实现的接口：增加用户信息 */
    cfg.config_add_user.cb = _add_user;
    /* 用户需要实现的接口：删除用户信息 */
    cfg.config_del_user.cb = _del_user;
    /* 用户需要实现的接口：通过id查询用户信息*/
    cfg.config_query_user_info.cb = _query_user_info;
    /* 用户需要实现的接口：轮询用户信息*/
    cfg.config_polling_user_info.cb = _polling_userinfo_base;

    /* 用户需要实现的接口：清空底库. */
    cfg.config_clearall.cb = _clear_all;

    cfg.config_clear_usergroup.cb = _clear_gourp;
    /* 确保linkface + linkkit 初始化完成.*/
    while (1)
    {
        ret = linkkit_init(pk, dn, ds);
        if (ret < 0)
        {
            printf("linkkit sdk init failed, retry 1s later.\n");
            sleep(1);
            if(g_flag_is_running == 0)
            {
                return 0;
            }
            continue;
        }

        ret = linkface_init(&cfg);
```

```
        if (ret < 0)
        {
            printf("linkface sdk init failed, retry 1s later.\n");
            linkkit_destroy();
            sleep(1);
            if (g_flag_is_running == 0)
            {
                return 0;
            }
            continue;
        }
    }

    return 0;
}
```

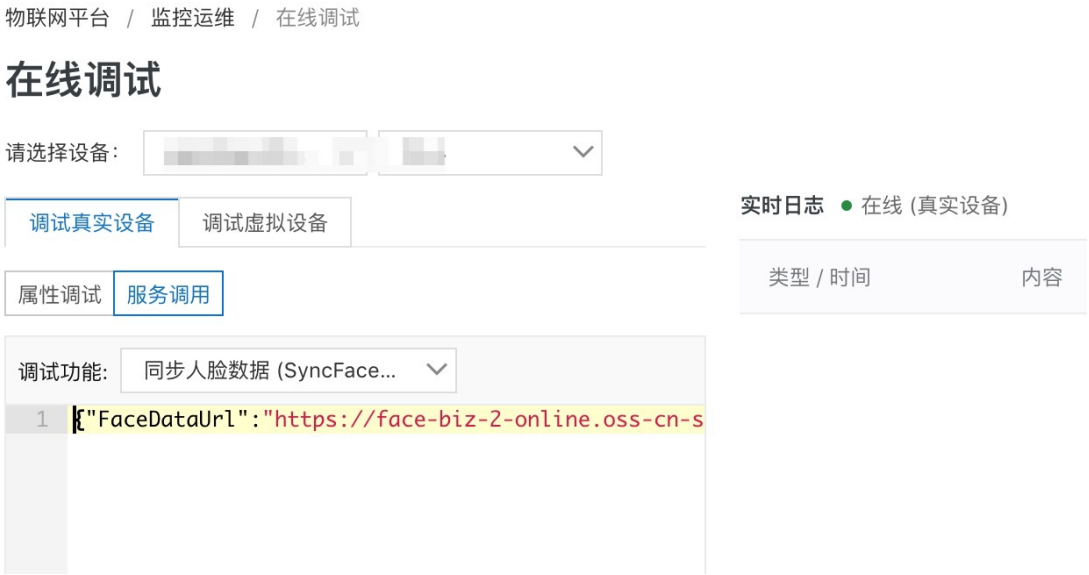
5. 单元测试方法

a. 设备自测，进入设备**在线调试**界面





b. 选择**同步数据**服务



a. 新增

```
{"FaceDataUrl":"https://linkface-doc.oss-cn-shanghai.aliyuncs.com/logdpx-add-2.json"}
```

c. 删除

```
{"FaceDataUrl":"https://linkface-doc.oss-cn-shanghai.aliyuncs.com/logdpx-del.json"}
```

d. 清空

```
{"FaceDataUrl":"https://face-biz-2-online.oss-cn-shanghai.aliyuncs.com/05A19AEFC200431B9E1F7C371B754B48/a103B6PNtuKdx2Fr/a15qAuXknvo_lfv_4_MAIN_CURSOR_0_1597213340164.json?Expires=1599805340&OSSAccessKeyId=************&Signature=hgFXz%2FMr9lOr6CtfoNzVth8Jmko%3D"}
```

c. 选择**RSA公钥**服务




```
{  "RSAPublicKey": "MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAI6zQY57YYUh1xDOk9ZOXQapFpSGmZIukTtJkvM5rf43b4H6tmzH27anYW8csBKmFAdJir9ZaJY1GhZO6FY/8d0CAwEAAQ=="}
```

以下二维码是私钥加密的测试二维码

a. 有效期至：2022年的二维码（明文：ALIF24DD9END8CD3:1661011200，密文：Pd3fZ3VDviUjZ6dREW1wIeITxKQBAmmxJ1ZRL6kmAfq2RmXGiZBiv8wh39ySPODojwPCpYwsF4Ken5znf/sGtQ==）



b. 已过期的二维码（明文：ALIF24DD9END8CD3:1597939200，密文：XW5qDO45+wJe31/q5mMkx6EZDKf754yg2pg3QhUoSL6VBtwIzUlkKW+NilPC5tXpdlRll92PNLhjtFlU707l2w==）



d. 选择**同步门禁权限**服务

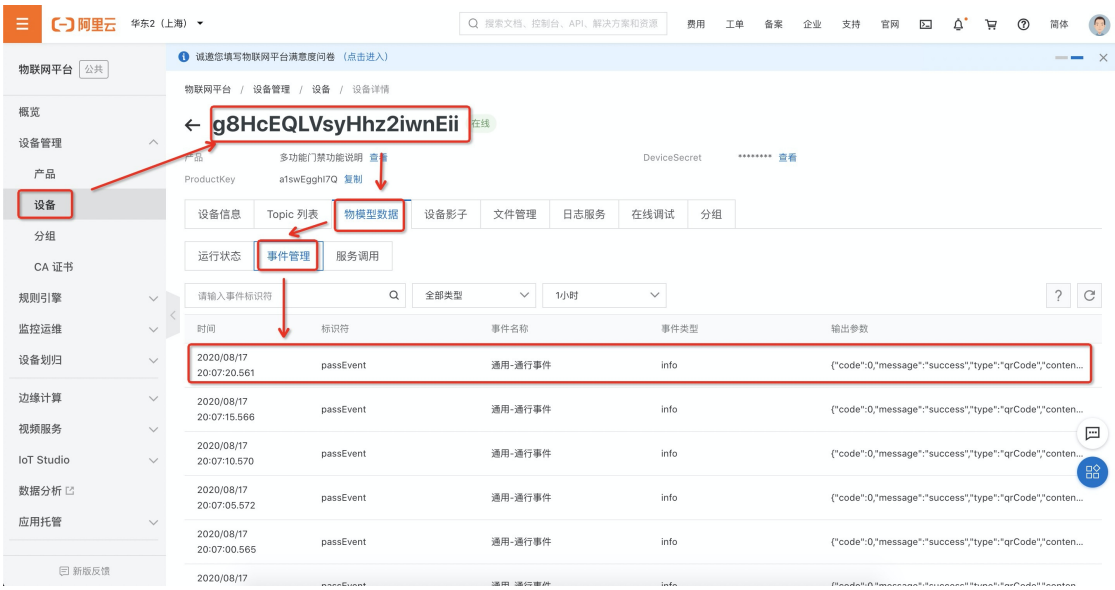



```
{"permissionUrl":"https://product-info.oss-cn-shanghai.aliyuncs.com/test.json"}
```

url为您上传文件在互联网上可直接下载的url地址，文件demo文件内容见下方

```
{
    "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"
        }
    ]
}
```

e. 二维码通行事件



在物联网平台中，查看设备的物模型数据，可以通过passEvent来进行过滤检索，检查上报的内容和下面的json数据一致

```
{"code":0,"message":"success","type":"qrCode","content":"ALIF24DD9END8CD3"}
```

2.3.2.2. Android设备端开发指南

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

SDK集成

依赖Linkkit SDK 的通道能力，在集成本SDK之前，请务必完成Linkkit SDK的集成。

依赖引入

```
// 1. 在根目录下的build.gradle中添加Aliyun Maven仓库的引用
allprojects {
    repositories {
        maven {
            url "http://maven.aliyun.com/nexus/content/repositories/releases"
        }
        maven {
            url "http://maven.aliyun.com/nexus/content/repositories/snapshot"
        }
    }
}

// 2. App build.gradle中添加依赖
implementation('com.aliyun.iotx:linkface:1.0.0')


// 3. App build.gradle中约束ABI，只使用armeabi-v7a
android {
    defaultConfig {
        externalNativeBuild {
            ndk {
                abiFilters "armeabi-v7a"
            }
        }
    }
}
```

> *注意：*
>
> *如果使用SNAPSHOT包，因Android Studio默认不会主动对SNAPSHOT包拉取更新，因而会导致SNAPSHOT包更新时本地与远程不一致，请在App的build.gradle文件中添加以下配置，以便Android Studio禁用本地cache：*
>
> *configurations.all { resolutionStrategy.cacheChangingModulesFor 0, 'seconds'}*

混淆配置

```
# keep and don't warn LinkFace
-dontwarn com.aliyun.iotx.linkvisual.linkface.**
-keep class com.aliyun.iotx.linkvisual.linkface.** { *; }
```

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" />
```

最佳对接流程

1. 初始化

请在Linkkit SDK初始化完成后对LinkFace进行初始化，设备三元组必须传入。

```
@Override
public void onInitDone(Object data) {
    Log.d(TAG, "onInitDone() called with: data = [" + JSON.toJSONString(data) + "]");
    linkkitConnectState = ConnectState.CONNECTED;
    getServiceList();
    setServiceHandler();
    //初始化LinkFace SDK
    LinkFaceConfig linkFaceConfig = LinkFaceConfig.newBuilder()
        .productKey(productKey)
        .deviceName(deviceName)
        .deviceSecret(deviceName)
        .log_level(0)
        .build();
    int ret = LinkFace.getInstance().init(linkFaceConfig, MainActivity.this);
    appendLog("LinkFace SDK 初始化完成: " + ret);
}
```

2. 监听Linkkit通道消息

LinkFace SDK由Linkkit通道的下行消息来驱动，需要在Linkkit的onNotify回调中将下行topic派遣到LinkFace SDK进行处理。

```
    private IConnectNotifyListener connectNotifyListener = new IConnectNotifyListener() {
        @Override
        public void onNotify(String connectId, String topic, AMessage aMessage) {
            Log.d(TAG, "onNotify() called with: connectId = [" + connectId + "], topic = [" + topic + "], aMessage = ["
                    + new String((byte[]) aMessage.data) + "]");
            /**
             * 添加LinkFace SDK对LinkKit异步消息的监听。
             */
            LinkFace.getInstance().notifyTopicReceived(connectId, topic, aMessage);
        }

        @Override
        public boolean shouldHandle(String connectId, String topic) {
            return true;
        }

        @Override
        public void onConnectStateChange(String connectId, ConnectState connectState) {
            appendLog("Linkkit连接状态变更为"+ connectState);
            linkkitConnectState = connectState;
        }
    };
```

3. 创建数据库

建议分别建立人员信息表和底库表两个表，两表之间通过人员信息ID进行外键关联，表结构信息如下：

- 人员信息表 user_info

列名	类型	长度	是否为NULL	含义
id	integer	64	否	表的主键，递增
person_id	vchar	64	否	人员信息的唯一ID，实际可能为身份证号，学号，工号之类的。

name	vchar	128	可以	人员的姓名或别名，识别到后，可能需要在面板机上展示识别结果。
group_ids	vchar	1024	否	人员所在的分组信息，建议实际存储为JSON Array格式以便于检索，比如 ["group_1", "group_2", "group_3"]。
group_id_count	integer	64	否	用来标示@group_ids人员分组的个数。
timestamp_ms	integer	64	否	此条人员信息更新的时间，不能由端侧自主生成，必须使用SDK传递过来的时间。
extra_info	vchar	1024	可以	附加信息，格式为JSON String，建议长度为1K，可能不存在。

- 底库图片信息表 image_info

列名	类型	长度	是否为NULL	含义
id	integer	64	否	表的主键，递增
person_id	vchar	64	否	人员信息的唯一ID，实际可能为身份证号，学号，工号之类的。
faceimg_md5	vchar	64	否	图片的md5值，必须使用SDK传递过来的值。
faceimg_size	integer	32	否	图片的大小
timestamp_ms	integer	64	否	此条底库图片更新的时间，不能由端侧自主生成，必须使用SDK传递过来的时间。

4. 处理回调接口

4.1 对接底库管理相关接口

4.1.1 批量添加底库

```
/**
 * 云端下发底库同步请求时，sdk先缓存图片，再调用该接口将图片批量入库。
 *
 * @param imageInfos 注册图片信息列表
 * @param num 图片信息列表的数量
 * @return 每张图片入库结果. 参考: {@link com.aliyun.iotx.linkvisual.linkface.bean.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. 每张图片的添加结果，都要按序放入返回数组中<br>
 * 4. 错误码请严格按照 {@link com.aliyun.iotx.linkvisual.linkface.bean.LinkFaceErrorCode} 中的定义<br>
 */
int[] onAddImageInfo(LinkFaceImageInfo[] imageInfos, int num);
```

把图片添加到算法特征库并执行SQL语句如下：

```
INSERT INTO image_info (person_id, faceimg_md5, faceimg_size, timestamp_ms)
VALUES ("aaa", "asdfasdfasf", 12, 12356337889);
```

4.1.2 删除底库

```
    /**
     * 云端需要删除某个底库时，sdk会调用该接口将信息逐个传递给厂商，厂商需要将该数据从数据库中删除
     * @param   imageInfo 带删除图片信息
     * @ret     如果删除成功，则返回0，否者返回其他值. 参考：{@link com.aliyun.iotx.linkvisual.linkface.bean.LinkFaceErrorCode}
     *
     * 注意:<br>
     * 1. 只有数据真实存在时(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);
```

把图片特征从算法库中删除并执行删除@image_info表的一行记录，SQL语句如下：

```
DELETE FROM image_info WHERE person_id = "aaa";
```

4.1.3 查询底库数据

```
    /**
     * 基于人员id，查询底库信息
     *
     * @param personId 人员ID
     * @return 图片信息
     */
    LinkFaceImageInfo onQueryImageInfo(String personId);
```

基于person_id查询image_info的一行记录，SQL语句如下：

```
SELECT * from image_info WHERE person_id = "aaa";
```

4.1.4 查询底库总数

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

SQL语句如下：

```
SELECT count(*) from image_info;
```

4.2 对接人员信息管理相关接口

4.2.1 添加人员信息

```
    /**
     * 添加人员信息
     * @param userInfo 待添加的人员信息.
     * @return  如果人员信息添加成功，则返回0，否者返回其他值. 参考：{@link com.aliyun.iotx.linkvisual.linkface.bean.LinkFaceErrorCode}
     */
    int onAddUserInfo(LinkFaceUserInfo userInfo);
```

对应SQL:

```
INSERT INTO user_info (person_id, name, group_ids, group_id_count, timestamp_ms, extra_info)
VALUES ("aaa", "阿里巴巴", "[\"aaa\", \"bbb\"]", 2, 12356337889, "");
```

4.2.2 删除人员信息

```
    /**
     * 删除用户信息
     * @param userInfo 待删除的人员信息
     * @return 如果删除人员信息成功，则返回0，否者返回其他值. 参考：{@link com.aliyun.iotx.linkvisual.linkface.bean.LinkFaceErrorCode}
     */
    int onDeleteUserInfo(LinkFaceUserInfo userInfo);
```

对应SQL:

```
DELETE FROM user_info WHERE person_id = "aaa";
```

4.2.3 查询人员信息

```
    /**
     * 基于人员id，查询人员信息
     *
     * @param personId 人员ID
     * @return 人员信息
     */
    LinkFaceUserInfo onQueryUserInfo(String personId);
```

对应SQL:

```
SELECT * from image_info WHERE person_id = "aaa";
```

4.3 对接其他接口

4.3.1 删除用户组下的所有数据

```
    /**
     * 清除用户分组下的所有用户信息和相关数据
     * @param groupId 用户组ID
     * @return 如果删除用户组成功，则返回0，否者返回具体错误码. 参考：{@link com.aliyun.iotx.linkvisual.linkface.bean.LinkFaceErrorCode}
     */
    int onClearUserGroup(String groupId);
```

4.3.2 清除所有数据

```
    /**
     * 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.linkvisual.linkface.bean.LinkFaceErrorCode}
     */
    int onClearAll();
```

对应SQL:

```
DELETE FROM user_info;
DELETE FROM image_info;
```

4.3.2 对接Linkkit通道状态感知接口

LinkFace内部的运行需要参考Linkkit通道的连接状态，需实现onLinkKitCheckStateNormal回调方法。

```
    @Override
    public int onLinkKitCheckStateNormal() {
        appendLog("查询连接状态" + linkkitConnectState);
        return linkkitConnectState == ConnectState.CONNECTED ? 1 : 0;
    }
```

5 对接检测接口

无论比对是否成功，均调用SDK

getInstance

().uploadMatchResult(linkFaceMatchResult)接口，差异在于比对失败的case里，person id可以填空。

```
    /**
     * 当门禁机检测到后，无论是否能正确识别，均需要调用本接口将识别结果上报到云端。
     *
     * @param linkFaceMatchResult 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
     *
     * 注意：
     * 1. 该接口为同步接口，请勿并发调用。
     * 2. 若接口返回失败，建议最多重试3次。
     */
    public int uploadMatchResult(LinkFaceMatchResult linkFaceMatchResult);
```

6. 销毁

LinkFace应在长生命周期组件中运行，确认不使用时，请销毁.

```
    @Override
    protected void onDestroy() {
        super.onDestroy();
        faceDBHelper.close();
        LinkKit.getInstance().unRegisterOnPushListener(connectNotifyListener);
        // 销毁LinkFace
        LinkFace.getInstance().destroy();
    }
```

包内容介绍

```
├── README.pdf            // 本文档
├── api-doc               // java-doc api文档
└── sample                // 示例工程源码
```

FAQ

1. 人员信息表和底库表是否需要做关联删除？

不需要，删除人员信息和删除底图都是原子操作；假如需要同时删除人员信息及其底图，SDK会分别调用两次接口来完成，设备实现层无需做关联删除操作。

2. 运行时报liblinkface.so not found错误。

如果使用了SNAPSHOT包，遇到这种情况请Rebuild一下工程。

3.与LinkFace Linux 版本SDK有什么区别？

Android版本基于Linux C版本进行封装和移植，暂不支持设备数据校验码查询。

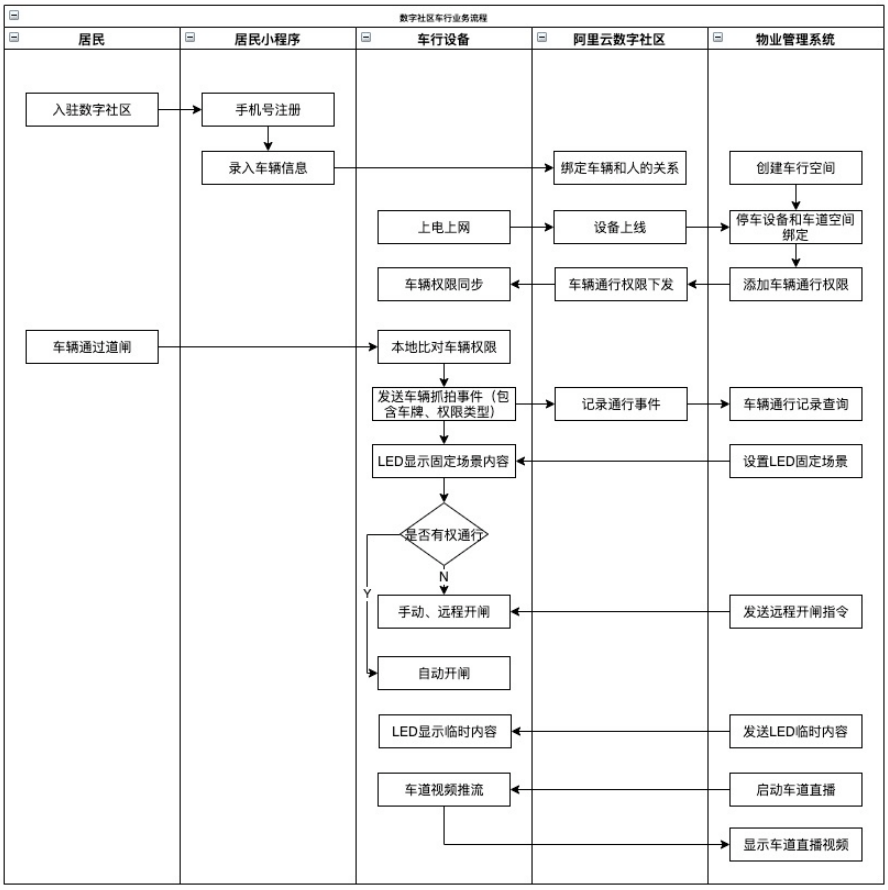
2.4. 车行接入

2.4.1. 设备端对接（C语言）

1 整体架构



2 业务流程



3 对接LinkVisual SDK

3.1 C语言SDK对接

车行设备由多功能停车摄像头、LED显示屏、道闸等设备组成，其中多功能停车摄像头负责车牌识别、车辆权限判定、云端通信，以及LED和道闸控制。多功能停车摄像头在标准摄像头功能基础上，扩展了与车行相关的服务与能力，是车行设备端对接的主要部件。

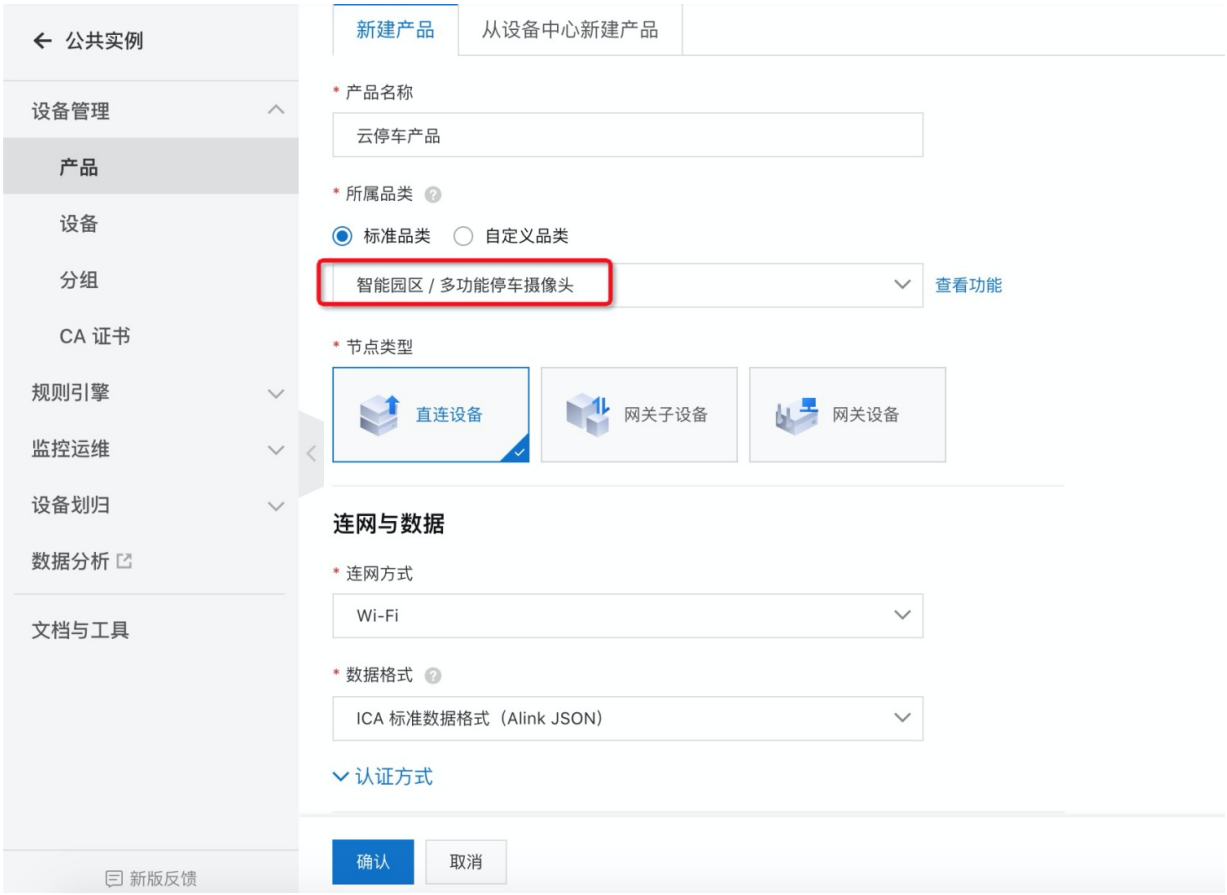
对接C语言SDK参见：https://help.aliyun.com/document_detail/108171.html

3.2 创建车行产品

1）访问地址：

https://iot.console.aliyun.com/product，

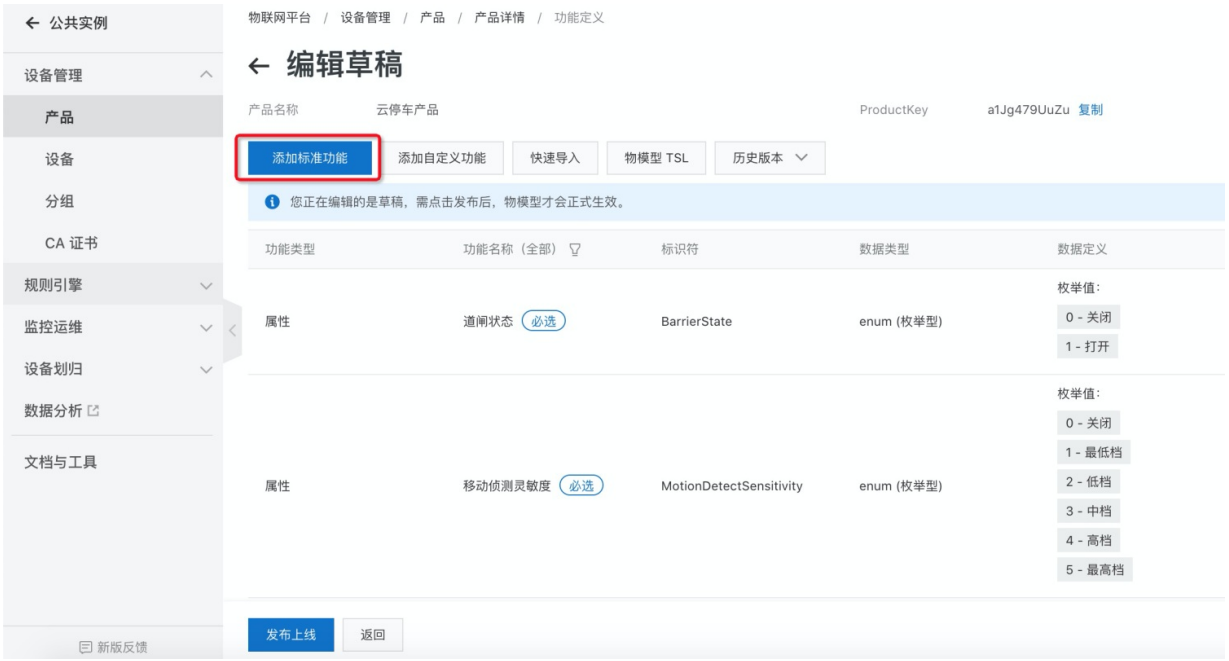
在物联网平台上创建产品，所属品类选择“智能园区/多功能停车摄像头”：



2）在产品详情页面，选择“功能定义-编辑草稿”：



3）然后在编辑草稿页面，选择“添加标准功能”：



4）在弹出的对话框中选择所需要的属性、服务、事件。例如：选择服务“下发LED数据”，事件“异常抬杆”：



5）选择完成后将物模型发布上线：

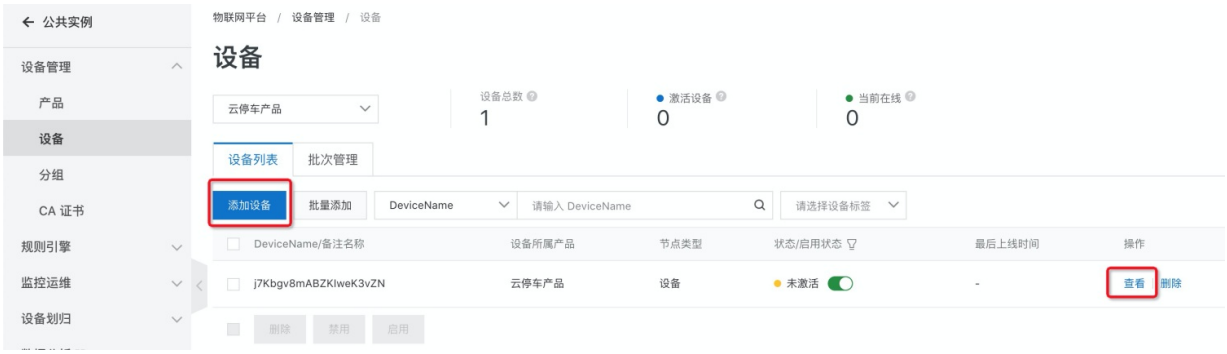


3.3 创建车行设备

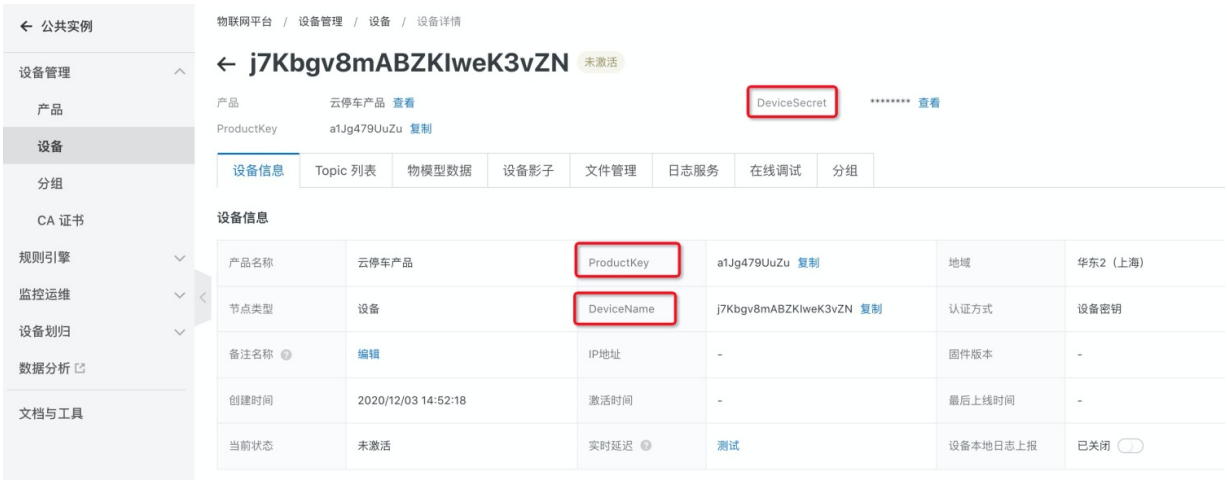
1）在产品列表中选择“设备管理”：



2）为产品添加设备，然后在设备列表中选择“查看”：



3）获得设备三元组信息：



3.4 设备连接物联网平台

复制车行设备三元组，根据以下文档所指示的步骤，将设备连入物联网平台：

https://code.aliyun.com/edward.yangx/public-docs/wikis/user-guide/linkkit/Archived/V230_Snapshot/programme/Auth_Connect#%E4%B8%80%E6%9C%BA%E4%B8%80%E5%AF%86%E7%BC%96%E7%A8%8B

3.5 设备物模型开发

根据物模型定义的属性、事件、服务，完成设备端开发。设备端物模型开发参考：

https://code.aliyun.com/edward.yangx/public-docs/wikis/user-guide/linkkit/Archived/V230_Snapshot/programme/DeviceModel_Prog

4 多功能停车摄像头物模型

与标准的“摄像头”物模型相比，多功能停车摄像头增加（或复用）了以下属性、事件、服务：

4.1 属性

名称	类型	描述
LedRowNumber	Integer	屏幕行数，只读属性，取值范围： -100 ~ 100 • 数字绝对值表示屏幕支持的行数 • 负号表示支持 换行符，调用费显屏显示服务时指定行数为1，内容为带 的字符串
LedNoticeMode	Enum	屏幕通知模式，只读属性： • 0 分离模式（支持单行刷新，语音、文字分别下发） • 1 混合模式（仅支持语音、文字同时下发）
IsSurportQrCode	Bool	是否支持二维码，只读属性： • 0 不支持 • 1 支持
GateState	Enum	道闸设备状态，只读属性： • 0 已打开 • 1 已关闭 • 2 开闸中 • 3 关闸中
VehicleState	Enum	车辆等待状态，只读属性： • 0 无车 • 1 有车 • 2 不支持
Volume	Integer	音量大小，取值范围：0 - 100

Brightness	Integer	屏幕亮度，取值范围：0 - 100

4.2 服务

4.2.1 下发车辆权限

名称	AddVehicleIOfPermission
描述	通过文件异步下发车辆权限到设备。

入参：

参数	类型	描述
VehiclePermissionUrl	String	车辆权限文件URL

出参：

参数	类型	描述
VehiclePermissionStatus	Enum	车辆权限文件下载状态： • 0 文件下载成功 • 1 文件下载失败

车辆权限通过JSON文件下发、上报。JSON文件中字段定义如下：

参数	类型	描述	是否必须
total	Integer	文件包含的记录数量	是
permList	List	车辆权限数组	是
-type	Integer	操作类型：1-增加（或覆盖），2-删除	是-下发权限时 否-返回权限时
-plateNumber	String	车牌号	是
-effectiveDate	String	生效时间（UTC）单位精确到毫秒	是-下发、返回权限 否-黑白名单
-expiryDate	String	终止时间（UTC）单位精确到毫秒	是-下发、返回权限 否-黑白名单
-permissionType	Integer	权限类型：1-普通，2-白名单，3-黑名单	是

下发车辆权限文件示例：

```
{
    "total": 100,
    "permList": [
    {
        "type": 1,
        "plateNumber": "京N1***45",
        "effectiveDate": "时间戳",
        "expiryDate": "时间戳",
        "permissionType": 1
    },
    {
        "type": 2,
        "plateNumber": "京N***67",
        "effectiveDate": "",
        "expiryDate": "",
        "permissionType": 2
    }
    ]
}
```

设备端下载并解析JSON权限文件，按照指定操作类型更新本地车辆权限数据库。

4.2.2 获取车辆权限

名称	GetVehiclePermission
描述	获取设备本地存储的车辆权限，用于校验云端与设备端权限是否一致

请求参数：无

出参：

参数	类型	描述
VehiclePermissionUrl	String	设备端权限文件URL

设备端按约定格式导出权限文件，使用OSS STS Token将文件上传到指定位置，并返回文件URL。

4.2.3 查询车辆权限下发状态

名称	GetPermissionDownloadProgress
描述	查询车辆权限下发状态

请求参数：无

出参：

参数	类型	描述
DownloadStatus	Enum	车辆权限下发状态： • 0 完成 • 1 进行中

4.2.4 下发OSS STS Token

名称	SendOssStsToken
描述	云端下发OSS STS Token到设备，允许设备使用临时访问权限上传文件

入参：

参数	类型	描述

Credentials	String	加密的STS Token证书，包含如下信息： • AccessKeyId 访问密钥标识 • AccessKeySecret 访问密钥 • StsToken 安全令牌 • Expiration 失效时间
BucketName	String	允许设备访问的OSS Bucket Name
ObjectName	String	指定设备上传文件的路径
EndPoint	String	允许设备访问的OSS接入地址

出参：

参数	类型	描述
Status	Enum	STS Token下发状态： • 0 完成 • 1 进行中

设备端通过解密STS Token证书，获取上传文件所需的权限。当证书过期后，设备端上报OSS STS Token请求事件，云端重新下发有效的STS Token证书。

4.2.5 下发LED显示数据

根据设备属性LedNoticeMode定义的模式，下发屏幕提示数据。以分离模式下发需要实现以下服务：

- NoticeBroadcast：下发提示语音
- NoticeDisplay：下发提示显示
- DefaultDisplay：下发默认显示

以混合模式下发需要实现以下服务：

- AddLedData：提示语音、提示显示同时下发

1）下发提示语音

名称	NoticeBroadcast
描述	下发提示语音信息

入参：

参数	类型	描述
VoiceContent	String	语音播报内容，最大长度不超过2048
PlaySpeed	Integer	语音播报速度，取值范围：1 - 10，步长：1

返回参数：无

2）下发提示显示

名称	NoticeDisplay
描述	下发提示显示信息

入参：

参数	类型	描述
DisplayContent	String	显示内容，最大长度不超过2048

RowNumber	Integer	屏幕行号，取值范围：1 - 100
MovingSpeed	Integer	文字移动速度，取值范围：1 - 10，步长：1
DisplayTime	Integer	显示时长，取值范围1-3600，单位：秒
MovingType	Enum	文字移动方式：0：禁止（默认）1：从右到左2：从左到右3：从下到上4：从上到下

返回参数：无

3）下发默认显示

名称	DefaultDisplay
描述	下发默认显示信息

入参：

参数	类型	描述
DisplayContent	String	显示内容，最大长度不超过2048
RowNumber	Integer	屏幕行号，取值范围：1 - 100
DefaultSituation	Enum	默认显示场景：0:空闲1:普通权限车辆、白名单车辆过车2:黑名单车辆过车3:临时车量过车4:无牌车辆过车

返回参数：无

4）混合下发LED提示数据

名称	AddLedData
描述	下发LED显示数据，控制LED在不同场景下的显示与播报内容

入参：

参数	类型	描述
LedData	JSON	以JSON描述的LED显示内容，最大长度不超过2048

出参：

参数	类型	描述
Result	Enum	LED内容解析状态： • 0 解析正常 • 1 解析异常

云端通过JSON字符串下发LED显示数据，JSON字符串内容定义如下：

属性名	类型	描述
Type	Integer	下发内容的类型： • 0 固定场景 • 1 临时场景 • 2 个性化场景

Situation	Integer	固定场景（Type=0）下的场景类型： • 0 空闲 • 1 普通权限车辆、白名单车辆过车 • 2 黑名单车辆过车 • 3 临时车量过车 • 4 无牌车辆过车
ShowTime	Integer	内容显示时长，单位为秒
AudioContextInfo	String	自定义语音播报内容
ScreenLineNum	Integer	屏幕行数
ContextList	JSONArray	屏幕内容数组，每个元素代表一行
-RunType	Integer	文字滚动方式： • 0 禁止 • 1 从右到左 • 2 从左到右 • 3 从下到上 • 4 从上到下
-ColorType	Integer	文字颜色： • 1 红色 • 2 绿色
-CustomContext	String	自定义内容，长度1-128。内容支持以下通配符： • #T 当前时间 • #t 车辆到期时间 • #p 车牌号码
-Audio	Integer	是否进行语音播报： • 0 否 • 1 是

以支持四行文字，红、绿两种颜色的LED显示器为例，下发“固定场景-空闲”时显示的内容：

```
{
      "Type": 0,
      "Situation": 0,
      "PlateNumber": "",
      "ShowTime": ,
      "AudioContextInfo": "",
      "ScreenLineNum": 4,
      "ContextList":[
          {
              "RunType": 0,
              "ColorType": 1,
              "CustomContext":"智慧停车",
              "Audio": 0
          },
          {
              "RunType": 0,
              "ColorType": 1,
              "CustomContext":"智能识别",
              "Audio": 0
          },
          {
              "RunType": 0,
              "ColorType": 1,
              "CustomContext":"一车一杆",
              "Audio": 0
          },
          {
              "RunType": 0,
              "ColorType": 2,
              "CustomContext":"#T",
              "Audio": 0
          }
      ]
}
```

4.2.6 下发二维码

名称	QrCodeDisplay
描述	下发二维码信息

入参：

参数	类型	描述
QrCode	String	二维码信息，最大长度不超过2048

返回参数：无

4.2.7 手动触发车牌识别

名称	TriggerRecognition
描述	触发设备上报智能告警事件

请求参数：无，或者返回参数：无

设备端抓拍车辆图片，并通过智能告警事件上报图片和相关信息。

4.3 事件

4.3.1 智能告警事件

名称	IntelligentAlarm
描述	上报智能告警事件，包含车辆图片和车牌识别结果等信息

与其他事件不同，设备端需要调用LV SDK提供的“

lv_post_intelligent_alarm”接口上报智能告警事件，相关参数和接口定义以LV SDK为准：

```
/* 智能事件参数集 */
typedef struct {
    lv_intelligent_event_type_e type; //智能事件类型，车辆识别为10005
    lv_media_format format;           //智能事件的媒体数据类型，图片类型为0
    struct {
        char *p;
        uint32_t len;
    } media;                          //智能事件的媒体数据,不大于1MB
    struct {
        char *p;
        uint32_t len;
    } addition_string;                //智能事件的附加字符串信息，不大于2048B
} lv_intelligent_alarm_param_s;

/* 上报智能告警事件图片、告警信息 */
int lv_post_intelligent_alarm(const lv_intelligent_alarm_param_s *param);
```

参数说明：

参数	类型	描述
type	Enum	告警类型，固定值10005（车辆识别）
format	Enum	媒体数据类型，固定值0（JPEG格式图片）
media	Struct	媒体数据，不大于1MB
addition_string	Struct	告警内容字段定义参见车辆通行记录。 以下字段不需要上报： • direction • channelSpaceId

4.3.2 OSS STS Token请求事件

名称	OssStsTokenRequest
描述	上报OSS STS Token请求事件，请求云端下发OSS STS Token

上报参数：

参数	类型	描述
RequestType	Enum	请求上传的文件类型： • 0 图片文件 • 1 文本文件 • 2 其他类型文件

4.3.3 异常抬杆事件

名称	AbnormalOpenGate
描述	当闸机非自动抬杆时，上报异常抬杆事件

上报参数：

参数	类型	描述
OpenType	Enum	抬杆类型：0 遥控器控制抬杆1 设备自身按键抬杆99 其他方式

ImagePath	String	抓拍图片地址，数据长度2048

5 通过OSS STS Token上传文件

设备端集成OSS SDK参见：https://www.alibabacloud.com/help/zh/doc-detail/32132.htm

设备端使用签名URL上传文件示例参考：https://www.alibabacloud.com/help/zh/doc-detail/32139.htm?#title-bj5-3h9-tap

2.5. 专题介绍

2.5.1. 企业支付宝账号申请

您可以通过阅读该篇文章了解支付宝企业账号如何申请

> 说明
> 企业支付宝账号是数字社区用户申请小区、物业缴费等功能的前提，所以您需要提前根据企业实际情况申请企业支付宝账号

1. 进入PC端申请，营业执照、社会信用代码、法人身份证正反面图片等企业关键信息进行注册



2. 移动端申请，进入支付宝小程序，搜索“企业管家”小程序，进行企业注册
   i. 在支付宝顶部搜索栏，搜索“企业管家”
   ii. 进入企业管家小程序，进行新企业注册
   iii. 根据您单位类型，进行单位选择
   iv. 上传相关证件信息

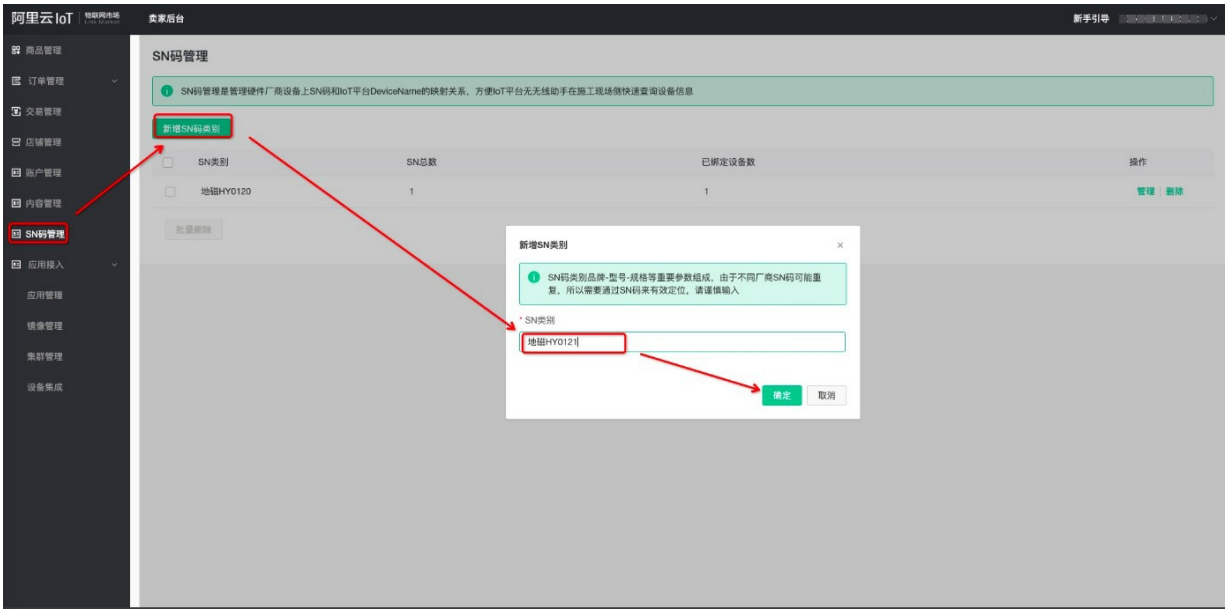


2.5.2. 门禁设备SN码管理

SN码为设备序列号，可与设备三元组进行关联，SN码管理是维护设备制造商设备SN码和物联网平台设备证书之间的映射关系。有了这类映射关系，数字社区SaaS应用中可以快速的通过SN码进行门禁设备的绑定操作。

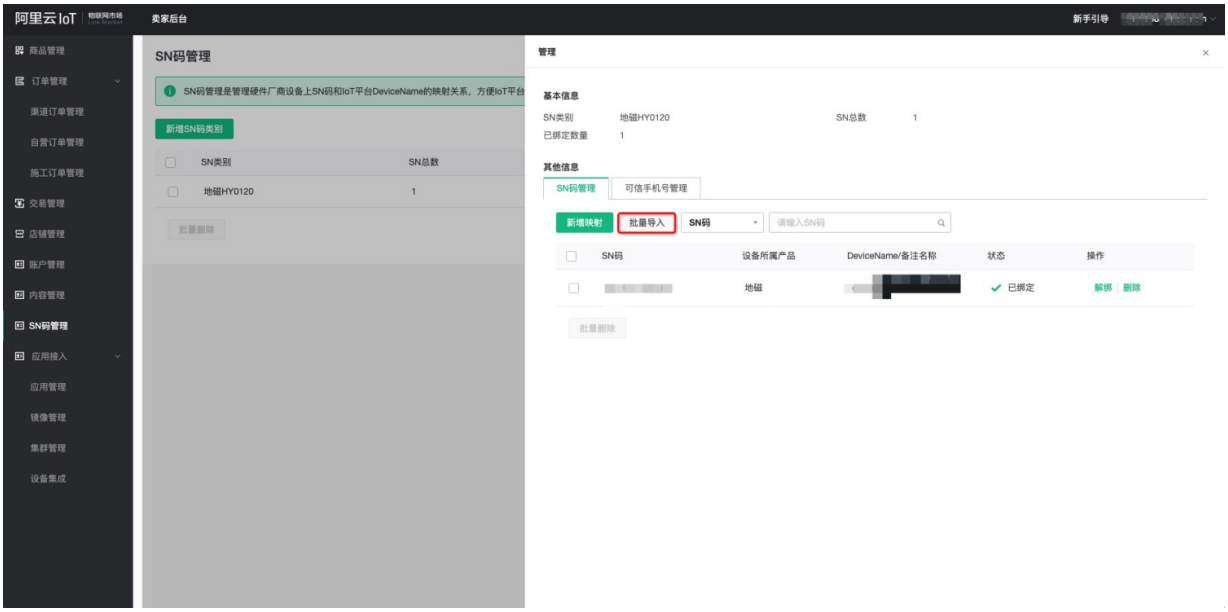
1. SN码类别创建

请登录 卖家后台 ->SN码管理->新增SN码类别（SN类别是一类设备的分类）。



2. SN码批量导入

请登录 卖家后台 ->SN码管理-> 选择具体SN类别 -> 管理 -> SN码管理 。



SN码批量导入模板

SN码批量上传模版格式为xlsx文件，每行分成四列：

VPC实例ID（vpcInstanceId）	SN码（snCode）	设备名（deviceName）	设备PK(productKey)
vpcinstance001	wang_0001	wangjing_0001	a1yKQvJ5gTo

- VPC实例ID（vpcInstanceId）：为设备所在的物联网平台实例ID，如果您的设备是公有云创建，该字段无需填写。
- SN码（snCode）： 是您展示在设备硬件上的SN码。
- 设备名（deviceName）：物联网平台创建的设备deviceName。
- 设备产品(productKey)：物联网平台创建的设备productKey。

⚠㊟注意事项：

1、其中vpcInstanceId可以为空，表示是公有云的设备，snCode, deviceName, productKey为必传。

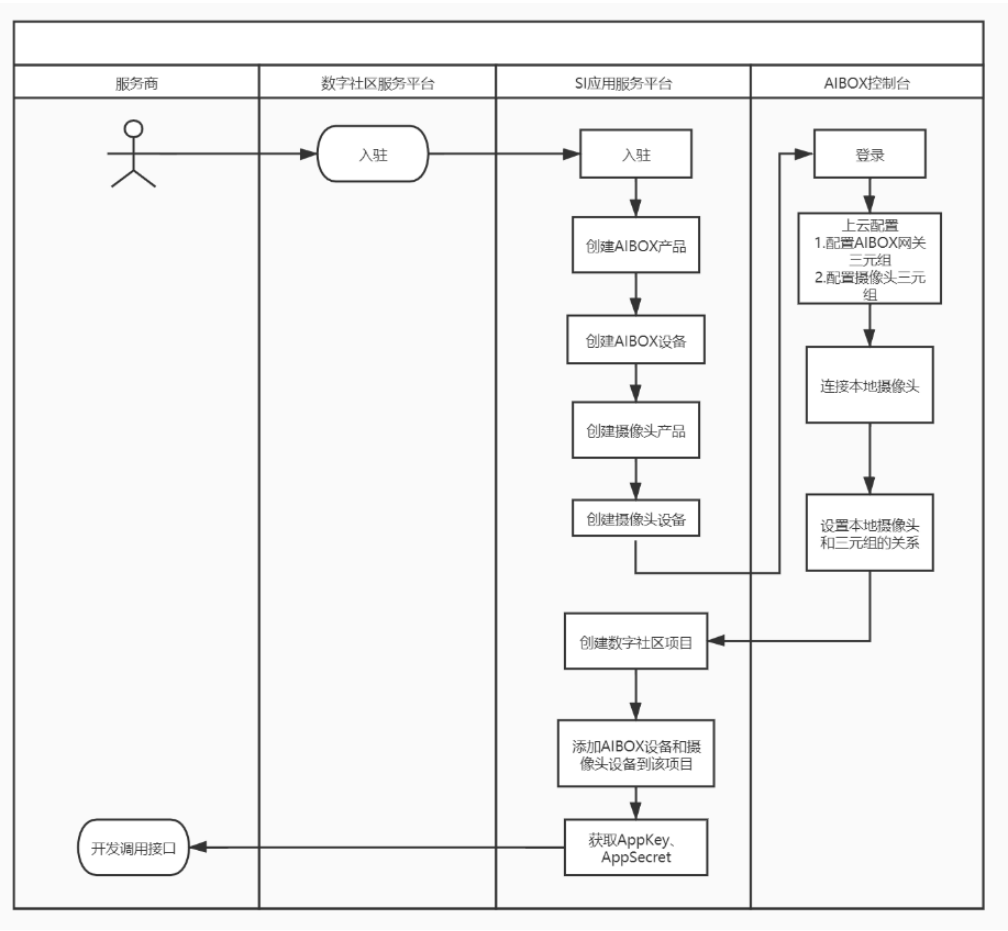
2、每次上传限制记录条数最多1000条，每条记录之间不能有空行。

3. SN码可信手机号管理

这部分您无需配置。

2.5.3. 社区AI一体机操作手册

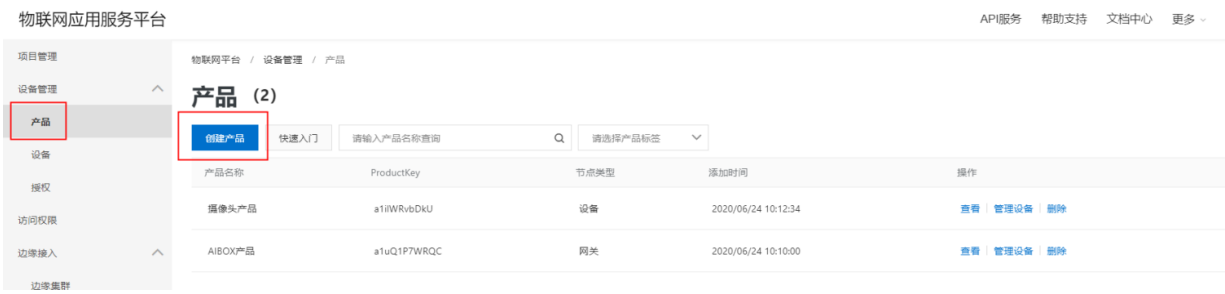
本章将为您介绍如果通过社区AI一体机和数字社区开放接口进行实施开发



1. SI应用服务平台产品设备创建

1.1 创建AIBOX产品与摄像头产品

登录阿里云应用服务平台，“设备管理 - 产品 - 创建产品”，如下图所示。



1.1.1 AIBOX产品

注意：在创建AIBOX产品时，所属品类请选择“标准品类：边缘计算/视频内容分析”，节点类型请选择“网关设备”，如下图所示。



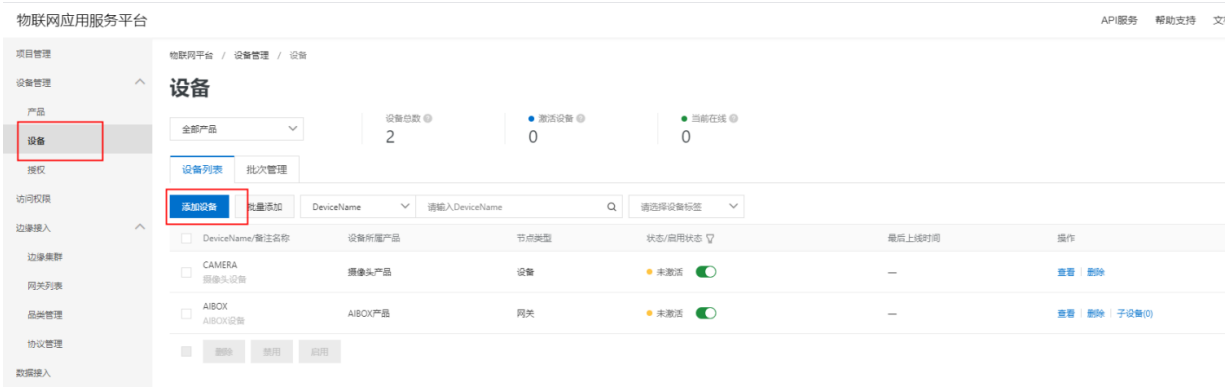
1.1.2 摄像头产品

注意：在创建摄像头产品时，所属品类请选择“标准品类：智能生活/家居安防/摄像头”，节点类型请选择“直连设备”，如下图所示。



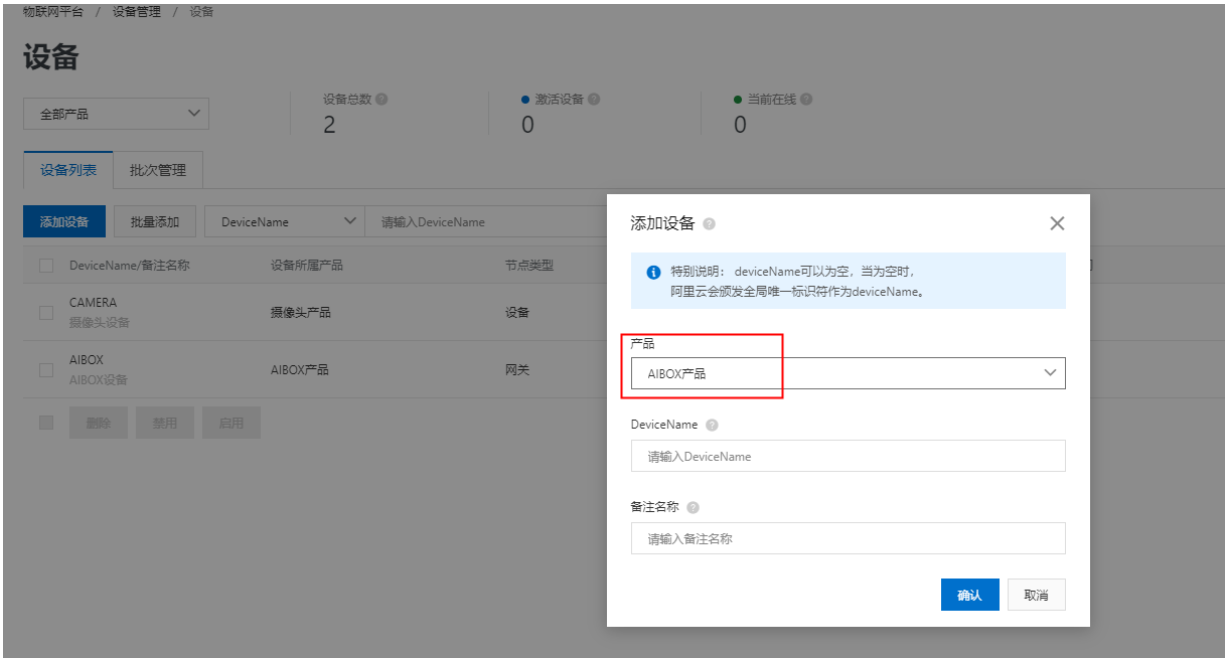
1.2 创建AIBOX设备与摄像头设备

登录阿里云应用服务平台，“设备管理 - 设备 - 添加设备”，如下图所示。



1.2.1 AIBOX设备

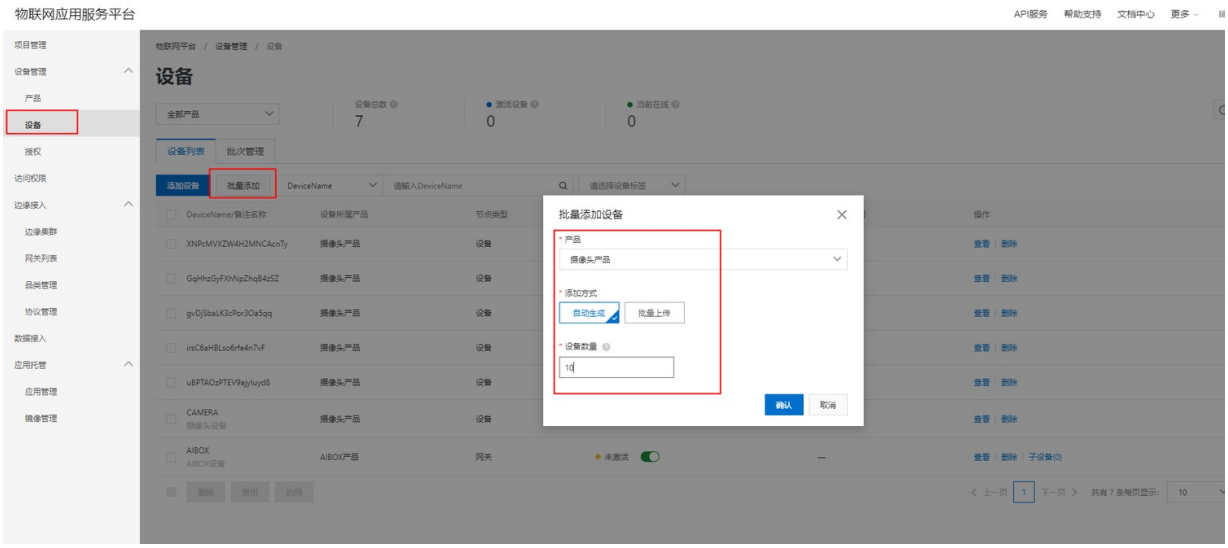
注意：在产品选择一栏，选择之前在1.1.1流程中创建的AIBOX产品，如下图所示。



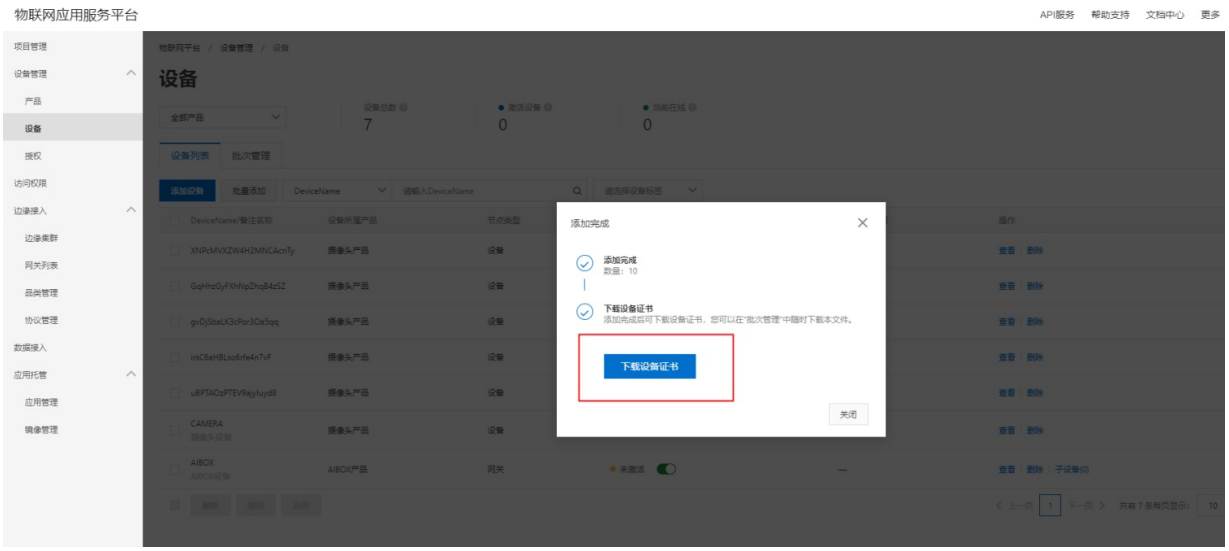
1.2.2 摄像头设备

注意：在产品选择一栏，选择之前在1.1.2流程中创建的摄像头产品。

在这里添加设备的方式有两种：添加设备（单台设备添加）和批量添加，这里我们选择批量添加，如下图所示。

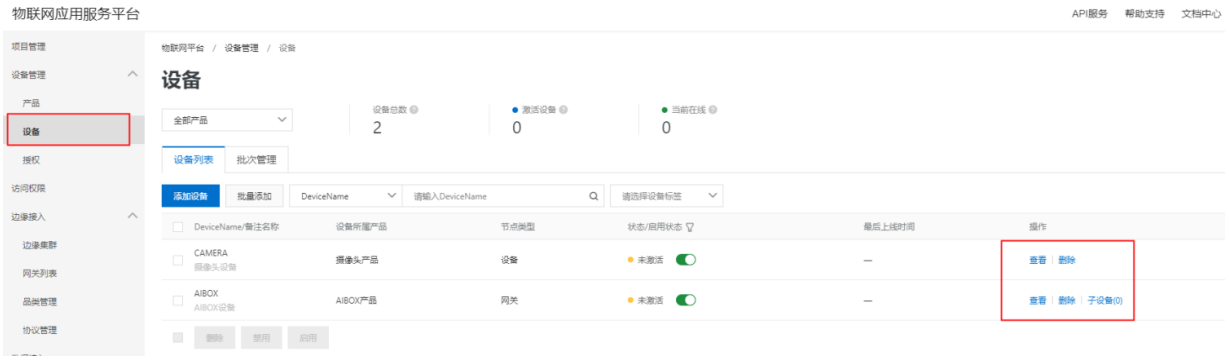


添加完成后点击“下载设备证书”（下载好的设备证书都以“.csv”结尾，例如“设备证书.csv”），如下图所示。

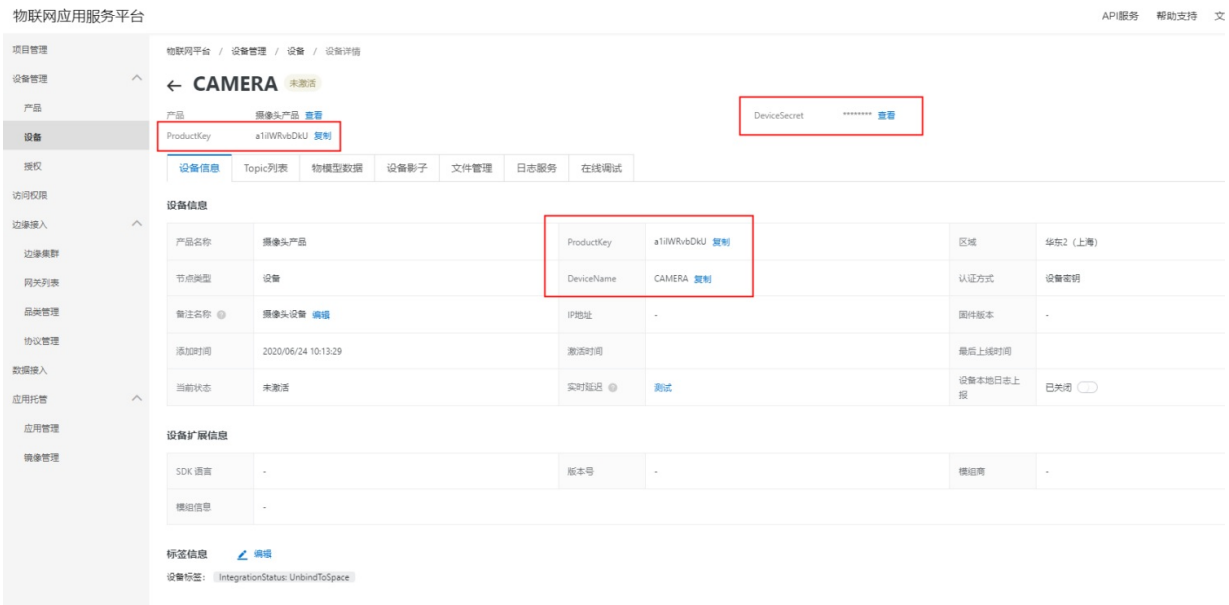


1.2.3 设备三元组介绍

在之后的流程中，会多次提及到的“设备三元组”，具体的查看方式：在“设备”中，选择想要获取设备三元组的设备，点击查看，如下图所示。



ProductKey、DeviceName、DeviceSecret三个数据即为设备唯一标识，本篇文档中统称为设备三元组，如下图所示。



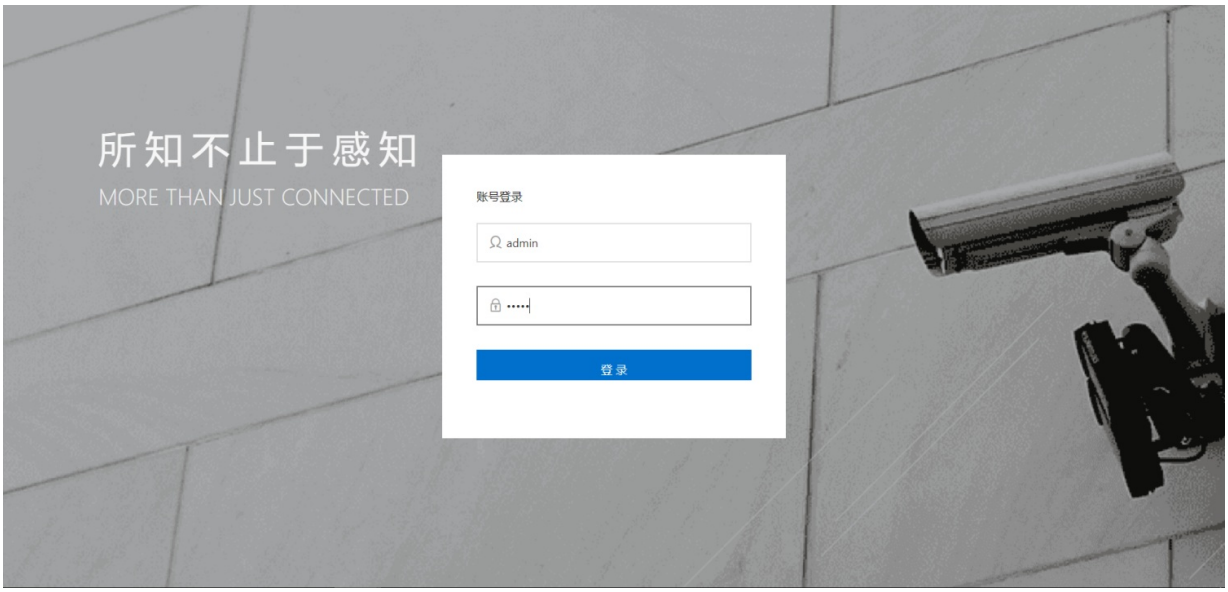
2. AIBOX控制台设备操作

2.1 登录AIBOX控制台

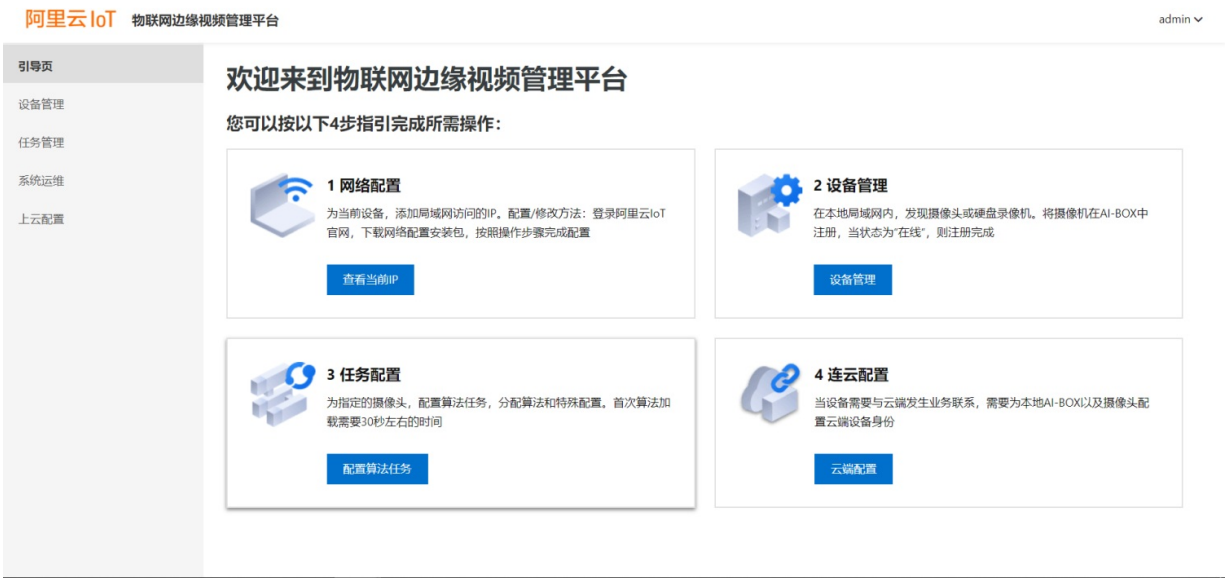
登录边缘控制台：即AIBOX的IP + 端口8080进行登录访问。

注意：主机网络需要和设备在同一个网段下。

登录的账号密码：admin/admin

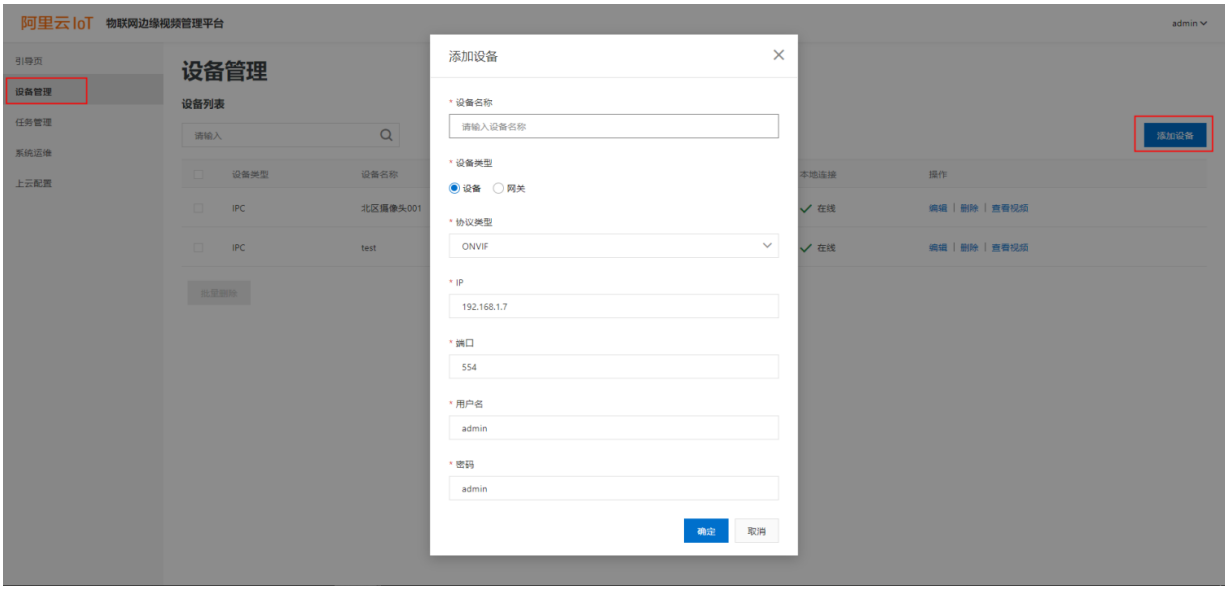


登录后可进行设备的相关配置操作，如下图所示。



2.2 添加本地摄像头

初始化的AIBOX不包含任何设备，您需要将您的IPC设备加入到AIBOX中，点击“设备管理 - 添加设备”进行设备信息的相关配置，如下图所示。



添加设备的弹出菜单，设备名称为自定义，请根据实际需要自行定义。

设备类型目前支持设备和网关两种。如您需要通过接入NVR等网关设备来接入挂载的IPC，则选择“网关”，如直接接入IPC设备则选择“设备“即可。

协议类型目前支持国标、ONVIF两种标准协议和部分摄像头厂商的私有协议，以及rtsp直接接入。

2.2.1 国标设备

请在相机配置页面中，打开GB连接，设置GB的IP地址、用户名、用户认证ID、密码信息

2.2.2 ONVIF设备

与国标一致，在摄像头参数中确认ONVIF信息，需要在控制台上填入IP、 端口、用户名和密码。



2.3 算法任务管理

AIBOX为很多算法提供了SaaS应用及相应的界面，在SaaS应用中，您可以很方便的配置不同算法独有的参数，查看算法实时和历史结果，如下图所示。



介绍如何给摄像头配置算法

2.4 设备上云

将获取的AIBOX三元组导入控制台，即可使AIBOX在云端上线。



在“上云配置 - 设置三元组”栏目中，点击“上传三元组 - 文件上传”可以导入您从云端获取到的摄像头三元组到AIBOX中（“文件上传”中的文件选择我们在之前1.2.2步骤中下载好的“设备证书.csv”即可），如下图所示。

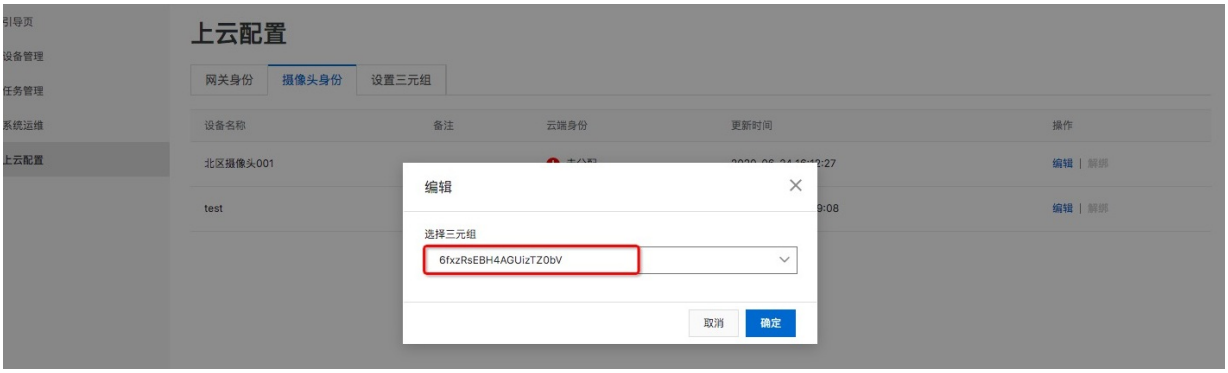


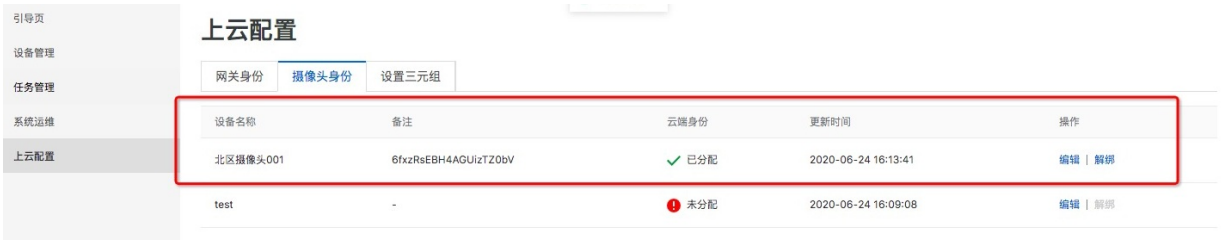
2.5 本地摄像头设备和设备三元组映射

选择“摄像头身份 - 编辑”，进行本地设备与云端设备身份映射，如下图所示。



选择相对应的云端设备三元组后点击确定进行匹配，匹配成功后云端身份显示“已分配”，如下图所示。

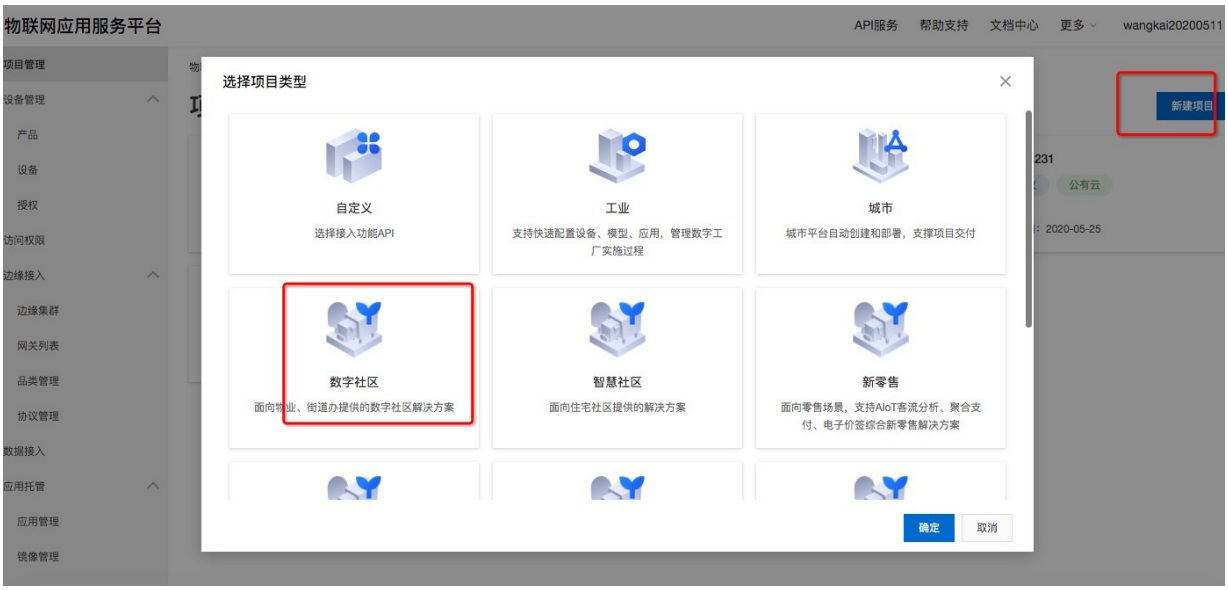




3. SI应用服务平台项目配置

3.1 创建项目

登录阿里云应用服务平台创建“数字社区”项目，如下图所示。



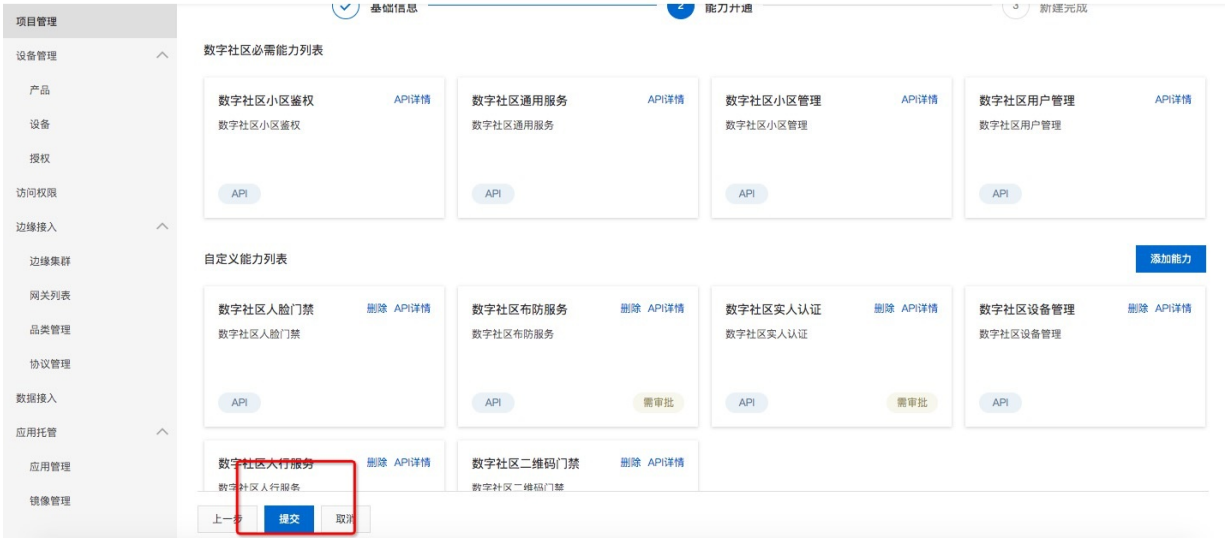
3.1.1 完善项目信息

- 基础信息：填写项目名称、部署环境（选择公有云）、项目时间、相关人员信息等，如下图所示。



- 能力：选择数字社区能力，点击提交，如下图所示。

注意：“数字社区布防服务”和“数字社区实人认证”是需要管理员审批的，需要开通请联系管理员审批。

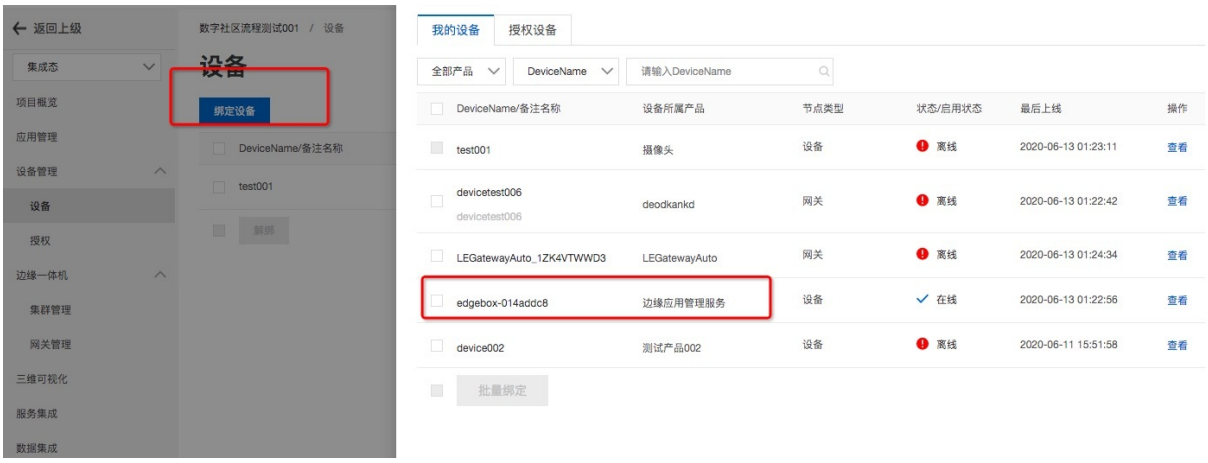


3.2 项目绑定设备

打开创建好的数字社区项目，项目状态选择“集成态”，点击“设备管理”模块，进行设备绑定，如下图所示。



点击“绑定设备”，选择需要绑定的设备，完成设备和项目的绑定，如下图所示。



3.3 获取AppKey和AppSecret

项目状态选择“开发态”，我们可以获取到“AppKey和AppSecret”，这两对数值是项目在SI应用服务平台的唯一标识，可用于之后的API接口调用的鉴权（API接口调用请看流程“5.数字社区API接口调用”），如下图所示。



4. 数字社区入驻

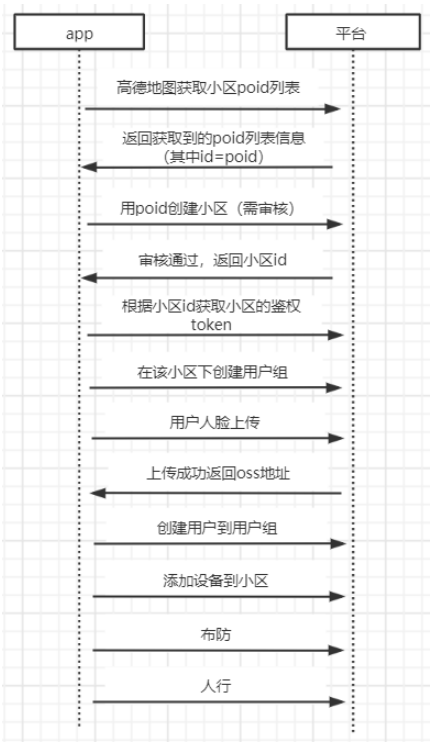
4.1 申请入驻数字社区管理平台

请注意，入驻成功后不要通过页面上的创建小区等功能进行操作，请使用接口对小区进行创建。

**



5. 数字社区API接口调用



5.1 小区管理API示例

5.1.1 获取POID列表

/solution/community/paas/poid/query	1.0.0	获取小区piod列表

示例代码：

```
/**
 * 数字社区--用户组管理--获取POID列表API
 *
 * @author wangkai
 *返回示例：  "{"parent":"B023B0A9PY","address":"紫荆花路69号","distance":[],"biz_ext":{"cost":[],"rating":[]},"pname":"浙江省","importance"
:[],"biz_type":[],"cityname":"杭州市","type":"地名地址信息;门牌信息;楼栋号","photos":[],"typecode":"190403","shopinfo":"2","poiweight":[],"c
hildtype":"308","adname":"西湖区","name":"桂花城紫云苑5幢","location":"120.091789,30.281922","tel":[],"shopid":[],"id":"B023B19DJA"}",
 */
public class CommunityGetPoidTest {
    private final static String appKey = "30239797";//项目的Appkey
      private final static String appSecret = "****************";//项目的Appsecret
    private final static String iotGatewayHost = "api.link.aliyun.com";//固定请求网关路由
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        request.putParam("keywords", "九龙");
        request.putParam("city", "香港");
        request.putParam("citylimit", true);
        request.putParam("pageSize", 20);
        request.putParam("pageNo", 5);
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/poid/query", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
```

返回示例：

其中id的返回值即为piod。

```
⊟{
    "code":200,
    "data":⊟[
        "{"parent":"B073C045CN","address":"九龙城文成街26号文苑楼前座15字楼(15/F)","distance":
[],"biz_ext":{"cost":[],"star":
[],"rating":"3.7","lowest_price":"368.00","hotel_ordering":"0"},"pname":"香港特别行政
区","importance":[],"biz_type":"hotel","cityname":"香港特别行政区","type":"住宿服务;宾馆酒店;宾馆酒
店","photos":[{"provider":[],"title":
[],"url":"http://store.is.autonavi.com/showpic/3cae9c77e6ef50bd3ce7be9b25aa2ff1"},
{"provider":[],"title":
[],"url":"http://store.is.autonavi.com/showpic/b357310b426d30bd596b4244e04d9f2b"},
{"provider":[],"title":
[],"url":"http://store.is.autonavi.com/showpic/aa94ac89ed8e32df3a8904710a0c1c4d"}],"typecode"
:"100100","shopinfo":"0","poiweight":[],"childtype":"202","adname":"油尖旺区","name":"九龙站宾
馆","location":"114.171102,22.304353","tel":"00852-28996579;00852-89303065","shopid":
[],"id":"B0FFF0EG44"}",
        "{"parent":[],"address":"九龙道3a号附近","distance":[],"biz_ext":{"cost":[],"star":
[],"rating":[],"lowest_price":[],"hotel_ordering":[]},"pname":"香港特别行政区","importance":
[],"biz_type":"hotel","cityname":"香港特别行政区","type":"住宿服务;宾馆酒店;宾馆酒店","photos":
[{"provider":[],"title":
[],"url":"http://store.is.autonavi.com/showpic/4946d3da1d044460d12801b9565da520"}],"typecode"
:"100100","shopinfo":"0","poiweight":[],"childtype":[],"adname":"深水埗区","name":"桃源宾馆(九龙
道)","location":"114.167558,22.331475","tel":[],"shopid":[],"id":"B073C03KBP"}",
```


5.1.2 创建小区

/solution/community/paas/community/apply	1.0.0	创建小区

示例代码：

```
/**
 * 创建小区【需审核】
 */
public class CommunityApplyTest {
    private final static String appKey = "30239797";
    private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        request.putParam("poid", "B0FFF0EG44");
        request.putParam("name", "九龙站宾馆");
        request.putParam("address", "金龙楼(九龙城道)");
        request.putParam("serviceStartTime", "2020-06-18");
        request.putParam("serviceEndTime", "2020-07-18");
        request.putParam("certificate", "http://store.is.autonavi.com/showpic/b357310b426d30bd596b4244e04d9f2b");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/community/apply", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
```

返回示例：


```
⊟{
    "code":200,
    "data":⊟{
        "communityId":"1504"
    },
    "id":"61e2fd81-71d0-4079-b4a9-8638f07c7dd8"
}
```


5.1.3 获取小区列表

/solution/community/paas/community/query	1.0.0	获取小区列表

示例代码：

```
/**
 * 获取小区列表
 */
public class CommunityQueryTest {
    private final static String appKey = "30239797";
    private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        //request.putParam("projectId", "a124sMKXmb6CWBL2");
        //request.putParam("applyStatus", "2");
        //request.putParam("pageSize", 20);
        //request.putParam("pageNo", 5);
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/community/query", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
```

返回示例：


```
{
    "code":200,
    "data":{
        "total":1,
        "data":[
            {
                "name":"香港/九龙的士",
                "communityId":1504,
                "applyStatus":2,
                "projectId":"a124v7PyOenhoBB1"
            }
        ],
        "pageNo":1,
        "pageSize":20
    },
    "id":"f45ae961-900e-4e67-9d8a-8d64355c7870"
}
```


5.1.4 查询小区信息【包括审核状态】

/solution/community/paas/community/detail/get	1.0.0	获取小区详情

示例代码：

```
/**
 * 查询小区信息
 */
public class CommunityGetDetailTest {
    private final static String appKey = "30239797";
    private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        request.putParam("communityId", "1504");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/community/detail/get", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
}
```

返回示例：

```
{
    "code":200,
    "data":{
        "serviceEndTime":"2020-07-18",
        "spaceId":"0d015edd270b4ca0b9f52ada1760738a",
        "address":"九龙城文成街26号文苑楼前座15字楼(15/F)",
        "serviceStartTime":"2020-06-18",
        "name":"香港/九龙的士",
        "communityId":1504,
        "poiCode":"B0FFI7FNTK",
        "applyStatus":2
    },
    "id":"66eb0300-afe2-4472-b896-4a355f4d5eb8"
}
```


5.1.5 根据设备查询小区信息

/solution/community/paas/device/community/get	1.0.0	根据项目id获取

示例代码：

```
/**
 * 根据设备查询小区信息
 */
public class DeviceCommunityGetTest {
    private final static String appKey = "30239797";
    private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
       String deviceIdentityDTO="{'type' : 'PK_AND_DN','iotId' : '','productKey' : 'a12BH8167id' ,'deviceName' : 'edgebox-014addc8'}";
        request.putParam("deviceIdentity", JSON.parse(deviceIdentityDTO));
        System.out.println(request);
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/device/community/get", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
}
```

返回示例：


```
{
    "code":200,
    "data":{
        "serviceEndTime":"2020-07-18",
        "spaceId":"0d015edd270b4ca0b9f52ada1760738a",
        "address":"九龙城文成街26号文苑楼前座15字楼(15/F)",
        "serviceStartTime":"2020-06-18",
        "name":"香港/九龙的士",
        "communityId":1504,
        "poiCode":"B0FFI7FNTK",
        "applyStatus":2
    },
    "id":"d0e90aa4-6d8a-4752-ba66-6a607f0b7cf6"
}
```


5.2 数字社区小区鉴权API

5.2.1 获取小区授权token

/solution/community/paas/token/get	POST	1.0.0	获取云端资源Token；该接口对于同一个res（小区id，参见请求参数说明）在token失效前仅需要调用一次，如果再次调用，则会生成新的cloudToken，并会导致之前的cloudToken失效

示例代码：

```
/**
 * 获取小区授权token
 */
public class PaasTokenGetTest {
    private final static String appKey = "30239797";
    private final static String appSecret = "****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        Long communityId= Long.valueOf(1504);
        request.putParam("communityId", communityId);
        //request.putParam("appid", "12p3ii-4i1-49-1230");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/token/get", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
}
```

返回示例：


```
{
    "code":200,
    "data":{
        "expireIn":7200000,
        "cloudToken":"db004377afcd46e8a71f043b6971daa6"
    },
    "id":"7e0dbf67-e855-4bac-9a4c-998d01937427"
}
```


5.2.2 刷新小区授权token

/solution/community/paas/token/refresh	1.0.0	刷新云端Token

请求示例：

```
/**
 * 刷新小区授权token
 */
public class PaasTokenRefreshTets {
    private final static String appKey = "30239797";
     private final static String appSecret = "****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        request.putParam("cloudToken", "db004377afcd46e8a71f043b6971daa6");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/token/refresh", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
```

返回示例：


```
{
    "code":200,
    "data":{
        "expireIn":7200000,
        "cloudToken":"f5b30f872e8046e3b931a7575e1c433d"
    },
    "id":"14dea10f-c014-462f-aca5-9ecdd632f49c"
}
```

5.3 图片上传API

5.3.1 获取图片上传ticket

/solution/community/oss/ticket/get	1.0.1	获取oss上传图片ticket

示例代码：

```
/**
 * 数字社区--用户组管理--新建用户组API
 *
 * @author wangkai
 */
public class AddGroupTest {
    private final static String appKey = "30239797";
     private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.1");
        request.putParam("type", "picture");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/oss/ticket/get", request, true);
        JSONObject data = JSON.parseObject(new String(response.getBody(), StandardCharsets.UTF_8));
        JSONObject keys = data.getJSONObject("data");
    }
}
```

返回示例：

其中data的值即为返回的ticket信息

```
{"accessId":"LTAI7HWGDtXi9VqD","signature":"GWMUuwa0JJy6TdHjlvr+wz+RNjA=","expire":1594184353,"host":"iotx-homelink-aquarius-prd.oss-cn-hangzhou.aliyuncs.com","dir":"4AF52878CE8B4AC5A8F9D8CCF2902C8B/picture/1c1145e0ebee47e4a2c9160f2f27cdca","policy":"eyJleHBpcmF0aW9uIjoiMjAyMC0wNy0wOFQwNDo1OToxMy43NjZaIiwiY29uZGl0aW9ucyI6W1siY29udGVudC1sZW5ndGgtcmFuZ2UiLDAsMTA0ODU3NjAwMF0sWyJzdGFydHMtd2l0aCIsIiRrZXkiLCI0QUY1Mjg3OENFOEI0QUM1QThGOUQ4Q0NGMjkwMkM4Qi9waWN0dXJlLzFjMTE0NWUwZWJlZTQ3ZTRhMmM5MTYwZjJmMjdjZGNhLyJdXX0="}
```

获取oss上传的授权后，采用http的post请求向oss上传图片文件

```
public static String upload2(File file, String host, String dir, String policy,
                             String accessKeyId, String signature)
            throws ClientProtocolException, IOException {
        CloseableHttpClient httpClient = HttpClientBuilder.create().build();
        CloseableHttpResponse httpResponse = null;
        RequestConfig requestConfig = RequestConfig.custom().setConnectTimeout(200000)
                .setSocketTimeout(200000000).build();
        HttpPost httpPost = new HttpPost(host);
        httpPost.setConfig(requestConfig);
        MultipartEntityBuilder multipartEntityBuilder = MultipartEntityBuilder.create();
        multipartEntityBuilder.setCharset(Charset.forName("UTF-8"));
        multipartEntityBuilder.setMode(HttpMultipartMode.BROWSER_COMPATIBLE);
        //处理文件名乱码
        ContentType contentType = ContentType.create(HTTP.PLAIN_TEXT_TYPE, HTTP.UTF_8);
        multipartEntityBuilder.addTextBody("key", dir + "/" + file.getName(), contentType);
        multipartEntityBuilder.addTextBody("policy", policy);
        multipartEntityBuilder.addTextBody("OSSAccessKeyId", accessKeyId);
        multipartEntityBuilder.addTextBody("success_action_status", "200");
        multipartEntityBuilder.addTextBody("callback", "");
        multipartEntityBuilder.addTextBody("signature", signature);
        multipartEntityBuilder.addBinaryBody("file", file);
        HttpEntity httpEntity = multipartEntityBuilder.build();
        httpPost.setEntity(httpEntity);
        httpResponse = httpClient.execute(httpPost);
        HttpEntity responseEntity = httpResponse.getEntity();
        int statusCode = httpResponse.getStatusLine().getStatusCode();
        if (statusCode == 200) {
            BufferedReader reader = new BufferedReader(
                    new InputStreamReader(responseEntity.getContent()));
            StringBuffer buffer = new StringBuffer();
            String str = "";
            while (!StringUtils.isEmpty(str = reader.readLine())) {
                buffer.append(str);
            }
            System.out.println(buffer.toString());
        }
        httpClient.close();
        if (httpResponse != null) {
            httpResponse.close();
        }
        // 返回oss中的目录文件地址
        return dir + "/" + file.getName();
    }
```

5.4 数字社区用户管理API

5.4.1 新建用户组

/solution/community/paas/user/group/add	1.0.0	新建用户组

示例代码：

```
/**
 * 数字社区--用户组管理--新建用户组API
 *
 * @author wangkai
 */
public class AddGroupTest {
    private final static String appKey = "30239797";
     private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        Long communityId= Long.valueOf(1504);
        request.setCloudToken("3ffd38999f0649dea102affc9397f4ee");
        request.putParam("communityId", communityId);
        request.putParam("userGroupName", "王凯");
        request.putParam("userGroupType", "RESIDENT");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/user/group/add", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(), "UTF-8"));
    }
}
```

返回示例：

其中data的值即为userGroupId（用户组id）

```
{
    "code":200,
    "data":"0vMh1RKgJQqkZtsu1ZBTa0jwG3KAntp5",
    "id":"468aa559-056f-4c44-96c9-643a23225e00"
}
```

5.4.2 更新用户组

/solution/community/paas/user/group/update	1.0.0	更新用户组

请求示例：

```
/**
 * 更新用户组
 */
public class UpdateGroupTest {
    private final static String appKey = "30239797";
     private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        request.setCloudToken("3ffd38999f0649dea102affc9397f4ee");
        request.putParam("userGroupName", "王凯02");
        request.putParam("userGroupId", "0vMh1RKgJQqkZtsu1ZBTa0jwG3KAntp5");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/user/group/update", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(), "UTF-8"));
    }
```

返回示例：

```
{
    "code":200,
    "data":"",
    "id":"885276b3-f7f7-4a6d-85a5-efee4eaa98b3"
}
```


5.4.3 删除用户组

/solution/community/paas/user/group/**delete**	1.0.0	删除用户组

请求示例：

```
/**
 * 删除用户组
 */
public class DeleteGroupTest {
    private final static String appKey = "30239797";
    private final static String appSecret = "****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        request.setCloudToken("3ffd38999f0649dea102affc9397f4ee");
        request.putParam("userGroupId", "0vMh1RKgJQqkZtsu1ZBTa0jwG3KAntp5");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/user/group/delete", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(), "UTF-8"));
    }
}
```

返回示例：


```
{
    "code":200,
    "data":"",
    "id":"885276b3-f7f7-4a6d-85a5-efee4eaa98b3"
}
```


5.4.4 查询用户组

/solution/community/paas/user/group/**query**	1.0.0	查询用户组

请求示例：

```
/**
 * 删除用户组
 */
public class DeleteGroupTest {
    private final static String appKey = "30239797";
   private final static String appSecret = "****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        request.setCloudToken("3ffd38999f0649dea102affc9397f4ee");
        request.putParam("userGroupId", "0vMh1RKgJQqkZtsu1ZBTa0jwG3KAntp5");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/user/group/delete", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(), "UTF-8"));
    }
}
```

返回示例：


```
{
    "code":200,
    "data":{
        "total":7,
        "data":[
            {
                "userGroupId":"t3M3Bc2V8CctMnhy5aiN3C4b6UWkpVQu",
                "userGroupType":"RESIDENT",
                "userGroupName":"用户组测试02",
                "communityId":1504
            },
            {
                "userGroupId":"ZMGlEBrFLgusNRr6357o6Ogv14AIyad9",
                "userGroupType":"RESIDENT",
                "userGroupName":"用户组测试01",
                "communityId":1504
            },
            {
                "userGroupId":"0vMh1RKgJQqkZtsu1ZBTa0jwG3KAntp5",
                "userGroupType":"RESIDENT",
                "userGroupName":"王凯02",
                "communityId":1504
            },
            {
                "userGroupId":"noHvFw7DBfiTnzQccNLjPZH1yga1np72",
                "userGroupType":"BLACKLIST",
                "userGroupName":"黑名单"
```


5.4.5 添加用户

path	版本	描述
/solution/community/paas/user/add	1.0.0	添加用户

请求示例：

```
/**
 * 添加用户
 */
public class UserAddTest {
    private final static String appKey = "30239797";
    private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        request.setCloudToken("3ffd38999f0649dea102affc9397f4ee");
        request.putParam("userGroupId", "0vMh1RKgJQqkZtsu1ZBTa0jwG3KAntp5");
        request.putParam("userName", "李白");
        request.putParam("mobile", "186****81");
        //request.putParam("idNo", "xzxxxx");
        request.putParam("sex", 2);
        request.putParam("age", 22);
        //  request.putParam("faceImageUrl", "https://iotx-homelink-aquarius-prd.oss-cn-hangzhou.aliyuncs.com/59FBB8C4DE304F29A1626B96B2
DE7096/faceimage/43339b66b592413681c2ece668e6587e/lALPD3lGqxlIEyrM0s0BJg_294_210.png_720x720g.jpg");
    request.putParam("faceImageUrl", "59FBB8C4DE304F29A1626B96B2DE7096/faceimage/43339b66b592413681c2ece668e6587e/lALPD3lGqxlIEyrM0s0BJg
_294_210.png_720x720g.jpg");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/user/add", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(), "UTF-8"));
    }
}
```

返回示例：


```
{
    "code":200,
    "data":{
        "identityId":"50720A366a7cf95c2c4b5d89c31cba1adc410163"
    },
    "id":"11e4236b-c7e0-4044-859f-11f2ad1116c9"
}
```

5.4.6 用户组添加用户

/solution/community/paas/user/group/member/add	1.0.0	用户组添加用户

请求示例：

```
/**
 * 用户组添加用户
 */
public class GroupMemberAddTest {
    private final static String appKey = "30239797";
    private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        request.setCloudToken("cfc08a3e8ff446548171b9ae48dd58a8");
        request.putParam("userGroupId", "0vMh1RKgJQqkZtsu1ZBTa0jwG3KAntp5");
        request.putParam("identityId", "50cb0A852dd16143fe34294606f1dbee16e40006");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/user/group/member/add", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(), "UTF-8"));
    }
}
```

返回示例：


```
{
    "code":200,
    "data":{
        "identityId":"504b0A8c40c8f11add45589ec7753289a91e3579",
        "faceId":"c78cba31886745b99929d8831d316327"
    },
    "id":"9834715c-a557-4c3e-871a-4e027095475a"
}
```


5.4.7 小区用户查询

/solution/community/paas/user/query	1.0.0	小区用户查询

请求示例：

```
/**
 * 小区用户查询
 */
public class UserQueryTest {
    private final static String appKey = "30332082";
    private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        request.putParam("communityId", 1504L);
        request.setCloudToken("65252cabdaa14cb19cf004f9b79996dd");
        //request.putParam("identityId", "141241");
        //request.putParam("faceId", "3423463fss");
        //request.putParam("userName", "桂花城");
        //request.putParam("faceImageUrl", "3242342");
        //request.putParam("idNo", "1321414124");
        //request.putParam("age", 23);
        //request.putParam("sex", 1);
        //request.putParam("mobile", "186*****801");
        request.putParam("pageSize", 20);
        request.putParam("pageNo", 1);
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/user/query", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(), "UTF-8"));
    }
}
```

返回示例：


```
{
    "code":200,
    "data":{
        "total":5,
        "data":[
            {
                "identityId":"50800A187b22c53ca3ca80a987e6c3b58c5b8152",
                "sex":1,
                "mobile":"18658822807",
                "userName":"薇思3",
                "age":25
            },
            {
                "userGroups":[
                    {
                        "userGroupId":"0vMh1RKgJQqkZtsu1ZBTa0jwG3KAntp5",
                        "userGroupType":"RESIDENT",
                        "userGroupName":"王凯02"
                    }
                ],
                "identityId":"50cb0A852dd16143fe34294606f1dbee16e40006",
                "sex":1,
                "mobile":"18658822804",
                "userName":"薇思",
                "age":25
            },
```


5.5 数字社区设备API

5.5.1 添加设备到小区

/solution/community/paas/device/add	1.0.0	添加设备到小区

请求示例：

```
/**
 * 添加设备到小区
 */
public class DeviceAddTest {
    private final static String appKey = "30239797";
    private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        Long communityId= Long.valueOf(1504);
        request.setCloudToken("cfc08a3e8ff446548171b9ae48dd58a8");
        request.putParam("communityId", communityId);
        String deviceIdentity="{'type' : 'PK_AND_DN','productKey' : 'a12BH8167id' ,'deviceName' : 'edgebox-014addc8'}";
        request.putParam("deviceIdentity", JSON.parse(deviceIdentity));
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/device/add", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
}
```

返回示例：


```
{
    "code":200,
    "data":{

    },
    "id":"11e4236b-c7e0-4044-859f-11f2ad1116c9"
}
```


5.5.2 小区设备列表

/solution/community/paas/device/query	1.0.0	小区设备列表

请求示例：

```
/**
 * 小区设备列表
 */
public class DeviceQueryTest {
    private final static String appKey = "30239797";
   private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        Long communityId= Long.valueOf(1504);
        request.setCloudToken("cfc08a3e8ff446548171b9ae48dd58a8");
        request.putParam("communityId", communityId);
        request.putParam("pageNo", 5);
        request.putParam("pageSize", 20);
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/device/query", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
}
```

返回示例：

```
{
    "code":200,
    "data":{
        "total":6,
        "data":[
            {
                "iotId":"WzXJv02NLMwDWr1PO3g5000000",
                "lastOnlineTime":"2020-06-23 09:16:05",
                "thingType":"DEVICE",
                "productKey":"a1uXB4wUvk0",
                "deviceName":"WzXJv02NLMwDWr1PO3g5",
                "productName":"摄像头",
                "status":1
            },
```


5.5.3 租户设备列表

/solution/community/paas/tenant/device/query	1.0.0	租户设备列表

请求示例：

```
/**
 *租户设备列表
 */
public class TenantDeviceQueryTest {
    private final static String appKey = "30239797";
   private final static String appSecret = "****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        request.putParam("queryType", "OWNED");
        request.setCloudToken("cfc08a3e8ff446548171b9ae48dd58a8");
        request.putParam("pageNo", 5);
        request.putParam("pageSize", 20);
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/tenant/device/query", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
}
```

返回示例：


```
{
    "code":200,
    "data":{
        "total":6,
        "data":[
            {
                "iotId":"WzXJv02NLMwDWr1PO3g5000000",
                "lastOnlineTime":"2020-06-23 09:16:05",
                "thingType":"DEVICE",
                "productKey":"a1uXB4wUvk0",
                "deviceName":"WzXJv02NLMwDWr1PO3g5",
                "productName":"摄像头",
                "status":1
            },
            {
                "iotId":"Qq3NzXSLSdIEJULVx6ihz20000",
                "lastOnlineTime":"2020-06-22 20:49:55",
                "thingType":"DEVICE",
                "productKey":"a1GxXQmp1tF",
                "deviceName":"Qq3NzXSLSdIEJULVx6ih",
                "productName":"deodkankd",
                "status":1
            },
            {
                "iotId":"dGemEt4q7SgO213KgjTBz20000",
                "lastOnlineTime":"2020-06-13 01:23:11",
```


5.5.4 查询推流地址

/vision/stream/query	2.0.0	查询推流地址

请求示例：

```
/**
 * 查询推流地址
 */
public class VisionStreamQueryTest {
    private final static String appKey = "30239797";
    private final static String appSecret = "****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("2.0.0");
        request.setCloudToken("cfc08a3e8ff446548171b9ae48dd58a8");
        request.putParam("iotId", "23121421421");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/vision/stream/query", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
}
```

返回示例：

```
{
    "code":200,
    "data":{
        "path":"rtmp://106.15.173.203:8000/live?
token=195ffc8d811442cf8aa6703bb5e42b98&session=a003b6b902324807911781f34b29057d/z5JH-
HnV6q9tzaYJWyY_oLFJ65iZqeKcd5eB6hKEMuY_0"
    },
    "id":"20c80b63-fcb6-4dee-8a12-c67e4aa1b562"
}
```

可点击key和val

5.5.5 获取录像记录

/vision/record/query	2.0.0	获取录像记录

请求示例：

```
/**
 * 获取录像记录
 */
public class VisionRecordQueryTest {
    private final static String appKey = "30239797";
  private final static String appSecret = "****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("2.0.0");
        request.setCloudToken("cfc08a3e8ff446548171b9ae48dd58a8");
        request.putParam("iotId", "23121421421");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/vision/record/query", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
}
```

返回示例：

5.5.6 获取历史文件播放地址

/vision/vod/getbyfilename	2.0.0	获取录像记录播放地址

请求示例：

```
/**
 * 获取历史文件播放地址
 */
public class VisionRecordQueryTest {
    private final static String appKey = "30239797";
     private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("2.0.0");
        request.setCloudToken("cfc08a3e8ff446548171b9ae48dd58a8");
        request.putParam("iotId", "23121421421");
        request.putParam("fileName", "xxxx1231jjadxx");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/vision/record/query", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(),
            "UTF-8"));
    }
}
```

返回示例：

5.6 数字社区布防服务API

5.6.1 AIBOX用户组布防

/solution/community/paas/user/group/deploy	1.0.0	用户组布防

请求示例：

```
/**
 * AIBOX用户组布防
 */
public class deployTest {
    private final static String appKey = "30332082";
    private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        //request.setCloudToken("9e3c9efa1f194d11bc20872daaeccd1d");
        request.putParam("userGroupId", "0vMh1RKgJQqkZtsu1ZBTa0jwG3KAntp5");//
        request.putParam("iotId", "Tr9WBsRq15TmnhYUFV8v000000");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/user/group/deploy", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(), "UTF-8"));
    }
}
```

返回示例：

```
{
    "code":200,
    "data":{

    },
    "id":"11e4236b-c7e0-4044-859f-11f2ad1116c9"
}
```

5.6.2 小区布防记录查

请求方式	post
接口版本	1.0.0

请求示例：

```
/**
 * 小区布防记录查询
 */
public class queryTest {
    private final static String appKey = "30332082";
   private final static String appSecret = "****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        //request.setCloudToken("05647c0487134eeeb64752a2115c8a62");
        //request.putParam("userGroupId", "0vMh1RKgJQqkZtsu1ZBTa0jwG3KAntp5");
        request.putParam("communityId", 1504L);
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/solution/community/paas/user/group/deploy/query", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(), "UTF-8"));
    }
}
```

返回示例：


```
⊟{
    "code":200,
    "data":⊟{
        "total":1,
        "data":⊟[
            ⊟{
                "userGroupId":"0vMh1RKgJQqkZtsu1ZBTa0jwG3KAntp5",
                "schedule":100,
                "iotId":"Tr9WBsRq15TmnhYUFV8v000000",
                "userGroupType":"RESIDENT",
                "userGroupName":"王凯02",
                "deployTime":"1592908061885",
                "communityId":1504,
                "deployStatus":"deployed"
            }
        ],
        "pageNo":1,
        "pageSize":20
    },
    "id":"bdcb1b1b-5832-451d-9fcb-352da1ffc9df"
}
```


可点击key和v

5.7 数字社区人行服务API

5.7.1 人行轨迹检索

/home/face/user/trace/query	1.0.0	人行轨迹检索

请求示例：

```
/**
 * 人行轨迹检索
 */
public class UserTraceQueryTest {
    private final static String appKey = "30239797";
     private final static String appSecret = "*****************";
    private final static String iotGatewayHost = "api.link.aliyun.com";
    public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
            new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey(appKey);
        ioTApiClientBuilderParams.setAppSecret(appSecret);
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("1.0.0");
        long now = System.currentTimeMillis();
        long period = (1000*60*60*24)*365;
        long start = (now-period);
        long end = (now-period)+24;
        request.putParam("startTM", start);
        request.putParam("endTM", end);
        request.putParam("faceId", "50800A187b22c53ca3ca80a987e6c3b58c5b8152");
        ApiResponse response = syncClient.postBody(iotGatewayHost,
            "/home/face/user/trace/query", request, true);
        System.out.println("response code = " + response.getCode() + " response = " + new String(response.getBody(), "UTF-8"));
    }
}
```

返回示例：


```
{
    "code":200,
    "data":{
        "timeSeries":"
{"1592996168447":"1","1592996169886":"1","1592996171420":"1","1592996172904":"1","1592}",
        "faceId":"80097c30be234205a5ecc0cd44513fe3",
        "positionMap":"{"1":{"iotId":"6fxzRsEBH4AGUizTZ0bV000000"}}"
    },
    "id":"5c9b7196-55ba-4033-9356-fca493add255"
}
```


5.7.2 人行数据订阅

实时订阅人行数据，用于业务层做流程组装。

CustomerFlowBasicFaceData	1.1

请求示例：

```
/**
 * 人行数据订阅
 */
public class CustomerFlowBasicFaceTopic {
    private final static String appKey = "30332082";
    private final static String appSecret = "****************";
    private final static String endPoint = "https://%s.iot-as-http2.cn-shanghai.aliyuncs.com:443";
    public static void main(String[] args) {
        String endpoint = String.format(endPoint, appKey);
        // 连接配置
        Profile profile = Profile.getAppKeyProfile(endpoint, appKey, appSecret);
        // 构造客户端
        MessageClient client = MessageClientFactory.messageClient(profile);
        // 数据接收
        client.connect(messageToken -> {
            System.out.println(messageToken.getMessage());
            JSONObject jsonObject = JSON.parseObject(new String(messageToken.getMessage().getPayload()));
            return MessageCallback.Action.CommitSuccess;
        });
        try {
            Thread.sleep(1000000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        MessageCallback messageCallback = new MessageCallback() {
            @Override
            public Action consume(MessageToken messageToken) {
                Message m = messageToken.getMessage();
                System.out.println("++++++++"+m);
                return Action.CommitSuccess;
            }
        };
        String topic = String.format("/sys/%s/dop/model/data/change", appKey);
        client.setMessageListener(topic, messageCallback);
    }
}
```

返回示例：

2.5.4. 小程序不是最新版本的处置方式

每次数字社区平台对外发布新版本后，居民侧的小程序会按需进行新版本发布，支付宝小程序自身有版本刷新机制，正常情况下居民使用正常的情况下，支付宝都将自动默认升级为最新版，如果居民支付宝中的小程序未升级到最新版本，您可以参照以下说明进行操作，能够保证小程序刷新。

1.1 支付宝主页中存在“我的小程序”



image.png

1.2 支付宝主页中不存在小程序





image.png

2. 长按未升级到最新版本的小程序进行删除



image.png

3. 重新通过扫码或者搜索进入小程序

您将看到最新的小程序功能。

2.5.5. 房屋认证被驳回后的操作

由于业主填写信息后台物业可以在信息不正确时进行拒绝操作，拒绝后居民可以通过以下链路对错误的人员信息进行修改，请勿让业主重新认证房屋，这样会引起“用户已经存在”的错误。

1. 进入“我的房屋”
2. 点击顶部进入“房屋管理”
3. 点击“认证失败”的房屋
4. 选择“重新认证”进入信息填写页

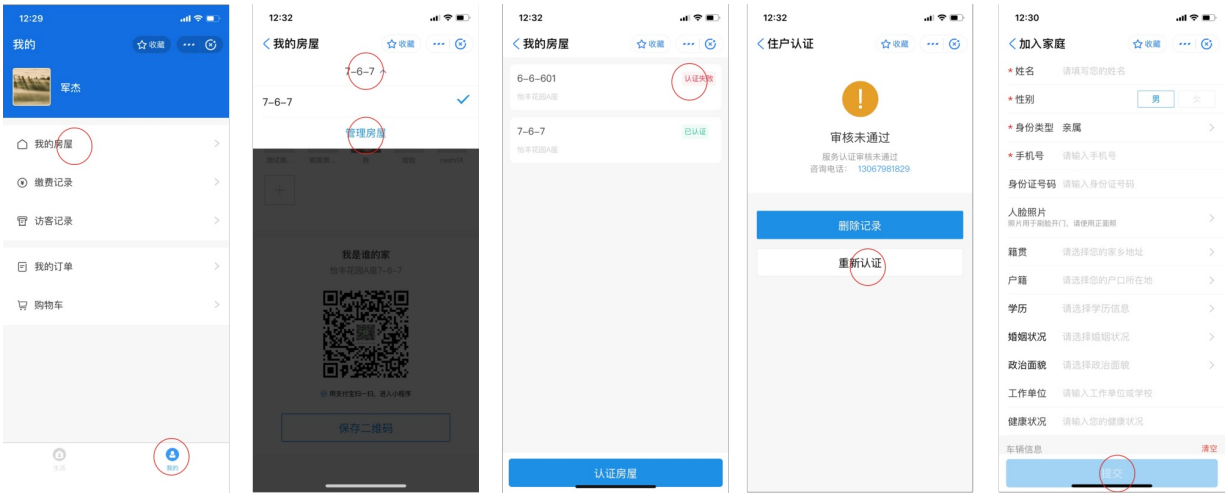


image.png

2.5.6. 人员扩展&车位储藏室录入说明

数字社区面向政府诉求进行了优化，补充了人员籍贯、户籍所在地、学历等更多信息，本章将为您介绍如何设置这些信息

1. 人员信息

1.1 首次认证房屋的居民

认证房屋过程中，您可以点击“更多信息”来录入人员的扩展信息


半夜12:50
认证房屋
收藏
房屋信息
* 房号
请选择您要认证的房屋
+ 继续添加
身份信息
* 姓名
请填写您的姓名
* 性别
男
女
* 身份类型
业主
* 手机号
请输入手机号
身份证号码
请输入身份证号码
人脸照片
照片用于刷脸开门，请使用正面照
更多信息
车辆信息
清空
车牌号码
请输入车牌号码
品牌车型
请输入品牌车型
车辆颜色
请选择车辆颜色
提交

image.png



image.png

1.2 已经成为居民的用户

进入“我的”窗口页面中进入我的房屋中对个人信息进行补充或修改

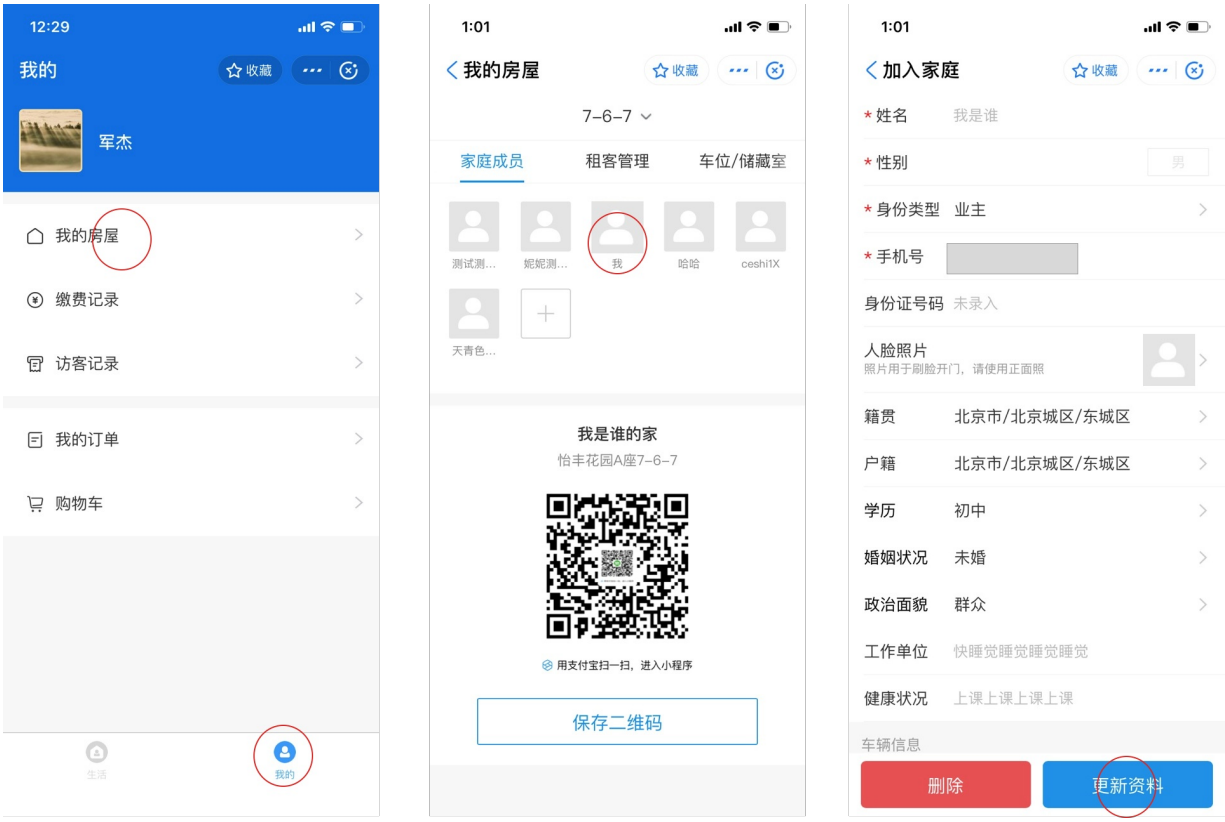


image.png

2.编辑车位储藏室

只允许业主对 车位储藏室的信息进行编辑，进入“我的”窗口页面中进入我的房屋中切换TAB至“车位/储藏室”

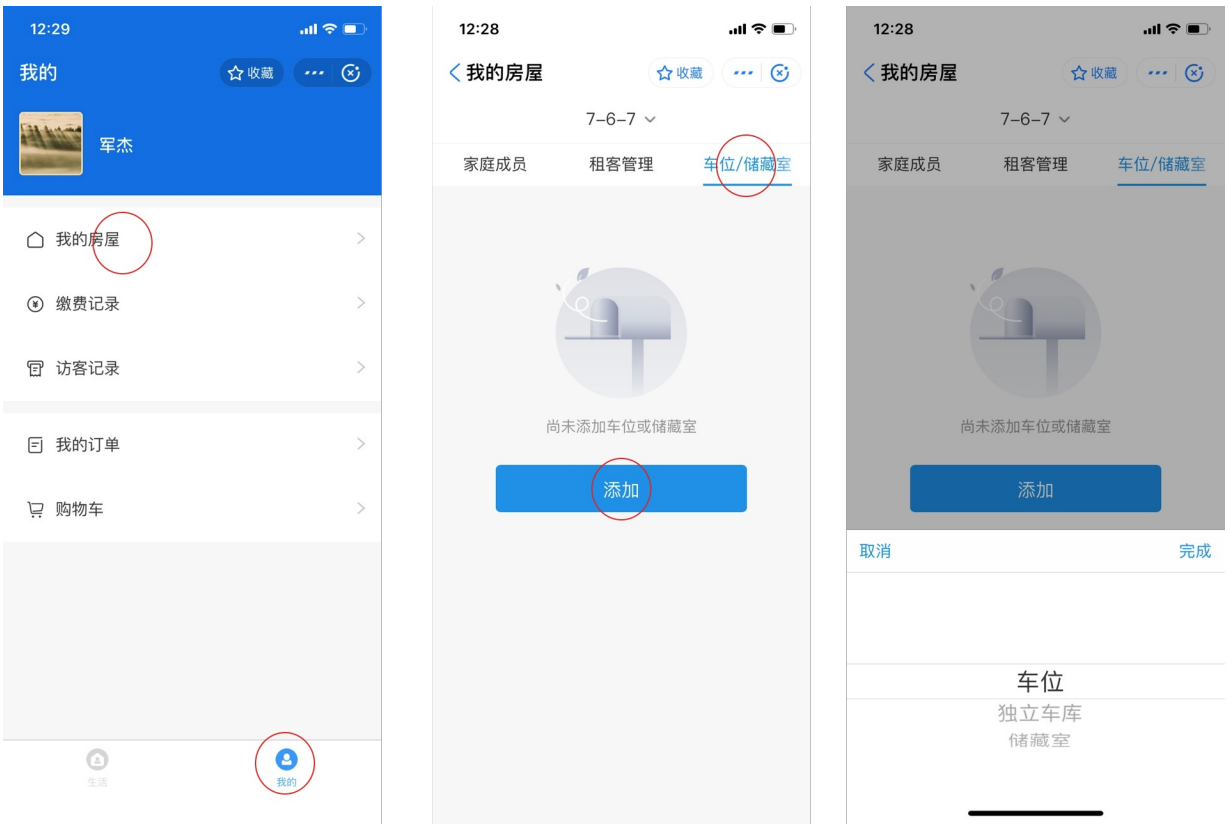
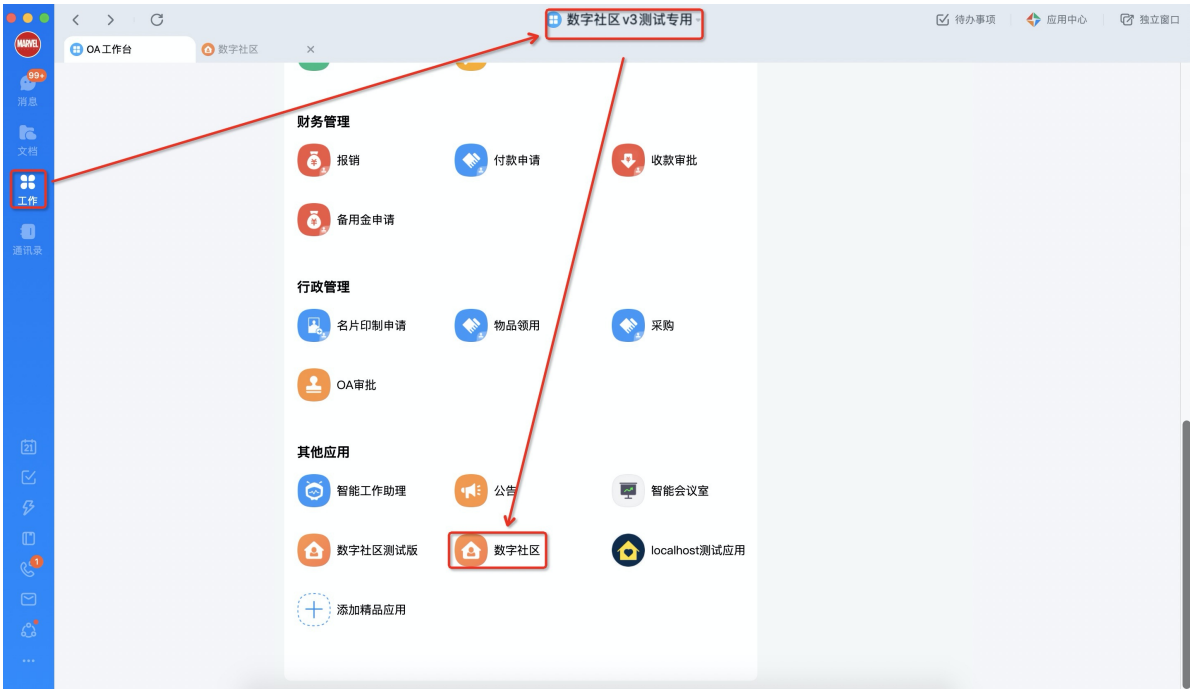


image.png

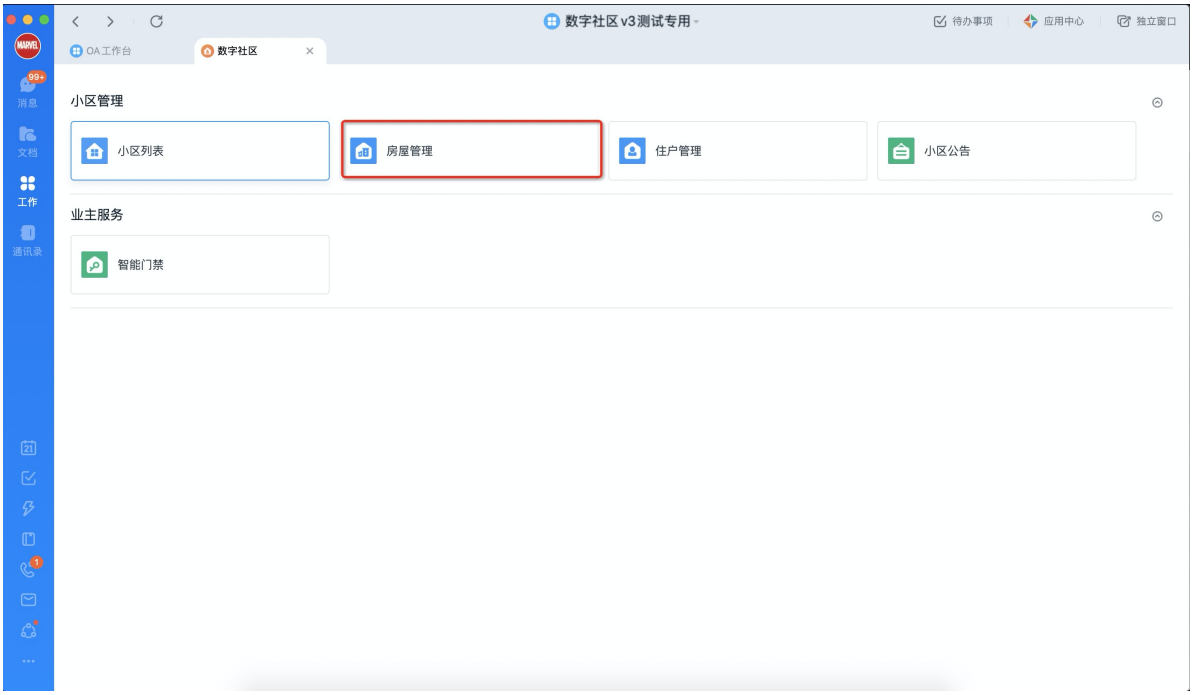
2.5.7. 小区快速迁移指引

本章将为您介绍如何将原有小区快速迁移至数字社区

1. 前提准备
    i. 第一步：钉钉账户注册及开通数字社区
    ii. 第二步：创建小区，设置小区管理员
2. 通过钉钉PC端进入数字社区服务



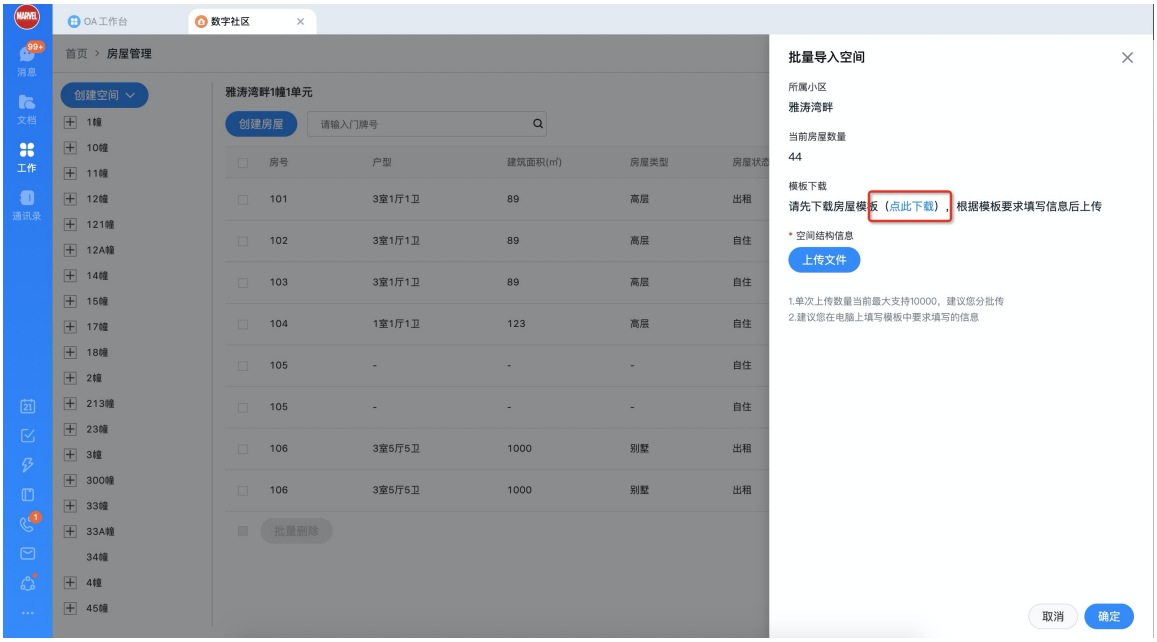
3. 进入小区房屋进行房屋结构的导入



i. 选择批量导入空间，注意右上角的小区名称，请选择您需要导入的小区进行操作



ii. 下载批量导入文件模板



iii. 模板文件说明

*楼栋（必填）	*单元（必填）	*房号（必填）	房屋户型	建筑面积	房屋类型	房屋状态	
1	1	101	3/1/1	89	1:高层	1:自住	输入示例（上传时请删除示例）

a. 楼栋，必填项，填写楼栋编号，支持大写英文和数字，最长10个字符，如果您的楼栋总数达到2位数，建议您将1幢录入为01

b. 单元，必填项，填写单元编号，支持大写英文和数字，最长10个字符，如果您的单元总数达到2位数，建议您将1单元录入为01

c. 房号，必填项，填写房屋编号，支持大写英文和数字，最长10个字符，如果您的房号总数达到4位数，建议您将101室录入为0101

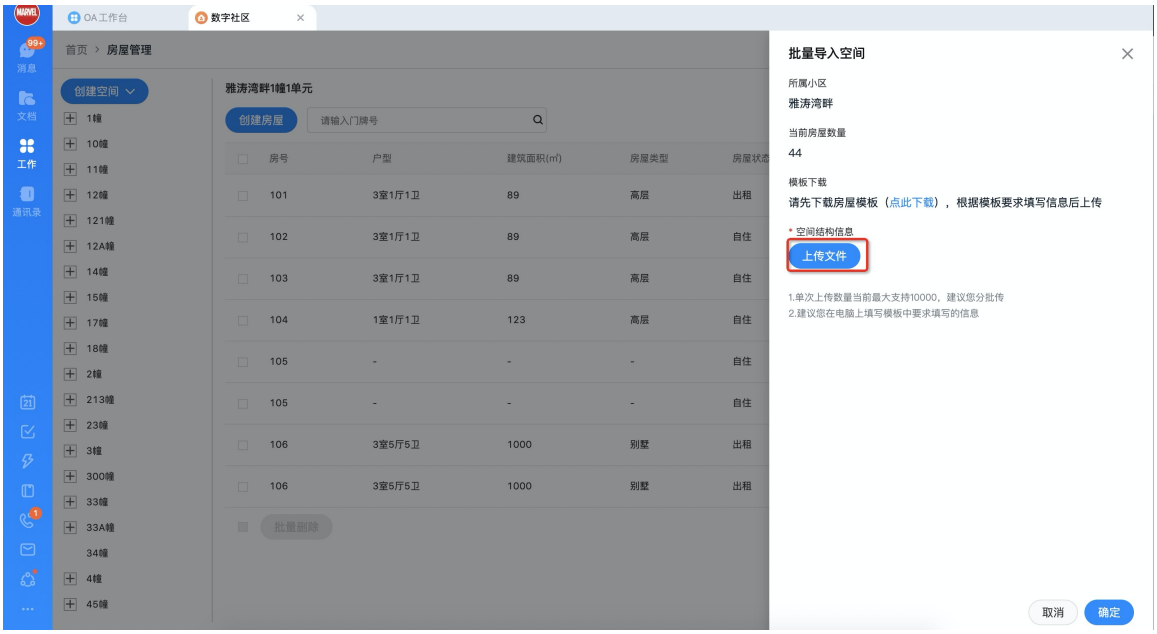
d. 房屋户型，非必填项，按照n/n/n的填写格式，表示几室几厅几卫

e. 建筑面积，非必填项

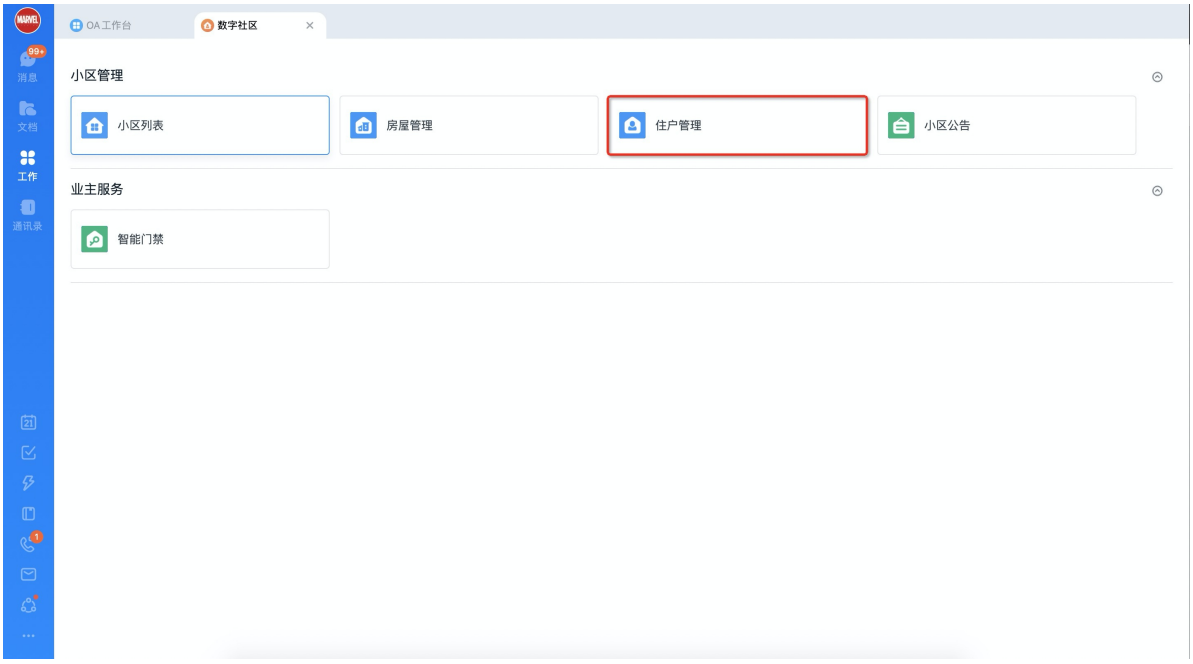
f. 房屋类型，非必填项，填写1:高层2:多层3:别墅4:联排5:叠排6:公寓7:其他

g. 房屋状态，非必填项，填写1:自住2:出租3:空置

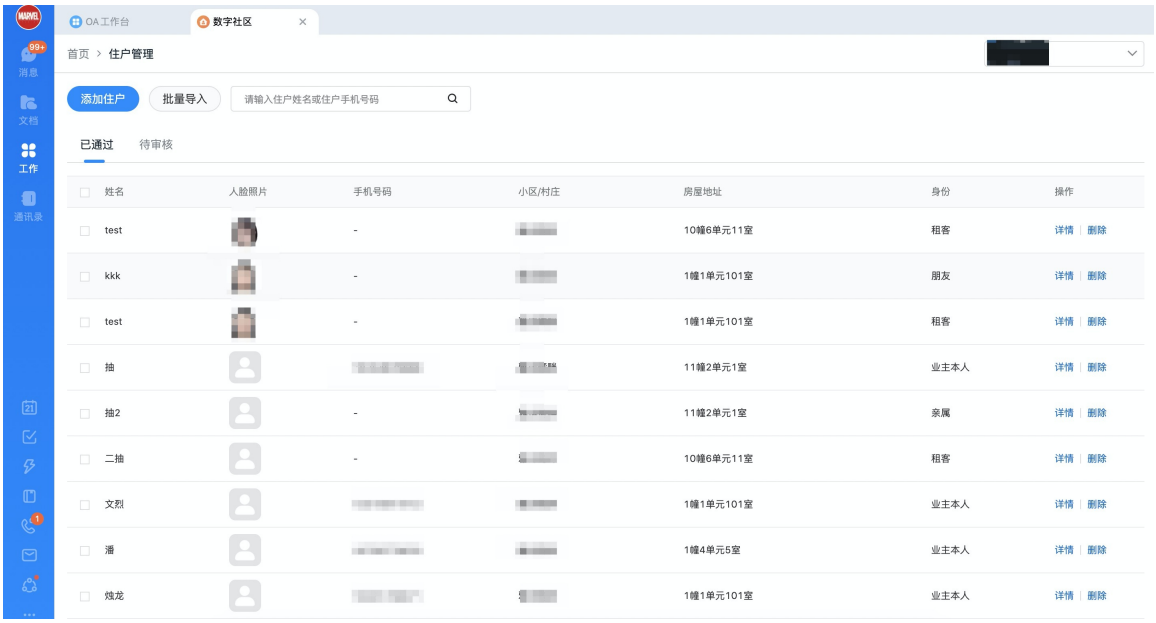
iv. 上传文件，将您编辑好的excel进行导入



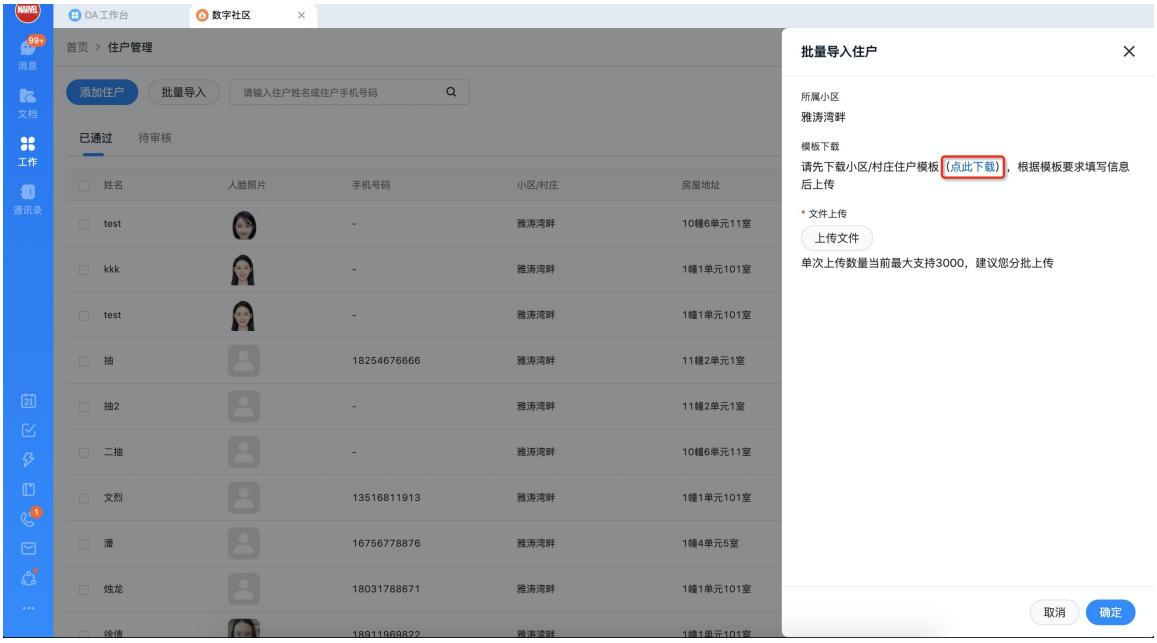
4. 进入小区住户管理进行人员信息的导入



i. 选择批量导入，注意右上角的小区名称，请选择您需要导入的小区进行操作



ii. 下载批量导入的模板



iii. 模板文件说明

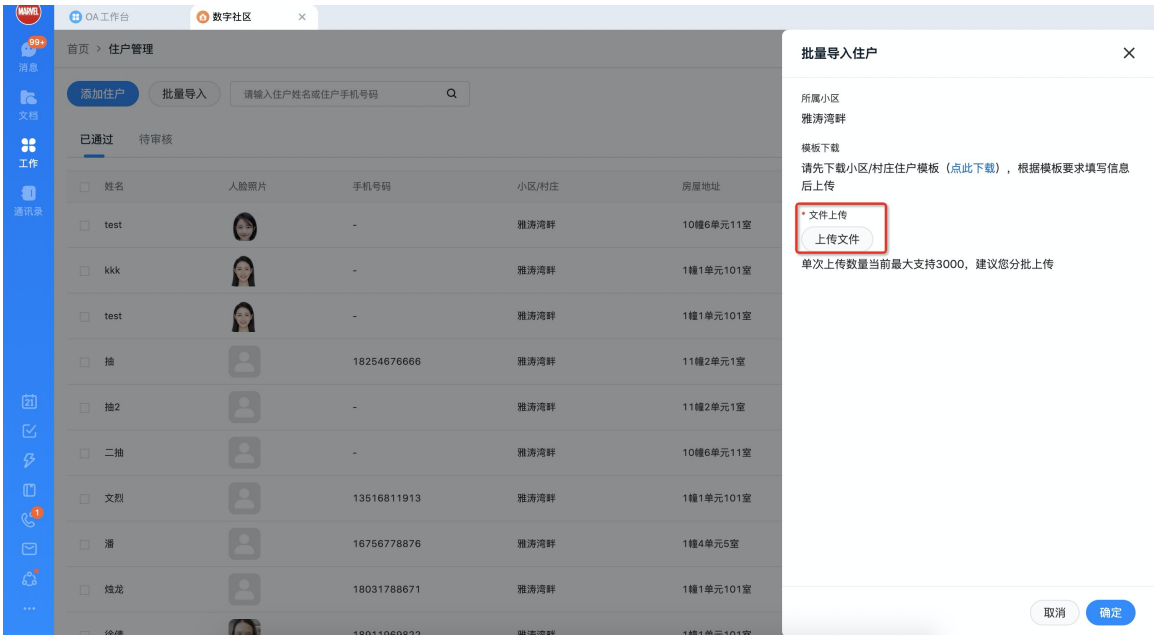
a. 模板文件中红色字体标注的列为比填列，其他项都为选填项

*姓名（必填）	*手机号（必填）

b. 房屋信息部分字段可以不填写， 由于您已经在房屋管理中导入这部分数据，所以请忽略

房屋户型	建筑面积	房屋类型	房屋状态
3/1/1	89	1:高层	1:自住

iv. 上传说明将您编辑好的模板进行导入



2.5.8. 数字社区体验说明

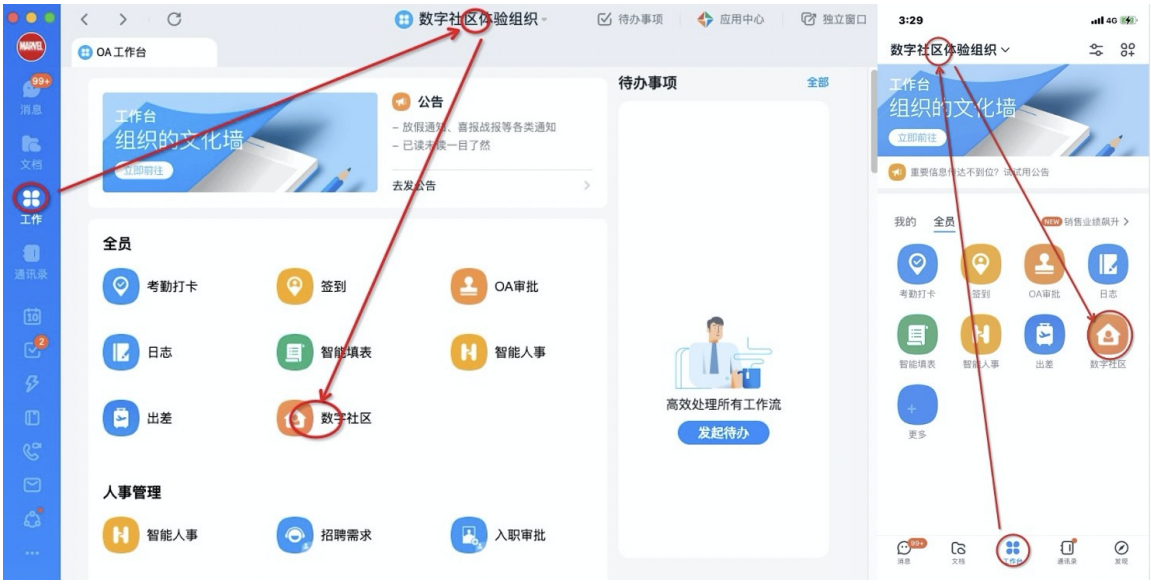
本章将为您介绍如何体验数字社区中的各类服务，由于正式小区的开通需要提供相关法务合同证明企业和小区间的管理关系，为了简化体验流程，数字社区已经申请数字社区钉钉组织，并创建了2个支付宝小程序，供体验用户使用。

1. 加入体验组织
   i. 使用钉钉扫码下方二维码，申请加入**数字社区体验组织**



   ii. 同学审批通过，并将您设置为小区管理员后，您即可体验数字社区的相关服务
2. 涉及端介绍
   i. 管控后台（登录地址，点我），只允许组织的管理员和子管理员登录，如需体验请在审批权限时告知审批同学，该入口可以配置物业/居民小程序部分功能，更多功能介绍请进入文档->功能介绍->管控后台目录中进行查看详细功能

ii. 物管服务，提供钉钉移动端、PC两个端入口，请进入钉钉工作->切换组织->数字社区，更多功能介绍请进入文档->功能介绍->物管服务目录中进行查看详细功能



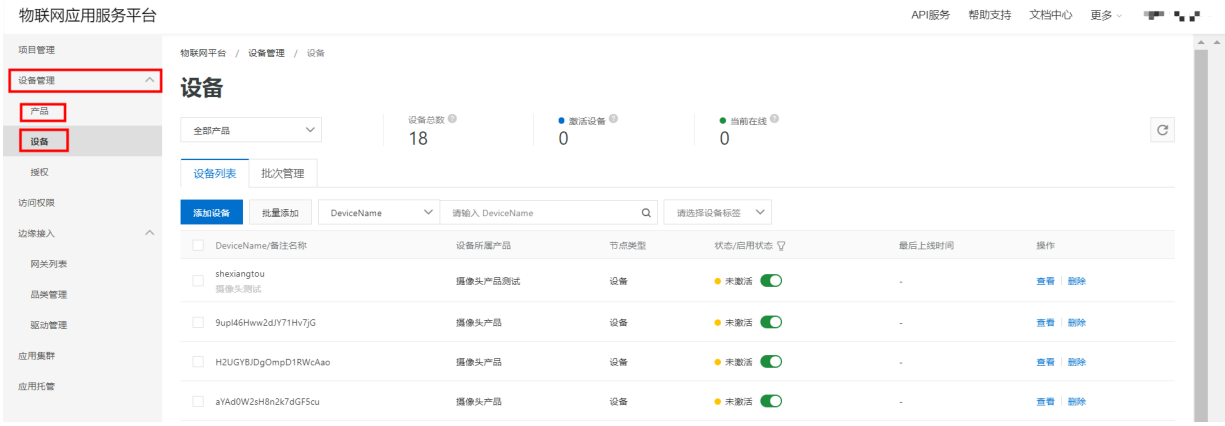
iii. 住户服务，提供支付宝小程序入口，请使用支付宝小程序扫描体验小区二维码，更多功能介绍请进入文档->功能介绍->住户服务目录中进行查看详细功能



2.5.9. 数字社区小区及设备集成指南

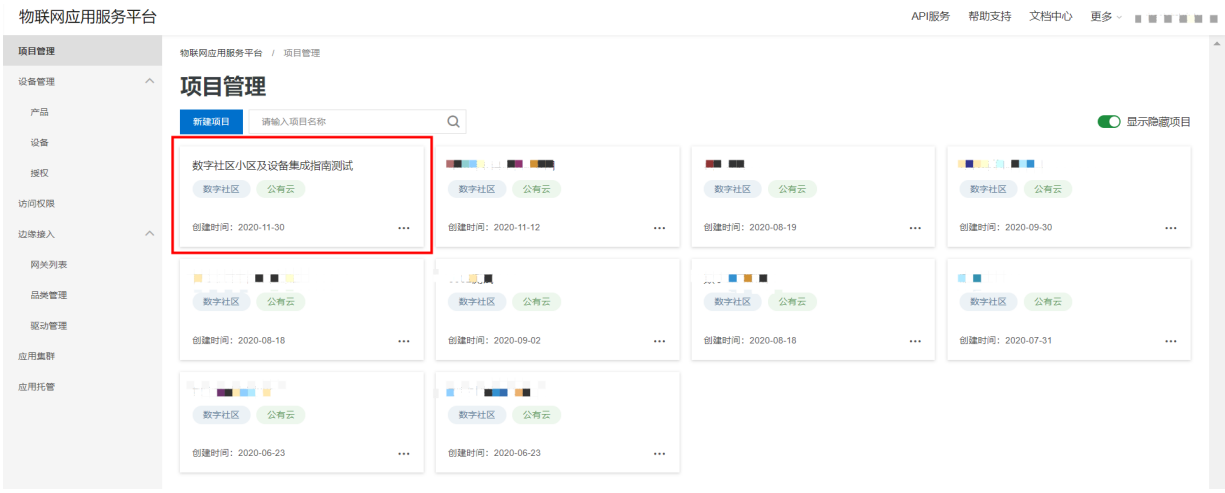
一、通用说明

1. 登入“物联网应用服务平台”后，左侧导航中的“设备管理-产品”与“设备管理-设备”为该账号下的所有的产品与设备。在创建小区后，可为小区绑定设备，一个设备只能绑定一个小区。整体的流程为：项目绑定设备、设备添加到小区。



二、创建小区

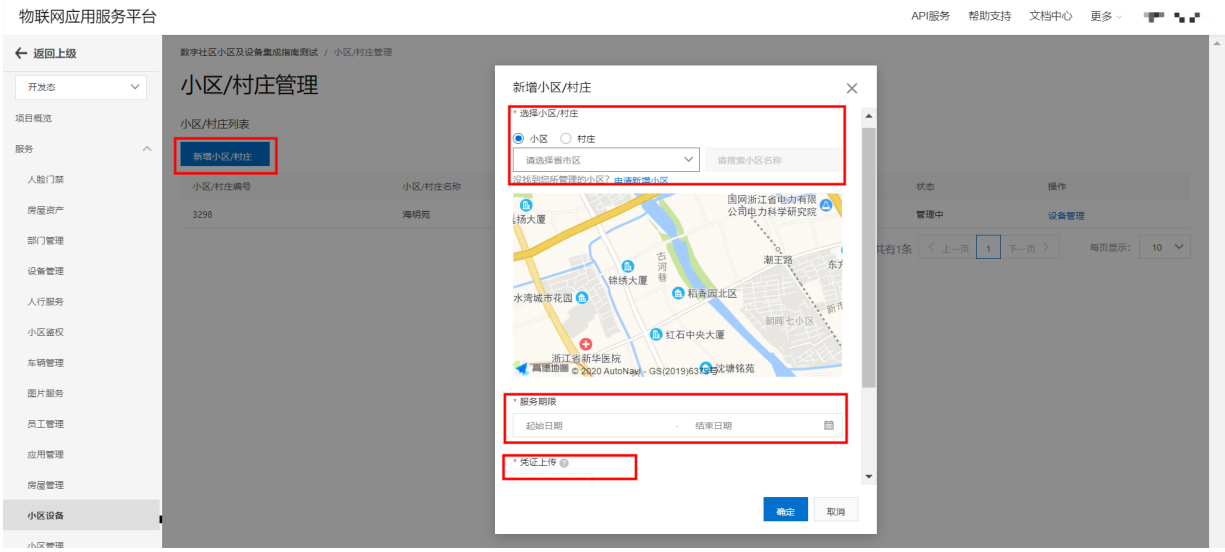
1. 进入创建好的项目。



2. 选择“开发态”，点击“服务-小区设备”进入“小区/村庄管理”。



3. 点击“新增小区/村庄”，填写小区相关信息，为该项目创建小区。创建后该小区会进入审核状态，请联系相关阿里同学审核。



4. 创建成功之后就可以对该小区进行设备管理操作。



三、小区设备管理

1. 切换到“集成态”先为该项目绑定设备。



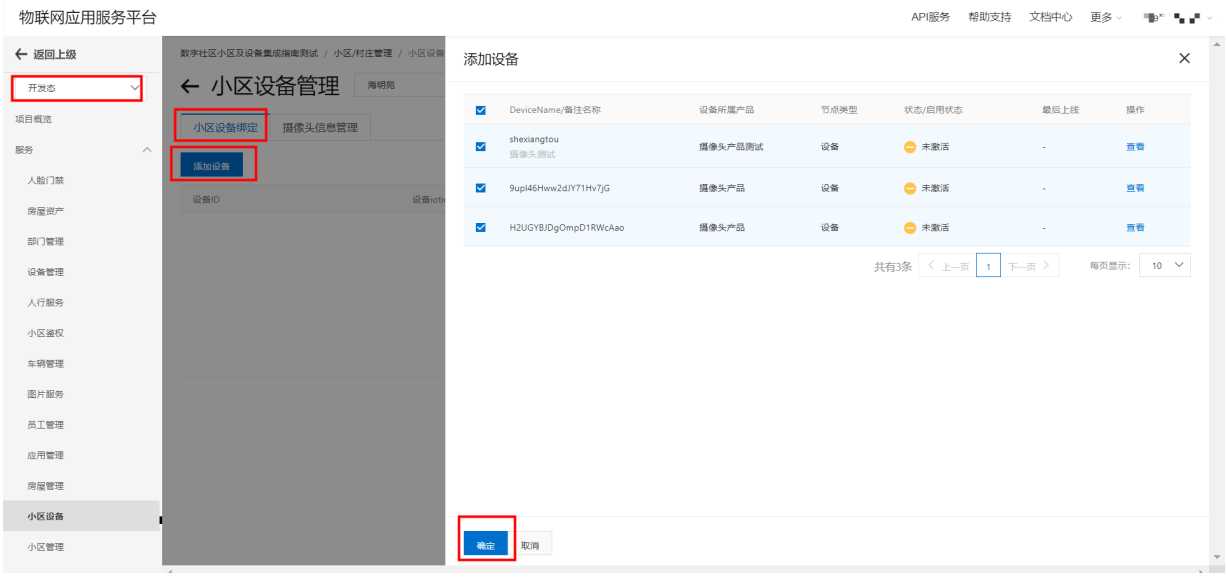
2. 选择需要绑定的设备进行批量绑定。



3. 切换到“开发态”选择需要管理设备的小区，点击“设备管理”进入该小区的设备管理界面。



4. 选择“小区设备绑定”点击“添加设备”选择对应的设备为该小区绑定设备。



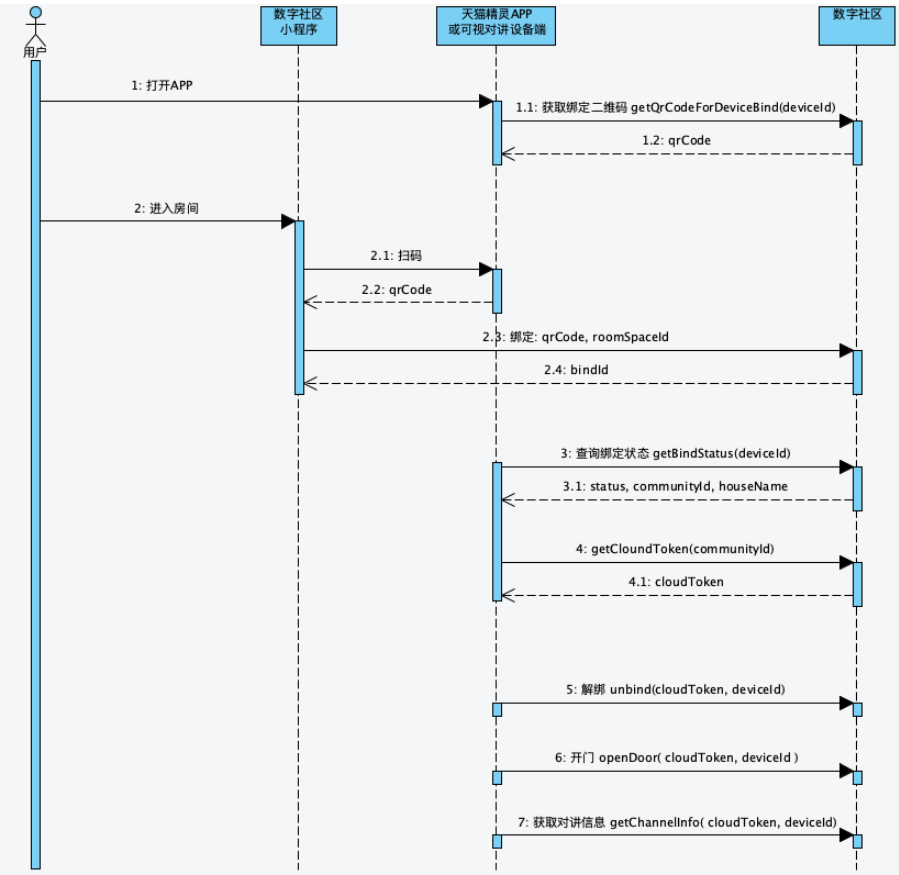
5. 选择“摄像头信息管理”可对摄像头设备进行批量导入、清空、编辑的操作。



2.5.10. 天猫精灵社区服务APK绑定服务

本章主要介绍如何通过用户自己开发的用户端将天猫精灵中安装的社区服务绑定至房屋中，以便可视对讲设备呼叫到天猫精灵设备

一、流程



二、API

1. 精灵端（或其他可视对讲设备端）

1.1 获取绑定二维码

path	版本	是否需要登录
/solution/community/intercom/device/qrcode/get	1.0.0	否

入参

入参名称	数据类型	是否必须	入参描述
deviceId	String	是	对讲设备标识，如天猫精灵设备ID
deviceType	String	否	对讲设备类型，默认TMALL_GENIE
ts	Long	是	请求时间戳（服务端校验±1小时）
sign	String	是	签名 str = {appKey}-{ts}-{deviceId}{deviceType} sign = md5(str)

出参

字段	类型	备注
code	Integer	返回码

message	String	返回信息
data	JSONObject	
-qrCode	String	绑定ID（可生成二维码透出给小程序端扫描）
-expireTime	String	绑定失效时间yyyy-MM-dd HH:mm:ss
-qrCodeUrl	String	二维码图片地址（将来可能要提供不同分辨率）

1.2 查询设备绑定状态

path	版本	是否需要登录
/solution/community/intercom/device/status/get	1.0.1	否

入参

入参名称	数据类型	是否必须	入参描述
deviceId	String	是	对讲设备标识，如天猫精灵设备ID
deviceType	String	否	对讲设备类型，默认TMALL_GENIE

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	JSONObject	
-bindId	String	绑定ID，bindStatus为true时有值
-bindStatus	Boolean	绑定状态，true已绑定，false未绑定
-houseName	String	绑定房号，bindStatus为true时有值
-communityId	Long	小区ID
-bindType	Integer	0 业主 1 物业

1.3 设备端解绑室内对讲设备

path	版本	是否需要登录
/solution/community/intercom/device/end/unbind	1.0.0	否

入参

入参名称	数据类型	是否必须	入参描述
deviceId	String	是	对讲设备标识，如天猫精灵设备ID

deviceType	String	否	对讲设备类型，默认TMALL_GENIE

出参

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

1.4 设备端开门

path	版本	是否需要登录
/solution/community/intercom/unitdoor/open	1.0.1	否

入参

入参名称	数据类型	是否必须	入参描述
deviceId	String	是	对讲设备标识，如天猫精灵设备ID
deviceType	String	否	对讲设备类型，默认TMALL_GENIE
iotId	String	否	目标门禁设备的设备ID

出参

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

1.5 查询对讲会话信息

path	版本	是否需要登录
/solution/community/intercom/device/channel/get	1.0.0	否

入参

入参名称	数据类型	是否必须	入参描述
deviceId	String	是	对讲设备标识，如天猫精灵设备ID
deviceType	String	否	对讲设备类型，默认TMALL_GENIE

出参

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

data	JSONObject	
-communityId	Long	小区
-houseName	String	房号
-unitSpaceId	String	单元空间ID
-houseSpaceId	String	房屋空间ID
-sipGroupId	String	SIP服务组名（对应房屋）
-sipNumber	String	SIP服务ID（对应语音设备）
-sipPassword	String	SIP服务密码
-sipHost	String	SIP服务地址
-sipPort	String	SIP服务端口

1.6 查询可用的门禁设备列表

path	版本	是否需要登录
/solution/community/intercom/entrance/device/list	1.0.0	否

入参

入参名称	数据类型	是否必须	入参描述
deviceId	String	是	对讲设备标识，如天猫精灵设备ID
deviceType	String	否	对讲设备类型，默认TMALL_GENIE
pageSize	Integer	是	页面大小
pageSize	Integer	是	页号

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	JSONObject	
-pageSize	Integer	页面大小
-pageSize	Integer	页号
-total	Integer	门禁设备总数

-data	JSONArray	房屋空间ID
--communityId	Long	小区ID
--iotId	String	设备ID
--deviceName	String	设备名称
--nickName	String	设备昵称
--productKey	String	设备PK
--productName	String	产品名称
--thingType	String	设备类型。可以是VIRTUAL、WEB、APP、DEVICE
--status	Integer	设备状态。0-初始化；1-在线；3-离线；8-禁用
--nodeType	Integer	产品类型，0 DEVICE 1 GATEWAY
--netType	String	产品入网方式
--categoryId	Long	品类ID
--categoryName	String	品类名称

2. 小程序端

关于paasIdentityId的使用方法，下面二选一

- 一方SaaS应用时，可以不填写，使用登录态获取用户信息
- 三方PaaS应用时，必填代表用户的identityId

2.1 绑定室内对讲设备

path	版本	是否需要登录
/solution/community/intercom/device/bind	1.0.0	可选 需要cloudToken

入参 ChannelAuthDTO

入参名称	数据类型	是否必须	入参描述
paasIdentityId	String	否	paas调用时的用户ID
qrCode	String	是	绑定二维码，见1.1
houseSpaceId	String	是	房屋的空间ID
deviceNickName	String	是	设备别名

出参

字段	类型	备注
code	Integer	返回码

message	String	返回信息
data	JSONObject	
-bindId	String	绑定ID

2.2 更新室内对讲设备别名

path	版本	是否需要登录
/solution/community/intercom/device/rename	1.0.0	可选 需要cloudToken

入参 ChannelAuthDTO

入参名称	数据类型	是否必须	入参描述
paasIdentityId	String	否	paas调用时的用户ID
bindId	String	是	绑定ID
deviceNickName	String	是	设备别名，长度最大64，中文、字母、数字、横线、下划线、点组成

出参

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

2.3 全量查询室内对讲设备列表

path	版本	是否需要登录
/solution/community/intercom/device/list	1.0.0	可选 需要cloudToken

入参 ChannelAuthDTO

入参名称	数据类型	是否必须	入参描述
paasIdentityId	String	否	paas调用时的用户ID

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	JSONObject	
-bindId	String	绑定ID

-intercomDeviceId	String	脱敏的对讲设备ID
-intercomDeviceType	String	对讲设备类型
-houseSpaceId	String	房屋的空间ID
-houseName	String	房间号

2.4 小程序端解绑室内对讲设备

path	版本	是否需要登录
/solution/community/intercom/device/unbind	1.0.0	可选 需要cloudToken

入参 ChannelAuthDTO

入参名称	数据类型	是否必须	入参描述
paasIdentityId	String	否	paas调用时的用户ID
bindId	String	是	绑定ID

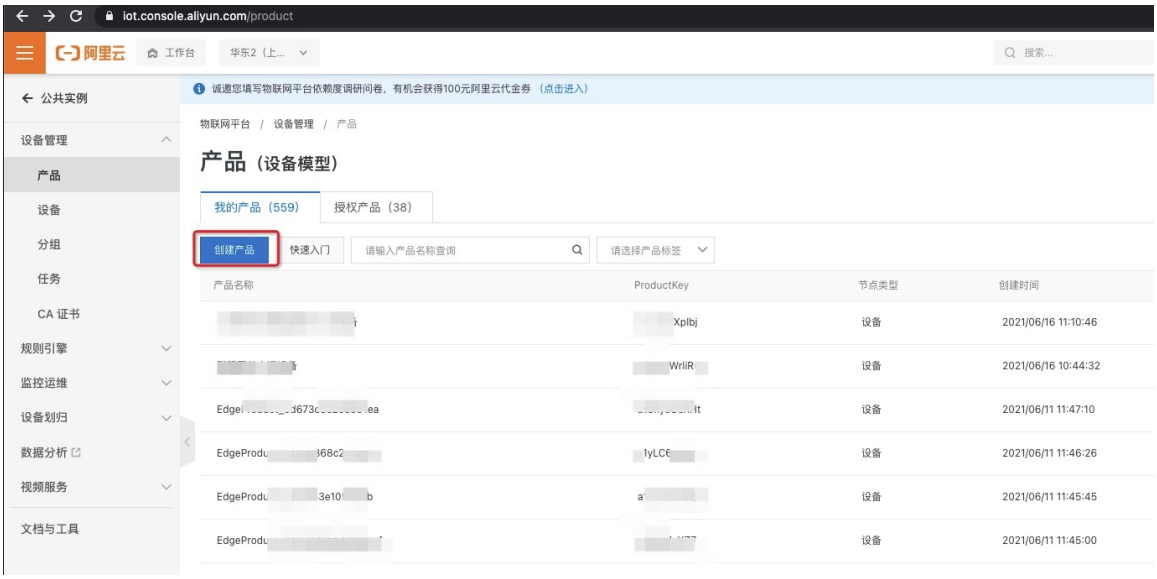
出参

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

2.5.11. 伙伴门禁机集成手册

本章将为您介绍如何通过平台云端集成门禁设备（已经接入数字社区的门禁设备）

1. 生产三元组，三元组是每一台门禁机的平台身份，只有配置了三元组的设备才能链接至平台，并将数据同步至平台中
   i. 登录物联网平台，点击创建产品，如果您还未开通，请根据页面指引开通即可



ii. 输入产品相关信息（参照以下示意内容做输入），在所属品类中选在“多功能门禁”，点击确定

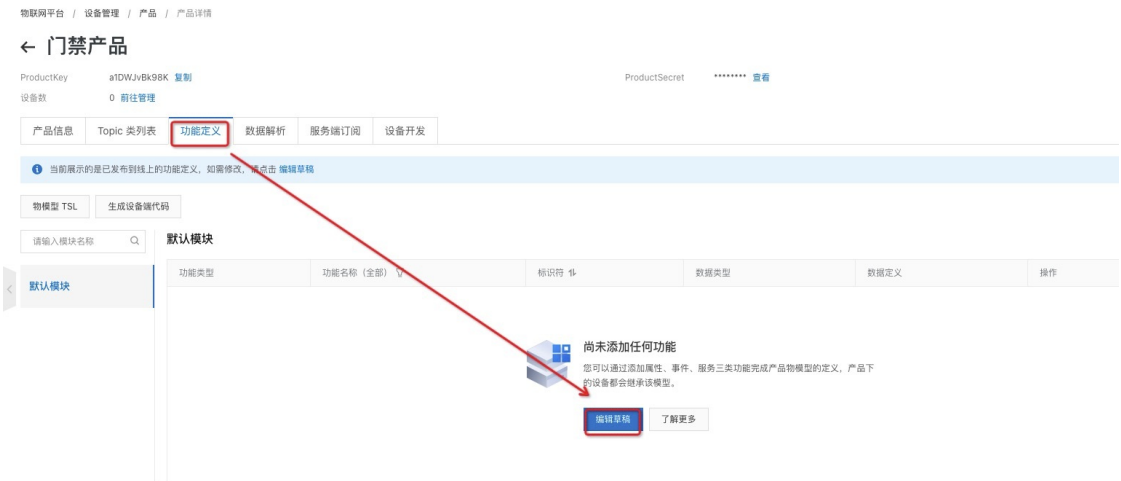


iii. 创建产品成功后进入详情进行产品属性编辑

a. 进入产品详情页



b. 选择“功能定义”并点击“编辑草稿”

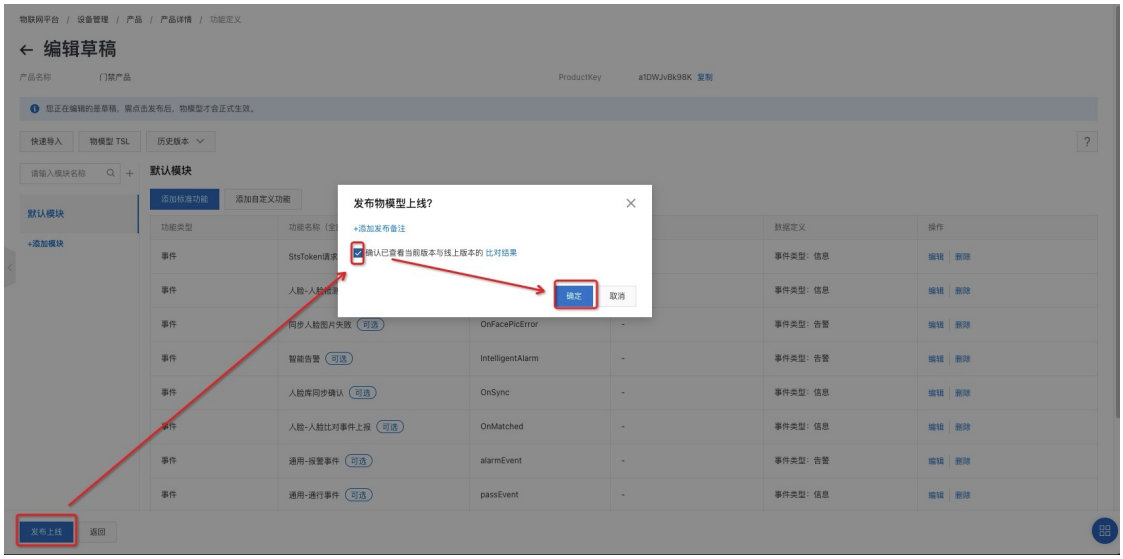


c. 注意

进入编辑标准功能，并勾选除“SyncFaceDat a”以外的全部“多功能门禁”下的所有属性

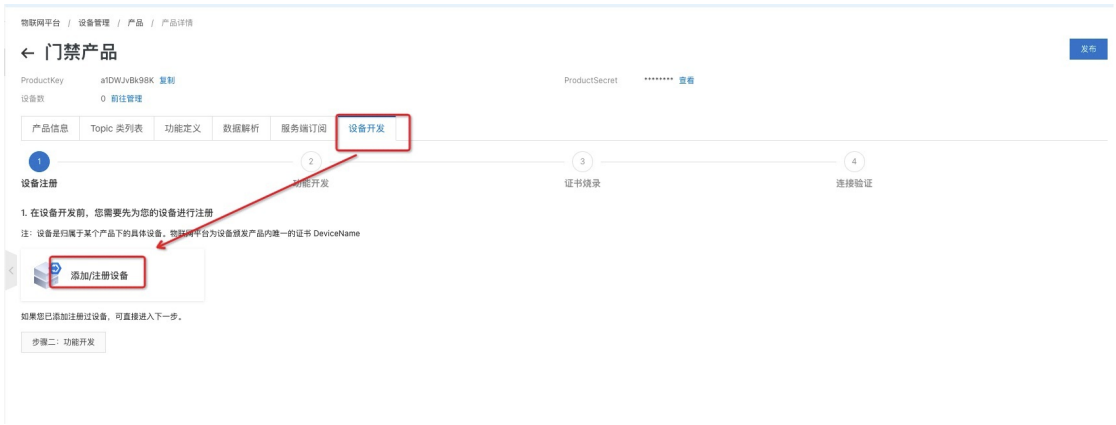


d. 属性选择完成后，点击发布上线



iv. 编辑属性成功后创建设备

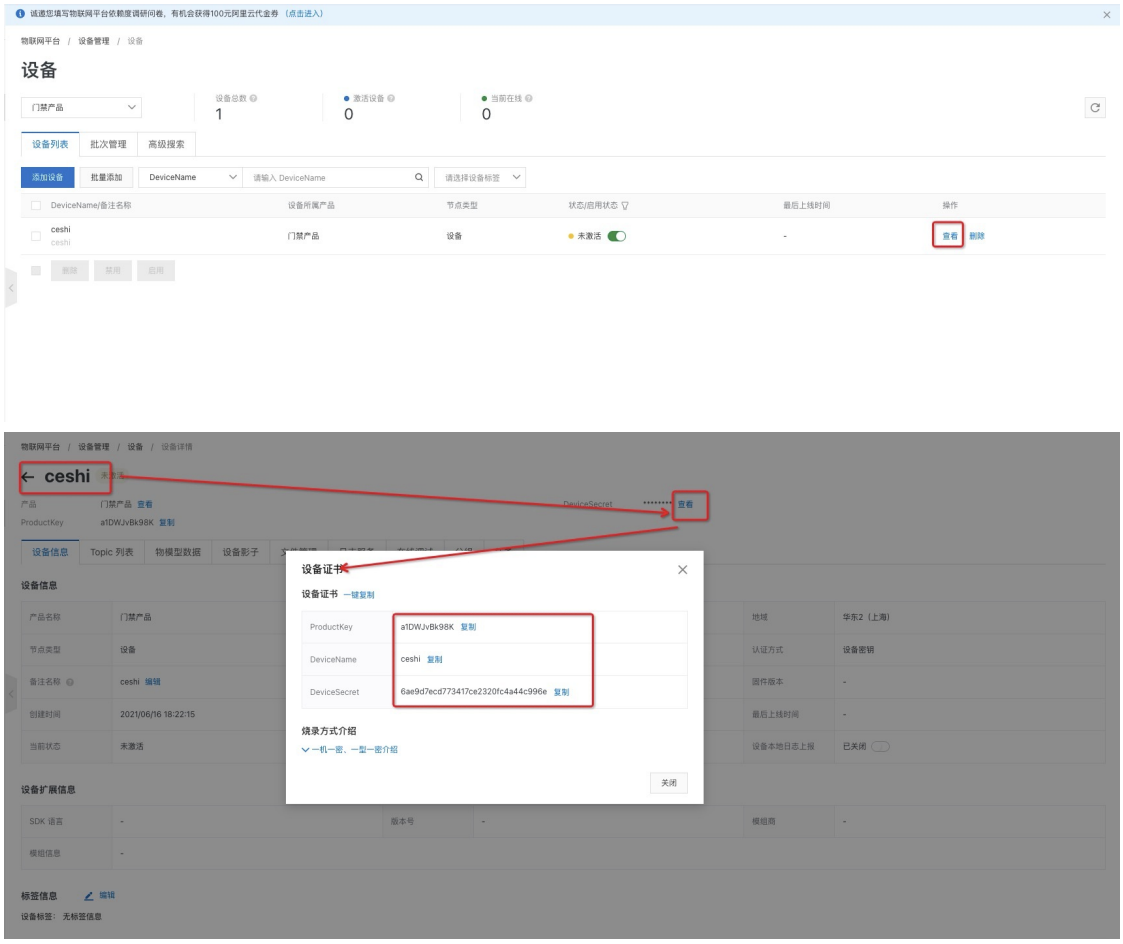
a. 进入设备开发，并点击“添加设备”



b. 输入设备的DeviceName和备注名

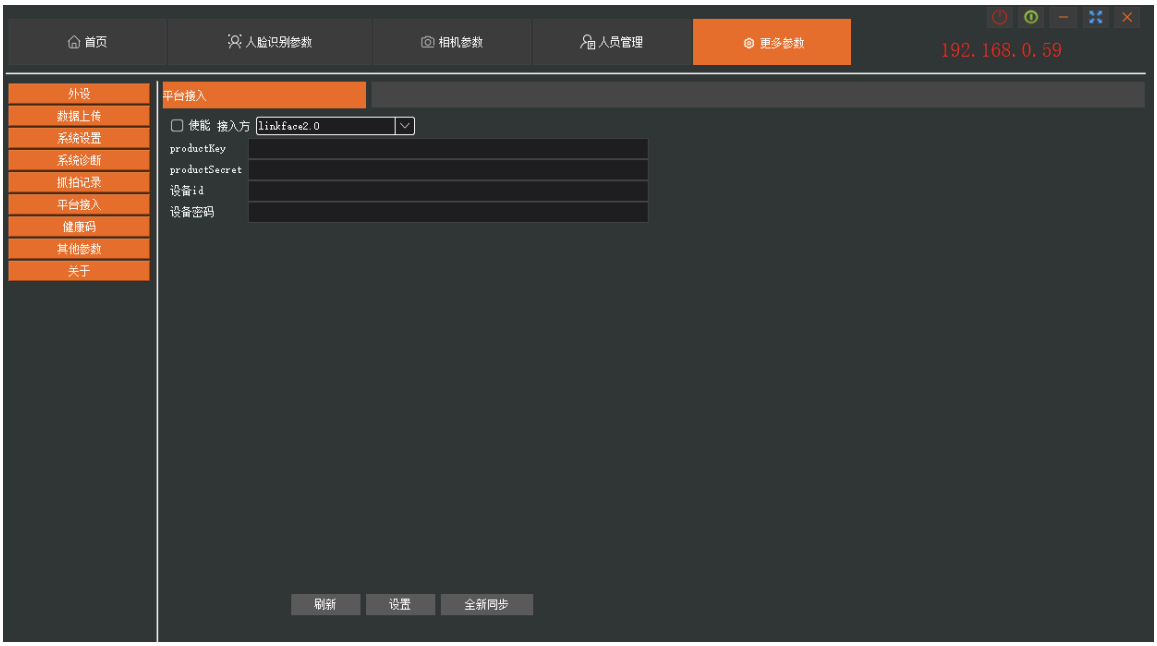


a. 设备创建成功后，可以在设备列表中查看设备三元组信息

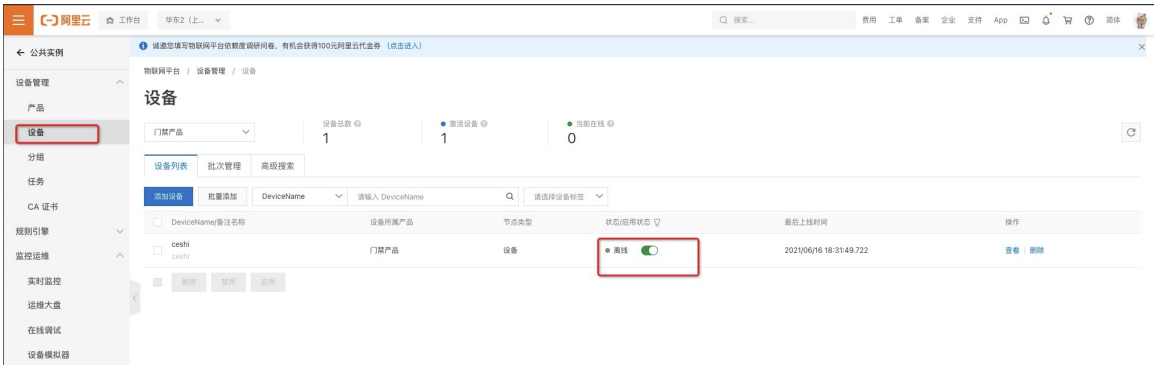


2. 将三元组配置至门禁设备中

i. 每个对接的门禁设备都会有本地的配置页面，可咨询对应供应商获取使用手册



ii. 填写完成后建议重启下设备，重启后可在平台中查看设备是否在线，从而确认配置的正确性



3. 使用平台接口对门禁设备进行物业场景的组装，由于API接口授权需要进入应用服务平台操作，请参照接入准备文档进行平台入驻和AK申请
4. 门禁相关详细能力接口，请访问门禁管理服务文档

2.6. 相关资料

2.6.1. 入驻协议

阿里云数字社区平台服务协议

【首部及导言】

欢迎您使用阿里云数字社区平台的服务！

为使用数字社区平台服务，您应当阅读并遵守《阿里云数字社区平台服务协议》（以下简称“本协议”）和《阿里云隐私政策》。请您务必审慎阅读、充分理解各条款内容，特别是免除或者限制阿里云计算有限公司责任的条款、对用户权利进行限制的条款、约定争议解决方式和司法管辖的条款等，以及开通或使用某项服务的单独协议或规则。限制、免责条款或者其他涉及您重大权益的条款可能以加粗、加下划线等形式提示您重点注意。

除非您已充分阅读、完全理解并接受本协议所有条款，否则您无权使用阿里云数字社区平台服务。您点击“同意”或“下一步”，或您使用阿里云数字社区平台服务，或者以其他任何明示或者默示方式表示接受本协议的，均视为您已阅读并同意签署本协议。本协议即在您与阿里云计算有限公司之间产生法律效力，成为对双方均具有约束力的法律文件。

如果您因年龄、智力等因素而不具有完全民事行为能力，请在法定监护人（以下简称“监护人”）的陪同下阅读和判断是否同意本协议，并特别注意未成年人使用条款。

如果您是中国内地、中国香港、中国澳门地区以外的用户，您订立或履行本协议还需要同时遵守您所属和/或所处国家或地区的法律。

一、【协议的范围】

1.1

本协议是用户与阿里云计算有限公司（以下简称“阿里云”）之间关于用户使用阿里云数字社区平台服务所订立的协议。阿里云“用户”是指阿里云数字社区平台的服务的使用人——**物业公司或者物业公司授权的代运营公司（统称为“物业”）**，在本协议中更多地称为“您”。

1.2

阿里云的数字社区平台服务（以下简称“本服务”）是指阿里云计算有限公司基于互联网及移动互联网向用户提供的数字社区平台系统软件，**通过此软件，物业在钉钉（PC和APP端）组织内可对负责的社区实现小区住户管理、访客管理、物业缴费、小区公告在线化、社区电商等功能（具体服务以阿里云实际提供的为准），并通过物业建立的支付宝小程序完成前述功能的实现，业主可在支付宝小程序端享受物业设定的社区服务。**

1.3本协议内容同时包括《阿里云隐私政策》（链接地址https://terms.aliyun.com/legal-agreement/terms/suit_bu1_ali_cloud/suit_bu1_ali_cloud201710161525_98396.html**），且您在使用阿里云某一特定服务时，该服务可能会另有单独的协议、相关业务规则等（以下统称为“单独协议”）。上述内容一经正式发布，即为本协议不可分割的组成部分，您同样应当遵守。您对前述任何单独协议的接受，即视为您对本协议全部的接受。您对本协议的接受，即视为您对《阿里云隐私政策》的接受。**

二、【账号与密码安全】

2.1您在使用本服务时可能需要注册一个钉钉账号（阿里云有权自主决定更换搭载本系统软件的产品，如需要届时会统一进行账号迁移），请您自己合法使用自己注册的账号。

2.2阿里云特别提醒您应妥善保管您的账号和密码。当您使用完毕后，应安全退出。因您保管不善可能导致遭受被盗号或密码失窃，责任由您自行承担，阿里云不会承担任何法律责任。

2.3为增强用户体验和/或技术便利，本服务的账号可能包括数字、字母或者其组合，以及手机号码、电子信箱等多种形式。在您注册某一形式的账号时，阿里云可能附赠该账号的另一表现形式。具体的账号形式、账号体系及账号之间的关联关系等，以阿里云实际提供的为准。

2.4您在使用本服务过程中，可以为您使用的账号设置昵称、头像、签名、留言等信息。您应当保证这些信息的内容和形式符合法律法规（本协议中的“法律法规”指用户所属/所处地区、国家现行有效的法律、行政法规、司法解释、地方法规、地方规章、部门规章及其他规范性文件以及对于该等法律法规的不时修改和补充，以及相关政策规定等，下同）、公序良俗、社会公德以及阿里云数字社区平台规则，且不会侵害任何主体的合法权益。

2.5部分第三方网站或者服务可能可以用您的其他社交账号等作为其登录途径之一。您知晓，除非阿里云特别说明外，这些网站或者服务并非阿里云运营，您应自行判断此类第三方网站或者服务的安全性和可用性，并自行承担相关风险和责任。

三、【用户个人信息保护】

3.1保护用户个人信息是阿里云的一项基本原则。阿里云将按照本协议及《阿里云隐私政策》（链接地址：https://terms.aliyun.com/legal-agreement/terms/suit_bu1_ali_cloud/suit_bu1_ali_cloud201710161525_98396.html**）的规定收集、使用、储存和分享您的个人信息。本协议对个人信息保护相关内容未作明确规定的，均应以《阿里云隐私政策》的内容为准。**

3.2您在注册账号或使用本服务的过程中，可能需要填写一些必要的信息。若国家法律法规有特殊规定的，您需要填写真实的身份信息。若您填写的信息不完整，则可能无法使用本服务或在使用过程中受到限制。

3.3一般情况下，您可根据相关产品规则浏览、修改自己提交的信息，但出于安全性和身份识别（如号码申诉服务等）的考虑，您可能无法修改注册时提供的初始注册信息及其他验证信息。

3.4阿里云将尽可能运用各种安全技术和程序建立完善的管理制度来保护您的个人信息，以免遭受未经授权的访问、使用或披露。

3.5阿里云不会将您的个人信息转移或披露给任何第三方，除非：

（1）相关法律法规或司法机关、行政机关要求。

（2）为完成合并、分立、收购或资产转让而转移。

（3）为提供您要求的服务所必需。

（4）依据《阿里云隐私政策》或其他相关协议规则可以转移或披露给任何第三方的情形。

3.6阿里云非常重视对未成年人个人信息的保护。若您不具备完全民事行为能力，在使用阿里云的服务前，应事先取得您的监护人的同意。

四、【使用本服务的方式】

4.1本服务为您自身非商业及商业性质的使用。

4.2您依本协议条款所取得的权利不可转让。

4.3您不得使用任何方式（包括但不限于第三方软件、插件、外挂、系统、设备等）对本服务进行干扰、破坏、修改或施加其他影响。

4.4您应当通过阿里云提供或认可的方式使用本服务，不得通过任何第三方软件、插件、外挂、系统、设备等登录或使用本服务。

4.5任何人未经阿里云授权不得使用任何第三方软件、插件、外挂、系统等查看、获取本服务中所包含的阿里云、阿里云合作伙伴或用户的任何相关信息、数据等内容。

4.6针对社区电商功能，您不可撤销的同意接入阿里云提供的商品库，

在您的支付宝数字社区小程序页面展示阿里云提供的商品信息。如您不认可前述规则，**请您停止使用本服务。**

4.7

除双方另行协商一致，阿里云在中华人民共和国境内的数据中心存储您使用本服务的全部业务数据。

五、【按现状提供服务】

您理解并同意：

5.1本服务是按照现有技术和条件所能达到的现状提供的。阿里云会尽最大努力保障服务的连贯性和安全性，但阿里云不能随时预见和防范法律、技术以及其他风险，阿里云对此类风险在法律允许的范围内免责，包括但不限于不可抗力、病毒、木马、黑客攻击、系统不稳定、第三方服务瑕疵、政府行为等原因可能导致的服务中断、数据丢失以及其他的损失和风险。

5.2因经营策略安排或调整等原因，不同地区的用户可使用的具体阿里云服务的内容可能会存在差异，具体以阿里云实际提供的为准。

六、【自备设备】

6.1您应当理解，您使用本服务需自行准备与相关服务有关的终端设备（如电脑、移动终端和必要的网络接入设备等装置），并承担所需的费用（如电话费、上网费等费用）。

6.2您理解并同意，您使用本服务时会耗用您的终端设备和带宽等资源。

七、【广告】

7.1您同意阿里云可以自行或由第三方通过短信、电子邮件或电子信息等多种方式向您发送、展示广告或其他信息（包括商业与非商业信息），广告或其他信息的具体发送及展示形式、频次及内容等以阿里云实际提供为准。

7.2阿里云将依照相关法律法规要求开展广告业务。您同意，对本服务中出现的广告，您应审慎判断其真实性和可靠性，除法律明确规定外，您应对因该广告而实施的行为负责。

八、【服务费用】

8.1阿里云的部分服务是以收费方式提供的，如您使用收费服务，请遵守相关的协议。

8.2阿里云可能根据实际需要对收费服务的收费标准、方式进行修改和变更，阿里云也可能会对部分免费服务开始收费。前述修改、变更或开始收费前，阿里云将在相应服务页面进行通知或公告。如果您不同意上述修改、变更或付费内容，则应停止使用该服务。

8.3在阿里云降低收费服务的收费标准或者将收费服务改为免费服务提供时，阿里云保留不对原付费用户提供退费或者费用调整之权利。

九、【第三方提供的产品或服务】

您在阿里云数字社区平台上使用第三方提供的产品或服务时，除遵守本协议约定外，还应遵守第三方的用户协议。阿里云和第三方对可能出现的纠纷在法律规定和约定的范围内各自承担责任。

十、【基于软件提供服务】

若阿里云依托“本软件”向您提供服务，您还应遵守以下约定：

10.1您在使用本服务的过程中可能需要下载软件，对于这些软件，阿里云给予您一项个人的、不可转让及非排他性的许可。您仅可为使用本服务的目的而使用这些软件。

10.2为了改善用户体验、保障服务的安全性及产品功能的一致性等目的，阿里云可能会对软件进行更新。您应该将相关软件更新到最新版本，否则阿里云并不保证软件或服务能正常使用。

10.3阿里云可能为不同的终端设备开发不同的软件版本，您应当根据实际情况选择下载合适的版本进行安装。您可以直接从阿里云平台上获取软件，也可以从得到阿里云授权的第三方获取。如果您从未经阿里云授权的第三方获取软件或与软件名称相同的安装程序，阿里云无法保证该软件或服务能够正常使用，并对因此给您造成的损失不予负责。

10.4除非阿里云书面许可，您不得从事下列任一行为：

（1）删除软件及其副本上关于著作权的信息。

（2）对软件进行反向工程、反向汇编、反向编译，或者以其他方式尝试发现软件的源代码。

（3）对阿里云拥有知识产权的内容进行使用、出租、出借、复制、修改、链接、转载、汇编、发表、出版、建立镜像站点等。

（4）对软件或者软件运行过程中释放到任何终端内存中的数据、软件运行过程中客户端与服务器端的交互数据，以及软件运行所必需的系统数据，进行复制、修改、增加、删除、挂接运行或创作任何衍生作品，形式包括但不限于使用插件、外挂或非经阿里云授权的第三方工具/服务接入软件和相关系统。

（5）通过修改或伪造软件运行中的指令、数据等任何方式，增加、删减、变动软件的功能或运行效果，或者将用于上述用途的软件、方法进行运营或向公众传播，无论这些行为是否为商业目的。

（6）通过非阿里云开发、授权的第三方软件、插件、外挂、系统、设备等任何方式，登录或使用阿里云软件及服务，或制作、发布、传播非阿里云开发、授权的用于登录或使用阿里云软件及服务的第三方软件、插件、外挂、系统、设备等。

十一、【知识产权声明】

11.1阿里云在本服务中提供的内容（包括但不限于网页、文字、图片、音频、视频、图表、计算机软件等）的知识产权归阿里云所有，用户在使用本服务中所产生内容的知识产权归用户或相关权利人所有，除非您与阿里云另有约定。

11.2除另有特别声明外，阿里云提供本服务时所依托软件的著作权、专利权及其他知识产权均归阿里云所有。

11.3阿里云在本服务中所使用的 “阿里云”、“Link”等商业标识，其著作权或商标权归阿里云所有。

11.4上述及其他任何本服务包含的内容的知识产权均受到法律法规保护，未经阿里云、用户或相关权利人书面许可，任何人不得以任何形式进行使用或创造相关衍生作品。

十二、【用户违法违规行为】

12.1您在使用本服务时须遵守法律法规，不得制作、复制、发布、传播含有下列内容的信息或从事相关行为，也不得为制作、复制、发布、传播含有下列内容的信息或从事相关行为提供便利：

（1）反对宪法所确定的基本原则的。

（2）危害国家安全，泄露国家秘密，颠覆国家政权，破坏国家统一的。

（3）损害国家荣誉和利益的。

（4）煽动民族仇恨、民族歧视，破坏民族团结的。

（5）破坏国家宗教政策，宣扬邪教和封建迷信的。

（6）散布谣言，扰乱社会秩序，破坏社会稳定的。

（7）散布淫秽、色情、赌博、暴力、凶杀、恐怖或者教唆犯罪的。

（8）侮辱或者诽谤他人，侵害他人合法权益的。

（9）违反法律法规底线、社会主义制度底线、国家利益底线、公民合法权益底线、社会公共秩序底线、道德风尚底线和信息真实性底线的“七条底线”要求的。

（10）相关法律法规或本协议、相关协议、规则等禁止的。

12.2

如果您在使用本服务过程中违反了相关法律法规或本协议约定，相关国家机关或机构可能会对您提起诉讼、罚款或采取其他制裁措施，并要求阿里云给予协助。**因此给您或者他人造成损害的，您应自行承担全部责任，阿里云不承担任何责任。**

12.3如果您违反本协议约定，阿里云有权进行独立判断并采取相应措施，包括但不限于通过技术手段删除、屏蔽相关内容或断开链接等。同时，阿里云有权视用户的行为性质，采取包括但不限于暂停或终止向您提供服务，限制、中止、冻结或终止您对钉钉账号、支付宝账号的使用，追究法律责任等措施。

12.4您违反本协议约定，导致任何主体损失的，您应当独立承担责任；阿里云因此遭受损失的，您也应当一并赔偿。

十三、【遵守当地法律监管】

13.1

您在使用本服务过程中应当遵守当地相关的法律法规，并尊重当地的道德和风俗习惯。**如果您的行为违反了当地法律法规或道德风俗，您应当为此独立承担责任。**

13.2您应避免因使用本服务而使阿里云违反法律法规或卷入政治和公共事件，否则阿里云有权暂停或终止对您的服务。

十四、【物业收集、发送、传播的内容与投诉处理】

14.1您完全拥有本服务涉及的业务数据，因您上传、储存、发送或传播的数据及内容违反法律法规、部门规章或国家政策而造成的全部结果及责任均由您自行承担。您可自行对业务数据进行删除、更改等操作。

14.2您通过本服务收集、处理、发送或传播的内容（尤其业务信息）应有合法来源，相关内容为您所有或您已获得必要的授权。

14.3如果您通过本服务收集、处理、发送或传播的内容违法违规或侵犯他人权利的，阿里云有权进行独立判断并采取删除、屏蔽或断开链接等措施。

14.4如您被他人投诉或您投诉他人，阿里云有权将争议中相关方的主体信息、联系方式、投诉相关内容等必要信息提供给相关当事方或相关部门，以便及时解决投诉纠纷，保护各方合法权益。

14.5您保证对您在投诉处理程序中提供的信息、材料、证据等的真实性、合法性、有效性负责。

14.6就您使用本服务产生的业务数据，阿里云除执行您的服务要求外，不进行任何未获授权的使用及披露；但以下情形除外：

14.6.1在国家有关机关依法查询或调阅业务数据时，阿里云具有按照相关法律法规或政策文件要求提供配合，并向第三方或者行政、司法等机构披露的义务；在此情形下，阿里云应及时通知物业。

14.6.2双方协商一致，并达成书面协议。

十五、【不可抗力及其他免责事由】

15.1您理解并同意，在使用本服务的过程中，可能会遇到不可抗力等风险因素，使本服务受到影响。不可抗力是指不能预见、不能克服并不能避免且对一方或双方造成重大影响的客观事件，包括但不限于自然灾害如洪水、地震、瘟疫流行和风暴等以及社会事件如战争、动乱、政府行为等。出现上述情况时，阿里云将努力在第一时间与相关单位配合，争取及时进行处理，但是由此给您造成的损失阿里云在法律允许的范围内免责。

15.2在法律允许的范围内，阿里云对以下情形导致的服务中断或受阻不承担责任：

（1）受到计算机病毒、木马或其他恶意程序、黑客攻击的破坏。

（2）用户或阿里云的电脑软件、系统、硬件和通信线路出现故障。

（3）用户操作不当或用户通过非阿里云授权的方式使用本服务。

（4）程序版本过时、设备的老化和/或其兼容性问题。

（5）其他阿里云无法控制或合理预见的情形。

15.3您理解并同意，在使用本服务的过程中，可能会遇到网络信息或其他用户行为带来的风险，阿里云不对任何信息的真实性、适用性、合法性承担责任，也不对因侵权行为给您造成的损害负责。这些风险包括但不限于：

（1）来自他人匿名或冒名的含有威胁、诽谤、令人反感或非法内容的信息。

（2）遭受他人误导、欺骗或其他导致或可能导致的任何心理、生理上的伤害以及经济上的损失。

（3）其他因网络信息或用户行为引起的风险。

15.4阿里云依据本协议约定获得处理违法违规内容的权利，该权利不构成阿里云的义务或承诺，阿里云不能保证及时发现违法行为或进行相应处理。

15.5在任何情况下，您不应轻信借款、索要密码或其他涉及财产的信息。涉及财产操作的，请一定先核实对方身份，并请经常留意阿里云有关防范诈骗犯罪的提示。

十六、【协议的生效与变更】

16.1您使用本服务即视为您已阅读本协议并接受本协议的约束。

16.2阿里云有权在必要时修改本协议条款。您可以在相关服务页面查阅最新版本的协议条款。

16.3本协议条款变更后，如果您继续使用阿里云提供的软件或服务，即视为您已接受变更后的协议。

十七、【服务的变更、中断、终止】

17.1您理解并同意，阿里云基于经营策略的调整，可能会对服务内容进行变更，也可能会中断、中止或终止服务。

17.2在阿里云发生合并、分立、收购、资产转让时，阿里云可向第三方转让本服务下相关资产；阿里云也可在单方通知您后，将本协议下部分或全部服务及相应的权利义务转交由第三方运营或履行。具体受让主体以阿里云通知的为准。

17.3如发生下列任何一种情形，阿里云有权不经通知而中断或终止向您提供服务：

（1）根据法律法规规定您应提交真实信息，而您提供的个人资料不真实、或与注册时信息不一致又未能提供合理证明。

（2）您违反相关法律法规的规定或违反本协议的约定。

（3）按照法律法规规定，司法机关或主管部门的要求。

（4）出于安全的原因或其他必要的情形。

17.4阿里云有权按本协议第8条的约定进行收费。若您未按时足额付费，阿里云有权中断、中止或终止提供服务。

17.5您提供的物业服务协议到期或因其他原因终止，阿里云将终止向您提供服务。阿里云会根据您提供的物业服务协议记载的有效期进行判断执行，如您因故提前终止物业服务，应主动停止使用本服务，并书面通知阿里云。

17.6您有责任自行备份存储在本服务中的数据。如果您的服务被终止，阿里云有权从服务器上永久地删除您的数据，法律法规另有规定的除外。服务中止或终止后，阿里云没有义务向您提供或返还数据。

十八、【管辖与法律适用】

18.1本协议的成立、生效、履行、解释及纠纷解决等相关事宜，均适用中华人民共和国大陆地区法律（不包括冲突法）。

18.2本协议签订地为中华人民共和国浙江省杭州市余杭区。

18.3若您和阿里云之间发生任何纠纷或争议，首先应友好协商解决；协商不成的，您同意将纠纷或争议提交本协议签订地（即浙江省杭州市余杭区）有管辖权的人民法院管辖。

18.4本协议所有条款的标题仅为阅读方便，本身并无实际涵义，不能作为本协议涵义解释的依据。

18.5本协议条款无论因何种原因部分无效或不可执行，其余条款仍有效，对双方具有约束力。

18.6若本协议有中文、英文等多个语言版本，相应内容不一致的，均以中文版的内容为准。

十九、【未成年人使用条款】

19.1若用户未满18周岁，则为未成年人，应在监护人监护、指导下阅读本协议和使用本服务。

19.2未成年人用户涉世未深，容易被网络虚像迷惑，且好奇心强，遇事缺乏随机应变的处理能力，很容易被别有用心的人利用而又缺乏自我保护能力。因此，未成年人用户在使用本服务时应注意以下事项，提高安全意识，加强自我保护：

（1）认清网络世界与现实世界的区别，避免沉迷于网络，影响日常的学习生活。

（2）填写个人资料时，加强个人保护意识，以免不良分子对个人生活造成骚扰。

（3）在监护人或老师的指导下，学习正确使用网络。

（4）避免陌生网友随意会面或参与联谊活动，以免不法分子有机可乘，危及自身安全。

19.3监护人、学校均应对未成年人使用本服务多做引导。特别是家长应关心子女的成长，注意与子女的沟通，指导子女上网应该注意的安全问题，防患于未然。

19.4已满18周岁的成年人因任何原因不具备完全民事行为能力的，参照适用本协议之未成年人使用条款之相关约定。

二十、【其他】

20.1

如果您对本协议或本服务有意见或建议，可与阿里云客户服务部门联系（https://www.aliyun.com/contact），我们会给予您必要的帮助。（正文完）

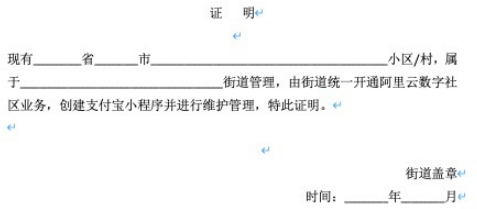
阿里云计算有限公司

2.6.2. 小区申领凭证

本章将为您介绍申请小区所需要的凭证及模板信息

街道管理小区/村庄

由街道代管的小区/村庄或者以街道主导的社区建设场景可提供以下证明材料



证　明

现有_____省_____市_____小区/村，属于_____街道管理，由街道统一开通阿里云数字社区业务，创建支付宝小程序并进行维护管理，特此证明。

街道盖章

时间：_____年_____月

物业公司管理的小区/村庄

由物业公司管理的小区/村庄请提供和业委会签署的服务合同或者街道备案的物业管理合同证明



合同编号：

杭州市物业服务合同

（示范文本）

项目名称：_____

项目地址：_____

委 托 方：_____

受 托 方:_____

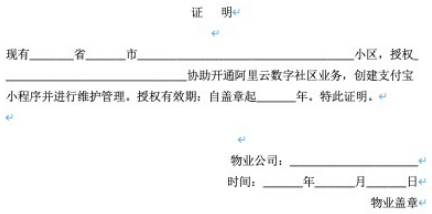
签订日期：_____

杭州市住房保障和房产管理局
杭州市市场监督管理局
杭州市消费者权益保护委员会
杭州市物业管理协会

二〇一八年九月

渠道代理管理的小区/村庄

由物业公司管理的小区/村庄请提供和业委会签署的服务合同或者街道备案的物业管理合同证明，并提供受物业委托证明



证 明

现有_____省_____市_____小区，授权_____协助开通阿里云数字社区业务，创建支付宝小程序并进行维护管理。授权有效期：自盖章起_____年。特此证明。

物业公司：_____
时间：_____年_____月_____日
物业盖章

2.7. 接口开放

2.7.1. 接入准备

本文档介绍数字社区平台接入前置准备工作

1.注册阿里云账号【如有请跳过】

https://www.aliyun.com/

2. 开通应用服务平台【如有请跳过】

https://si.iot.aliyun.com/

3. 创建项目

https://si.iot.aliyun.com/

指导文档：https://help.aliyun.com/document_detail/139047.html?spm=a2c4g.11174283.6.546.3eb152d0wFNRD0

注意创建的时候需要选择 **数字社区** 类型的项目

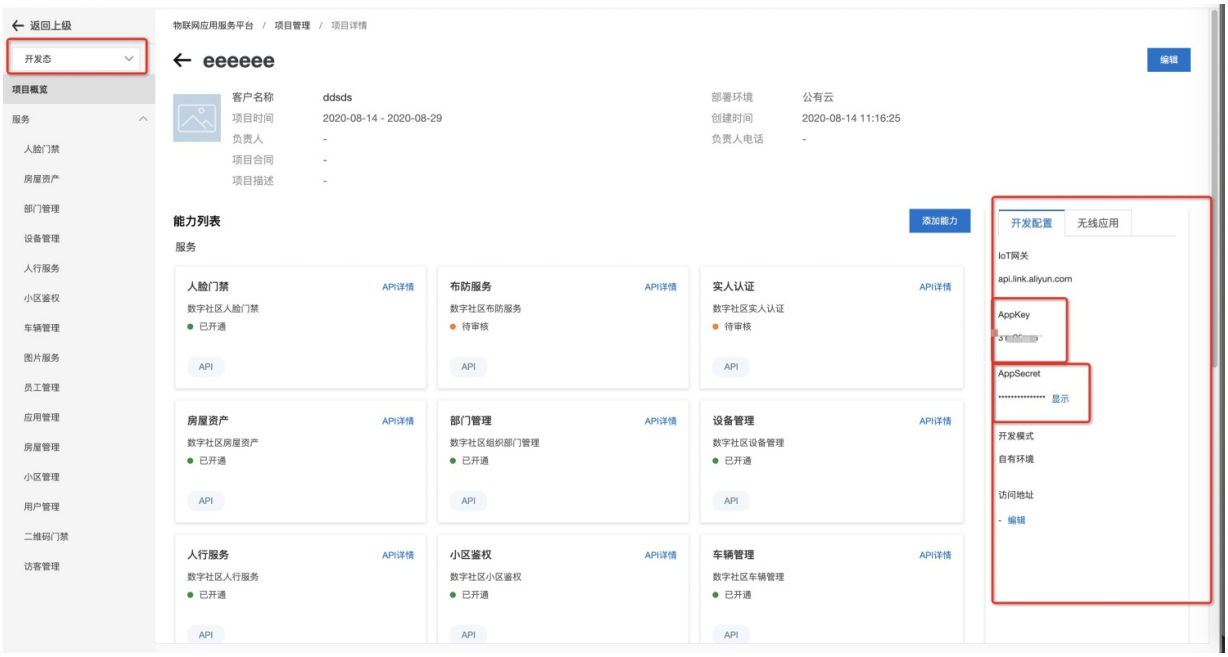


4.等待能力审核通过

提交审核后请联系对接的商务或者产品同学审核

5.获取appkey和appsecrect

确保项目处于开发态，获取appkey和appsecrect用于接口调用。



2.7.2. 通用返回

名称	类型	必填	备注
code	String	是	返回编码，200为成功

data	Object	否	返回结果
message	String	否	返回错误信息
localizedMsg	String	否	本地语言错误消息

2.7.3. 接口鉴权

1.1 接口调用及鉴权

阿里云api接口会为每一个项目颁发一套鉴权秘钥，秘钥包含appkey、appsecrt，您可以通过这套秘钥创建多个小区进行管理，请下载SDK进行接口调用https://github.com/aliyun/iotx-api-gateway-client

1.1.1 获取小区授权token

path	请求方法	版本	描述
/solution/community/paas/token/get	POST	1.0.0	获取云端资源Token；该接口对于同一个res（小区id，参见请求参数说明）在token失效前仅需要调用一次，如果再次调用，则会生成新的cloudToken，并会导致之前的cloudToken失效

请求参数

参数	类型	必填	描述
communityId	Long	是	小区Id

返回参数

参数	类型	描述
cloudToken	String	云端Token
expiresIn	long	Token有效期（单位：ms）

1.1.2 刷新小区授权token

path	版本	描述
/solution/community/paas/token/refresh	1.0.0	刷新云端Token

请求参数

参数	类型	必填	描述
cloudToken	String	是	-

返回参数

参数	类型	描述
cloudToken	String	云端Token
expiresIn	long	Token有效期（单位：ms）

获取token后，即以项目身份访问项目资源，账号，设备，空间，隔离Id均一致

使用方式：

IoTApiRequest request = new IoTApiRequest();

//设置api的版本

request.setApiVer("1.0.0");

request.setCloudToken("XXXXXXX");

注意：下文未做特殊说明情况下。接口访问均加上cloudToken

1.2 数据订阅及鉴权

请查看在线文档

https://help.aliyun.com/document_detail/114862.html?#h1-5-5

数据订阅示例

当正确按照在线文档配置好订阅逻辑后，当有记录变更（新增或者变更时），会受到消息通知payload中带有数据变更的id:

// 模型ID

String modelId;

// 变更数据的ID

List<Long> dataIds;

// 操作类型：insert/update/delete

String operateType;

// 订阅的appId, 以appKey授权时为空

String appId;

使用上文在线文档中的数据查询接口，使用该Id可以查到变更的数据行，除了模型定义的业务字段外，同时返回授权隔离维度，scope_id字段，例如：


```
{
    "code":200,
    "data":"{"count":13,"hasNext":true,"items":
[{"gmt_create":1572834844000,"heartRate":75,"scope_id":[redacted],"id":20,"gmt_modifie
d":1572834844000,"remarks":"test","memberId":"22","timestamp":1572613630000},
{"gmt_create":1572834442000,"heartRate":75,"scope_id":"[redacted]","id":16,"gmt_modified
":1572834442000,"remarks":"test","memberId":"20","timestamp":1572613630000},
{"gmt_create":1572834327000,"heartRate":75,"scope_id":"[redacted]","id":14,"gmt_modified
":1572834327000,"remarks":"test","memberId":"19","timestamp":1572613630000},
{"gmt_create":1572685905000,"heartRate":75,"scope_id":"[redacted]","id":13,"gmt_modified
":1572685905000,"remarks":"test","memberId":"18","timestamp":1572613630000},
{"gmt_create":1572685705000,"heartRate":75,"scope_id":"[redacted]","id":12,"gmt_modified
":1572685705000,"remarks":"test","memberId":"17","timestamp":1572613630000},
{"gmt_create":1572685405000,"heartRate":75,"scope_id":"[redacted]","id":11,"gmt_modified
":1572685405000,"remarks":"test","memberId":"16","timestamp":1572613630000},
{"gmt_create":1572615449000,"heartRate":75,"scope_id":"[redacted]","id":10,"gmt_modified
":1572615449000,"remarks":"test","memberId":"15","timestamp":1572613630000},
{"gmt_create":1572615322000,"heartRate":75,"scope_id":"[redacted]","id":9,"gmt_modified"
:1572615322000,"remarks":"test","memberId":"14","timestamp":1572613630000},
{"gmt_create":1572615209000,"heartRate":75,"scope_id":"[redacted]","id":8,"gmt_modified"
:1572615209000,"remarks":"test","memberId":"13","timestamp":1572613630000},
{"gmt_create":1572614829000,"heartRate":75,"scope_id":"[redacted]","id":7,"gmt_modified"
:1572614829000,"remarks":"test","memberId":"11","timestamp":1572613630000}],"pageNum":1,"pageS
ize":10}",
    "id":"dcdb6168-0f39-482a-b5a6-5647efb13c15"
}
```


该scope_id与项目Id一一对应，使用1.10.2查询小区列表接口，使用该Id作为入参可以对应到具体的小区信息。注意，推送时会有多个隔离Id的推送，过滤使用小区项目Id

的数据。

2.7.4. 小区管理

小区是上面所有接口调用和资源创建的基础，所有设备、人脸、通行记录等数据都是捆绑在小区下面。

1.1 获取POID列表

POID是高德标记每一个已存在小区的唯一id，为了后续数据能够和高德侧发生较好的关联，目前创建小区需要对应的POID信息

接口	版本	描述
/solution/community/paas/poid/query	1.0.1	获取小区piod列表

请求参数

字段名	类型	必填	备注
keywords	String	是	关键词
city	String	是	城市
citylimit	Boolean	否	仅返回指定城市数据
pageSize	Integer	否	每页记录数据
pageNo	Integer	否	当前页

返回结果

参数名	参数描述	数据类型
code	错误码	Int
message	错误信息	String
data	返回值	JsonArray

1.2 获取小区列表

获取当前租户下的有权限的小区列表

接口	版本	描述
/solution/community/paas/community/query	1.0.2	获取小区列表

请求参数

字段名	类型	必填	备注
projectId	String	否	按照项目Id搜索
applyStatus	String	否	审批状态 1审批中 2已审批 3拒绝
pageSize	Integer	否	每页记录数据默认20
pageNo	Integer	否	当前页默认1

返回结果

参数名	参数描述	数据类型
data	返回值	JsonArray 详见 CommunityInfoDTO

CommunityInfoDTO

字段名	类型	备注
communityId	Long	小区id
name	String	名称

applyStatus	Integer	状态：审核中/已审批
projectId	String	小区对应项目Id
gmtCreate	Long	创建时间
poiCode	String	

1.3 创建小区【需审核】

通过高德poid来创建小区，这个过程并非一定要系统完成，也可以让阿里云IoT后台小二帮忙创建（客户提供相关材料）

注意：certificate需要使用1.4.6获取OSS地址并上传

该接口不需要cloudtoken

接口	版本	描述
/solution/community/paas/community/apply	1.0.3	创建小区

请求参数

字段名	类型	必填	备注
poid	String	是	小区poid
name	String	是	小区名称
address	String	是	小区地址
serviceStartTime	String	是	服务起始时间
serviceEndTime	String	是	服务终止时间
certificateUploadId	String	是	证书地址文件上传ID（OSS or 安全url校验）。获取uploadId见“获取OSS上传地址”和“保存上传OSS文件”接口。 上传bizCode=COMMUNITY_FILE
certificateUploadId2	String	否	证书地址2
certificateUploadId3	String	否	证书地址3

返回结果

参数名	参数描述	数据类型
data	JSON	小区信息

小区信息

参数名	参数描述	数据类型
communityId	Long	小区id

1.4 删除小区

只允许删除申请失败的小区，不支持删除已经审批通过的小区

该接口不需要cloudtoken

接口	版本	描述
/solution/community/paas/community/delete	1.0.1	删除小区

请求参数

字段名	类型	必填	备注
communityId	Long	是	小区id

返回结果

通用网关消息体，无业务字段

1.5 查询小区信息【包括审核状态】

该接口不需要cloudtoken

接口	版本	描述
/solution/community/paas/community/detail/get	1.0.3	获取小区详情

请求参数

字段名	类型	必填	备注
communityId	Long	是	小区id

返回结果

参数名	参数描述	数据类型
data	返回值	JsonArray详见 CommunityDetailDTO

CommunityDetailDTO

字段名	类型	备注
communityId	Long	小区id
name	String	名称
applyStatus	Integer	状态：审核中/已审批
address	String	小区省份
serviceStartTime	Date	服务开始时间
serviceEndTime	Date	服务终止时间
poiCode	String	poid
spaceId	String	小区根空间Id
projectId	String	项目Id

gmtCreate	Long	创建时间
gmtModified	Long	更新时间

1.6 根据设备查询小区信息

接口	版本	描述
/solution/community/paas/device/community/get	1.0.3	根据项目Id获取

请求参数

字段名	类型	必填	备注
deviceIdentity	JsonObject	是	详见DeviceIdentityDTO

DeviceIdentityDTO

字段名	类型	必填	备注
type	String	是	IOT_ID/PK_AND_DN
iotId	String	否	设备id
productKey	String	否	产品PK
deviceName	String	否	产品DN

返回结果

参数名	参数描述	数据类型
data	返回值	JsonObject如下

CommunityDetailDTO

字段名	类型	备注
communityId	Long	小区id
name	String	名称
imgOssUrl	String	小区证明文件
applyStatus	Integer	状态：审核中/已审批
address	String	小区省份
serviceStartTime	Date	服务开始时间
serviceEndTime	Date	服务终止时间
poiCode	String	poid

spaceId	String	空间ID
spaceType	String	空间类型
projectId	String	项目Id
gmtCreate	Long	创建时间
gmtModified	Long	更新时间

2.7.5. 房屋管理

1.1 新建小区下区域

Path路径	/solution/community/paas/space/sub/create
请求方式	post
接口版本	1.0.1

请求参数

参数	类型	必填	描述
name	String	是	空间名称
parentId	String	是	所有区域都是小区下的空间。 如果创建小区下的一级空间，parentId为小区的根空间ID（见“获取小信息”接口） 否则是父区域ID
typeCode	String	是	空间类型代码： • area 区域 • building 幢 • unit 单元 • house 房屋 • parking_area 停车区域 • parking_space_channel 车道空间（出入口）
direction	Integer	否	出入口方向，当空间类型为parking_space_channel时必填： • 0 进口 • 1 出口

返回参数

参数	类型	描述
spaceId	String	区域ID

1.2 修改小区下区域

Path路径	/solution/community/paas/space/update

请求方式	post
接口版本	1.0.1

请求参数

参数	类型	必填	描述
name	String	是	空间名称
spaceId	String	是	区域空间ID
direction	Integer	否	出入口方向，当空间类型为parking_space_channel时可选： • 0 进口 • 1 出口

返回参数

参数	类型	描述
无		

1.3 删除小区下区域

Path路径	/solution/community/paas/space/delete
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
spaceId	String	是	空间ID

返回参数

参数	类型	描述
无		

1.4 层级查询子空间

按照小区空间层级结构逐层查询

Path路径	/solution/community/paas/space/query
请求方式	post
接口版本	1.0.2

请求参数

参数	类型	必填	描述
spaceId	String	是	空间ID
pageNo	Integer	是	页码
pageSize	Integer	是	页面大小
typeCodeList	JSONArray<String>	否	空间类型代码 • area 区域 • building 幢 • unit 单元 • house 房屋 • parking_area 停车区域 • parking_space_channel 车道空间（出入口）

返回参数

参数	类型	描述
total	Integer	
pageNo	Integer	页码
pageSize	Integer	页面大小
data	JSONArray	
spaceId	String	空间ID
name	String	空间名称
typeCode	String	空间类型代码 • area 区域 • building 幢 • unit 单元 • house 房屋 • parking_area 停车区域 • parking_space_channel 车道空间（出入口）
parentId	String	父空间ID
direction	Integer	出入口方向，当空间类型为parking_space_channel时该参数有效： • 0 进口 • 1 出口

1.5 平铺查询子空间

按照空间类型查询小区下所有满足条件的子空间，不存在层级结构，平铺返回。

Path路径	/solution/community/paas/space/sub/query
请求方式	post

接口版本	1.0.1

请求参数

参数	类型	必填	描述
pageNo	Integer	是	页码
pageSize	Integer	是	页面大小
typeCode	String	是	空间类型代码 • area 区域 • building 幢 • unit 单元 • house 房屋 • parking_area 停车区域 • parking_space_channel 车道空间（出入口）

返回参数

参数	类型	描述
total	Integer	总数
pageNo	Integer	页码
pageSize	Integer	页面大小
data	JSONArray	空间列表
spaceId	String	空间ID
name	String	空间名称
typeCode	String	空间类型代码 • area 区域 • building 幢 • unit 单元 • house 房屋 • parking_area 停车区域 • parking_space_channel 车道空间（出入口）
parentId	String	父区域ID
direction	Integer	出入口方向，当空间类型为parking_space_channel时该参数有效： • 0 进口 • 1 出口

1.6 查询空间详情

Path路径	/solution/community/paas/space/get

请求方式	post
接口版本	1.0.2

请求参数

参数	类型	必填	描述
spaceIds	JSONArray<String>	是	区域ID列表

返回参数

参数	类型	描述
code	Integer	响应码，200：成功
message	String	错误消息
localizedMsg	String	本地语言错误消息
data	JSONArray<String>	响应结果
data		
spaceId	String	空间id
name	String	空间名称
typeCode	String	空间类型代码 • area 区域 • building 幢 • unit 单元 • house 房屋 • parking_area 停车区域 • parking_space_channel 车道空间（出入口）
direction	Integer	出入口方向，当空间类型为parking_space_channel时该参数有效： • 0 进口 • 1 出口
namePath	String	名称路径
parentId	String	父空间id
pathIds	JSONArray<String>	空间的path数组
path	String	空间IDpath

1.7 新增房屋资产

Path路径	/solution/community/household/spaceasset/add
请求方式	post

接口版本	1.0.1

请求参数

参数	类型	必填	描述
houseId	String	是	房屋ID
type	Integer	是	资产类型： • 1 车位 • 2 车库 • 3 储藏室
name	String	是	车位/车库/储藏室号码

返回参数

参数	类型	描述
data	String	资产ID

1.8 更新房屋资产

Path路径	/solution/community/household/spaceasset/update
请求方式	post
接口版本	1.0.1

请求参数

参数	类型	必填	描述
houseId	String	是	房屋ID
assetId	Long	是	资产ID
type	Integer	否	资产类型： • 1 车位 • 2 车库 • 3 储藏室
name	String	否	车位/车库/储藏室号码

返回参数

参数	类型	描述
无		

1.9 删除房屋资产

Path路径	/solution/community/household/spaceasset/delete

请求方式	post
接口版本	1.0.1

请求参数

参数	类型	必填	描述
houseId	String	是	房屋ID
assetId	Long	是	资产ID

返回参数

参数	类型	描述
无		

1.10 查询房屋资产

Path路径	/solution/community/household/spaceasset/list
请求方式	post
接口版本	1.0.1

请求参数

参数	类型	必填	描述
houseId	String	是	房屋ID

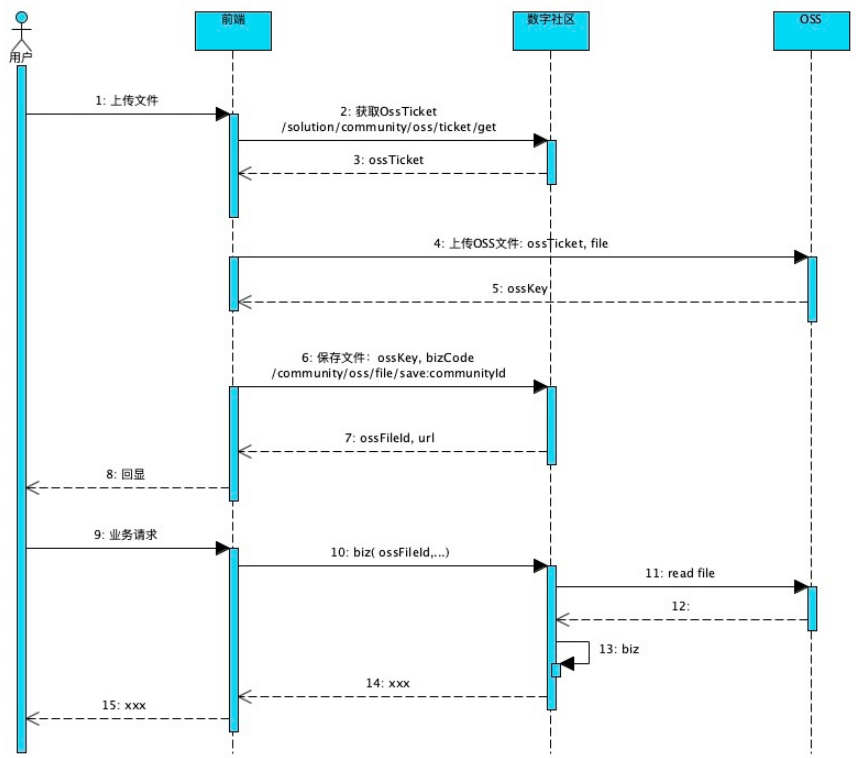
返回参数JSONArray

参数	类型	描述
assetId	Long	资产ID
type	Integer	资产类型： • 1 车位 • 2 车库 • 3 储藏室
name	String	车位/车库/储藏室号码

2.7.6. 图片服务

1. OSS流程示意

1. 获取ossTicket（/solution/community/oss/ticket/get）
2. 使用ossTicket连接OSS服务并上传，返回ossKey；
3. 使用ossKey保存文件，获取uploadId（/solution/community/oss/file/save）
4. 在需要上传文件的API中，使用uploadId作为参数，具体见接口说明。部分仍使用ossKey的接口，将来可能会升级为使用uploadId.



2. 获取OSS上传地址

Path路径	`/solution/community/oss/ticket/get`
请求方式	post
接口版本	1.0.2

请求参数

字段名	类型	必填	备注
type	String	是	ticket类型枚举；图片为picture，最大为2MB.

返回结果

字段名	类型	备注
data	JSON	ticket信息

Ticket信息

字段名	类型	备注
accessId	String	oss访问ID
policy	String	编码
signature	String	签名

dir	String	OSS路径
host	String	OSS服务host
expire	Long	过期时间

> 说明
> api接口1.0.1以前的版本，ossKey可以直接在业务接口中使用。此版本接口即将下线，目前仅兼容已对接过的ISV；
> api接口1.0.2以后的版本，需要在完成文件上传后，调用 “保存上传OSS文件” 接口获取uploadId（上传文件ID）

上传示例：

https://help.aliyun.com/document_detail/31988.html?pos=4 注意，示例代码中的对应字段，请使用上述返回值替代。一般用于从前端上传。

后端上传示例

```
public static void upload(File file,String host, String dir, String policy, String accessKeyId, String signature)throws ClientProtocolException, IOException{
        CloseableHttpClient httpClient = HttpClientBuilder.create().build();
        CloseableHttpResponse httpResponse = null;
        RequestConfig requestConfig = RequestConfig.custom().setConnectTimeout(200000).setSocketTimeout(200000000).build();
        HttpPost httpPost = newHttpPost("http://" + host);
        httpPost.setConfig(requestConfig);
        MultipartEntityBuilder multipartEntityBuilder= MultipartEntityBuilder.create();
        multipartEntityBuilder.setCharset(Charset.forName("UTF-8"));
        multipartEntityBuilder.setMode(HttpMultipartMode.BROWSER_COMPATIBLE);
        //处理文件名乱码
        ContentType contentType = ContentType.create(HTTP.PLAIN_TEXT_TYPE, HTTP.UTF_8);


        multipartEntityBuilder.addTextBody("key",dir+"/" + file.getName(),contentType);
        multipartEntityBuilder.addTextBody("policy",policy);
        multipartEntityBuilder.addTextBody("OSSAccessKeyId",accessKeyId);
        multipartEntityBuilder.addTextBody("success_action_status","200");
        multipartEntityBuilder.addTextBody("callback","");
        multipartEntityBuilder.addTextBody("signature",signature);
        multipartEntityBuilder.addBinaryBody("file",file);


        HttpEntity httpEntity = multipartEntityBuilder.build();
        httpPost.setEntity(httpEntity);


        httpResponse = httpClient.execute(httpPost);
        HttpEntity responseEntity = httpResponse.getEntity();
        int statusCode= httpResponse.getStatusLine().getStatusCode();
        if(statusCode == 200){
            BufferedReader reader = new BufferedReader(new InputStreamReader(responseEntity.getContent()));
            StringBuffer buffer = new StringBuffer();
            String str = "";
            while(!StringUtils.isEmpty(str = reader.readLine())) {
                buffer.append(str);
            }


            System.out.println(buffer.toString());
        }


        httpClient.close();
        if(httpResponse!=null){
            httpResponse.close();
        }
    }
```

3. 保存上传OSS文件

在 “获取OSS上传地址” 后，完成OSS文件上传，需要一步保存操作，把上传ossKey换成uploadId。

Path路径	`/solution/community/oss/file/save`

请求方式	post
接口版本	1.0.0

请求参数

字段名	类型	必填	备注
communityId	Long	否	小区ID
tmpOssKey	String	是	临时区的OSSkey
bizCode	String	是	业务标识，默认使用DEFAULT，其它取值： l 小区物业认证文件：COMMUNITY_FILE l 智能识别图片：HOUSEHOLD_FACE
urlTimeout	Integer	否	不指定时默认5分钟有效（300秒）

返回结果

字段名	类型	备注
uploadId	String	上传ID
url	String	访问URL，有效期由urlTimeout指定
urlTimeout	Integer	失效时间（秒）

注意：这个接口需要和”获取OSS上传地址配合使用。先获取临时区ossTicket，完成文件上传后，调用此接口获取uploadId（上传文件ID）。

4. 检测智能识别图片质量

Path路径	/solution/community/faceimage/check
请求方式	post
接口版本	1.0.1

请求参数

参数	类型	必填	描述
width	Integer	是	智能识别图片的宽度（像素）
height	Integer	是	智能识别图片的高度（像素）
url	String	是	智能识别图片的oss地址

返回结果

无

智能识别图使用限制

检查点	定义

文件格式	jpg
分辨率	800*800及以下
文件大小（byte）	10K~1M
智能识别数量	1
姿态	正脸
智能识别占整体图面积	大于1/4

2.7.7. 用户管理

用户组管理

支持四种类型的用户组：业主、物业、访客和黑名单。小区完成申请后，会自动添加四个不同类型的分组，通过查询接口可以查询到。

四个自动添加的分组名称为：业主、物业、访客和黑名单。删除时，四个默认的用户组是不能删掉的。

增加的住户会自动添加到业主组里。ISV也可以创建自定义用户组。

1.1新建用户组

用户组是管理人员的基础单位，用户组中可以添加用户，也可以将用户组中的智能识别下发到AIBOX或者门禁设备中。

Path路径	/solution/community/paas/user/group/add
请求方式	post
接口版本	1.0.1

字段名	类型	必填	备注
userGroupName	String	是	用户组名称
userGroupType	String	是	用户组类型RESIDENT - 居民VISITOR - 访客PROPERTY - 物业BLACKLIST - 黑名单

字段名	类型	备注
data	JSON	用户组id

用户组id

字段名	类型	备注
userGroupId	String	用户组id

1.2更新用户组

更新设备组的名称。

Path路径	/solution/community/paas/user/group/update
请求方式	post

接口版本	1.0.1

字段名	类型	必填	备注
userGroupName	String	是	用户组名称
userGroupId	String	是	用户组id

通用网关消息体，无业务字段

1.3删除用户组

用户组布防，是将对应的用户组智能识别同步至边缘AIBOX中，由AIBOX进行人员比对后形成人行数据，后续用户组的用户信息进行变更时，系统会自动进行用户照片的同步。

Path路径	/solution/community/paas/user/group/delete
请求方式	post
接口版本	1.0.1

字段名	类型	必填	备注
userGroupId	String	是	用户组id

通用网关消息体，无业务字段

1.4查询用户组

根据用户组名称查询已有的用户组列表。

Path路径	/solution/community/paas/user/group/query
请求方式	post
接口版本	1.0.1

字段名	类型	必填	备注
userGroupName	String	否	用户组名称
pageNo	Integer	是	页码
pageSize	Integer	是	页面数据条数

字段名	类型	备注
data	JSONArray	用户组结构

用户组结构

字段名	类型	备注
communityId	Long	小区Id
userGroupId	String	用户组ID

userGroupType	String	用户组类型
userGroupName	String	用户组名称
isDefault	Integer	是否系统默认，1-系统创建 0-用户创建，通过7、8、13三章创建的用户都会标识为1-系统创建

1.5结构化查询用户信息

根据用户组id查询组内的用户列表。

Path路径	/solution/community/paas/user/query
请求方式	post
接口版本	1.0.5

字段名	类型	必填	备注
userGroupId	String	否	用户组id
userName	String	否	用户名
mobile	String	否	手机号
modifiedStart	Long	否	更新起始毫秒时间（含）
modifiedEnd	Long	否	更新结束毫秒时间（含）
orderBy	JsonObject	否	详见OrderByDTO 排序规则，默认 gmt_create 降序排列
pageNo	Integer	是	页码
pageSize	Integer	是	页面数据条数

OrderByDTO

字段名	类型	必填	备注
columnName	String	是	排序列名，需要direction 配合使用，支持的输入：gmt_create 创建时间 gmt_modified 更新时间
direction	String	是	排序方向，需要columnName配合使用 支持的输入desc 降序

字段名	类型	备注
data	JSONArray	用户结构

人员结构

字段名	类型	备注
identityId	String	用户ID
faceId	String	智能识别ID
userName	String	用户名
faceImageUrl	String	用户图片地址
idNo	String	身份证号
userGroups	JSONArray	用户组列表
age	Integer	年龄
sex	Integer	性别：1男，2女
mobile	String	电话
gmtCreate	Long	创建时间
gmtModified	Long	更新时间
nationality	String	国家/地区
ext	String	自定义配置信息json结构

userGroups详情：

字段名	类型	备注
userGroupId	String	用户组ID
userGroupName	String	用户组名称
userGroupType	String	用户组类型

1.6查询用户信息

通过用户Id或智能识别Id查询人员信息

Path路径	/solution/community/paas/user/get
请求方式	post
接口版本	1.0.2

字段名	类型	必填	备注
identityId	String	否	用户Id
faceId	String	否	智能识别Id

字段名	类型	备注
data	JSONObject	人员结构

人员结构

字段名	类型	备注
identityId	String	用户ID
faceId	String	智能识别ID
userName	String	用户名
faceImageUrl	String	用户图片地址
idNo	String	身份证号
userGroups	JSONArray	用户组列表
age	Integer	年龄
sex	Integer	性别：1男，2女
mobile	String	电话

userGroups详情：

字段名	类型	备注
userGroupId	String	用户组ID
userGroupName	String	用户组名称
userGroupType	String	用户组类型

1.7智能识别图片查询用户信息

智能识别检索用户，是通过一张用户的照片来检索用户组中的人员信息

Path路径	/solution/community/paas/user/getbyimage
请求方式	post
接口版本	1.0.1

字段名	类型	必填	备注
ossFileKey	String	是	智能识别图片地址，OSS文件的KEY

字段名	类型	备注
data	JSONArray	人员结构

人员结构

字段名	类型	备注
identityId	String	用户ID
faceId	String	智能识别ID
userName	String	用户名
faceImageUrl	String	用户图片地址
idNo	String	身份证号
userGroups	JSONArray	用户组列表
age	Integer	年龄
sex	Integer	性别：1男，2女
mobile	String	电话

userGroups详情：

字段名	类型	备注
userGroupId	String	用户组ID
userGroupName	String	用户组名称
userGroupType	String	用户组类型

用户管理

2.1添加用户

Path路径	/solution/community/paas/user/add
请求方式	post
接口版本	1.0.2

字段名	类型	必填	备注
userGroupId	String	是	用户组Id
userName	String	是	用户名 长度不能超过12，且只能包含字母、数字、中文
mobile	String	是	手机号
idNo	String	否	身份证号码
sex	Integer	否	性别：1男，2女

age	Integer	否	年龄
faceImgUploadId	String	否	智能识别图片上传ID地址。获取uploadId见“获取OSS上传地址”和“保存上传OSS文件”接口。上传bizCode=HOUSEHOLD_FACE

字段名	类型	备注
data	JSON	用户id

用户id

字段名	类型	备注
identityId	String	用户Id
faceId	String	用户id

2.2修改用户

Path路径	/solution/community/paas/user/update
请求方式	post
接口版本	1.0.2

字段名	类型	必填	备注
identityId	String	是	用户id
userName	String	否	用户名
idNo	String	否	身份证号码
sex	Integer	否	性别：1男，2女
age	Integer	否	年龄
faceImgUploadId	String	否	智能识别图片上传ID。获取uploadId见“获取OSS上传地址”和“保存上传OSS文件”接口。上传bizCode=HOUSEHOLD_FACE

通用网关消息体，无业务字段

2.3用户组添加用户

Path路径	/solution/community/paas/user/group/member/add
请求方式	post
接口版本	1.0.1

字段名	类型	必填	备注

userGroupId	String	是	用户组Id
identityId	String	是	用户id

通用网关消息体，无业务字段

2.4用户组删除用户

Path路径	/solution/community/paas/user/group/member/delete
请求方式	post
接口版本	1.0.1

字段名	类型	必填	备注
userGroupId	String	是	用户组Id
identityId	String	是	用户id

通用网关消息体，无业务字段

2.5删除用户人脸

调用此接口后，平台上的用户相关的人脸图都会被删除

Path路径	/solution/community/paas/user/face/delete
请求方式	post
接口版本	1.0.0

字段名	类型	必填	备注
identityId	String	是	用户id

出参

无

2.6 删除用户

调用此接口后，平台上的用户关联的信息会被删除，仅保留identityId。

Path路径	/solution/community/paas/user/delete
请求方式	post
接口版本	1.0.0

字段名	类型	必填	备注
identityId	String	是	用户id

出参

无

2.7.8. 居民管理

1.1 新增房屋成员

Path路径	/solution/community/paas/house/member/save
请求方式	post
接口版本	1.0.5

请求参数

参数	类型	必填	描述
communityId	Long	否	小区ID
houseSpaceId	String	是	房屋空间的spaceId 要求房屋空间的typeCode为house类型
type	Integer	是	成员类型： 1业主本人；2亲属；3租客；4朋友；5同事；6装修人员；7保姆；8司机；9其它
name	String	是	姓名，更新时不允许修改
sex	Integer	否	性别：1 男，2 女，更新时不允许修改
idNum	String	否	身份证号
phoneNum	String	否	手机号
faceImgUploadId	String	否	智能识别照片上传文件ID，见“获取OSS上传地址”和“保存上传OSS文件”接口 上传bizCode=HOUSEHOLD_FACE
age	Integer	否	年龄
degree	Integer	否	学历：0无； 1小学；2初中；3职高；4普高；5中专；6大专；7本科；8硕士；9博士
health	String	否	健康状态
marriage	Integer	否	婚姻情况：0未婚； 1已婚； 2离异
politics	Integer	否	政治面貌：0群众；1党员；2预备党员；3积极分子；4入党申请人；5 民主党派
company	String	否	工作单位
registry	String	否	籍贯
residenceAddress	String	否	户籍所在地
registryCode	String	否	籍贯编码
residenceAddressCode	String	否	户籍所在地编码

nationality	String	否	国家/地区
ext	String	否	自定义配置信息

返回参数

参数	类型	描述
identityId	String	用户ID

1.2 修改房屋成员

Path路径	/solution/community/paas/house/member/save
请求方式	post
接口版本	1.0.5

请求参数

参数	类型	必填	描述
houseSpaceId	String	是	房屋空间
identityId	String	是	更新成员信息
type	Integer	否	成员类型： 1业主本人；2亲属；3租客；4朋友；5同事；6装修人员；7保姆；8司机；9其它
idNum	String	否	身份证号，更新时可选
faceImgUploadId	String	否	智能识别照片上传文件ID，见“获取OSS上传地址”和“保存上传OSS文件”接口 上传bizCode=HOUSEHOLD_FACE
age	Integer	否	年龄
degree	Integer	否	学历：0无； 1小学；2初中；3职高；4普高；5中专；6大专；7本科；8硕士；9博士
health	String	否	健康状态
marriage	Integer	否	婚姻情况：0未婚； 1已婚； 2离异
politics	Integer	否	政治面貌：0群众；1党员；2预备党员；3积极分子；4入党申请人；5 民主党派
company	String	否	工作单位
registry	String	否	籍贯
residenceAddress	String	否	户籍所在地

registryCode	String	否	籍贯编码
residenceAddressCode	String	否	户籍所在地编码
nationality	String	否	国家/地区
ext	String	否	自定义配置信息

返回参数

参数	类型	描述
identityId	String	用户ID

1.3 删除房屋成员

Path路径	/solution/community/paas/house/member/delete
请求方式	post
接口版本	2.0.0

请求参数

参数	类型	必填	描述
houseSpaceId	String	是	房屋空间
identityId	String	是	要删除的房屋成员 如果发起用户是业主，可以删除其它房屋成员； 如果不是业主，只能删除自己在房屋中的关系。

返回参数

参数	类型	描述
无		

1.4 查询房屋成员

Path路径	/solution/community/paas/house/member/list
请求方式	post
接口版本	3.0.3

请求参数

参数	类型	必填	描述
houseSpaceId	String	是	房屋空间
identityId	String	否	用户ID

communityId	String	否	小区ID
types	JSONArray<Integer>	否	成员类型：1业主本人；2亲属；3租客；4朋友；5同事；6装修人员；7保姆；8司机；9其它

返回参数

JSONArray

参数	类型	描述
identityId	String	用户ID
type	Integer	关系： 1业主本人；2亲属；3租客；4朋友；5同事；6装修人员；7保姆；8司机；9其它
alipayUid	String	支付宝用户ID
userName	String	姓名
sex	Integer	性别：1男 2女
idNo	String	身份证号
mobile	String	手机号
faceImageUrl	String	智能识别照片OSS KEY
source	String	来源：QRCODE, INPUT
age	Integer	年龄
degree	Integer	学历：0无； 1小学；2初中；3职高；4普高；5中专；6大专；7本科；8硕士；9博士
health	String	健康状态
marriage	Integer	婚姻情况：0未婚； 1已婚； 2离异
politics	Integer	政治面貌：0群众；1党员；2预备党员；3积极分子；4入党申请人；5 民主党派
company	String	工作单位
registry	String	籍贯
residenceAddress	String	户籍所在地
registryCode	String	籍贯编码
residenceAddressCode	String	户籍所在地编码
nationality	String	国家/地区

approveComment	String	审核拒绝原因
status	Integer	住户状态 1已迁入、2待审核、3未通过、4已迁出
ext	String	自定义扩展信息

1.5 查询住户列表

Path路径	/solution/community/paas/household/query
请求方式	post
接口版本	1.0.5

请求参数

参数	类型	必填	描述
communityId	Long	否	小区Id
houseSpaceId	String	否	房屋空间ID
mobile	String	否	手机号
type	Integer	否	关系： 1业主本人；2亲属；3租客；4朋友；5同事；6装修人员；7保姆；8司机；9其它
userName	String	否	人名
modifiedStart	Long	否	更新起始毫秒时间（含）
modifiedEnd	Long	否	更新结束毫秒时间（含）
orderBy	JsonObject	否	详见OrderByDTO 排序规则：默认 gmt_create 降序排列。如果orderBy非空，则columnName和direction必填
pageNo	Integer	否	页码
pageSize	Integer	否	页面大小

OrderByDTO

字段名	类型	必填	备注
columnName	String	是	排序列名，需要direction 配合使用，支持的输入：gmt_create 创建时间 gmt_modified 更新时间
direction	String	是	排序方向：需要columnName配合使用 支持的输入desc 降序

返回参数

参数	类型	描述
total	Integer	
pageNo	Integer	页码
pageSize	Integer	页面大小
data	JSONArray	
identityId	String	用户ID
type	Integer	关系： 1业主本人；2亲属；3租客；4朋友；5同事；6装修人员；7保姆；8司机；9其它
alipayUid	String	支付宝用户（该字段不对外输出）
userName	String	姓名
sex	Integer	性别：1男 2女
idNo	String	身份证号
mobile	String	手机号
faceImageUrl	String	智能识别照片OSS KEY
source	String	来源：QRCODE, INPUT
age	Integer	年龄
degree	Integer	学历：0无； 1小学；2初中；3职高；4普高；5中专；6大专；7本科；8硕士；9博士
health	String	健康状态
marriage	Integer	婚姻情况：0未婚； 1已婚； 2离异
politics	Integer	政治面貌：0群众；1党员；2预备党员；3积极分子；4入党申请人；5 民主党派
company	String	工作单位
registry	String	籍贯
residenceAddress	String	户籍所在地
registryCode	String	籍贯编码
residenceAddressCode	String	户籍所在地编码
gmtCreate	Long	创建时间

gmtModified	Long	更新时间
nationality	String	国家/地区
ext	String	自定义扩展信息
gmtCreate	Long	创建时间
gmtModified	Long	更新时间

1.6 查询用户的房屋列表

Path路径	/solution/community/paas/household/house/list
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
identityId	String	是	用户ID

返回参数

JSONArray

参数	类型	描述
communitySpaceId	String	小区空间ID
houseSpaceId	String	房屋空间ID
communityId	String	小区ID

2.7.9. 访客管理

访客申请是指访客在到访前，由业务或物业填写访客信息，再验证的流程。

1.1 新增访客申请

Path路径	/solution/community/house/visitor/apply/add
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
houseId	String	是	房屋ID
startTime	String	是	到访时间 格式：yyyy-MM-dd HH:mm:ss

endTime	String	是	结束时间 格式：yyyy-MM-dd HH:mm:ss
visitors	JSONArray	是	访客列表
--name	String	是	姓名
--type	String	是	身份类型：1业主本人，2亲属，3租客，4朋友，5同事，6装修人员，7保姆，8司机，9其它
--idCard	String	否	身份证号
--phone	String	否	手机号
--gender	Integer	否	性别
--plateNum	String	否	车牌

返回参数

参数	类型	描述
data	JSONArray	
--applyId	String	访问申请ID

1.2 作废访客申请

Path路径	/solution/community/house/visitor/apply/cancel
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
applyId	String	是	访问申请ID

返回参数

参数	类型	描述
无		

1.3 人工访客核销

Path路径	/solution/community/house/visitor/apply/confirm
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
applyId	String	是	访问申请ID

返回参数

参数	类型	描述
无		

1.4 查询访客详情

根据访客申请的ID查询访客详情

Path路径	/solution/community/house/visitor/apply/get
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
applyId	String	否	访客用户ID

返回参数

参数	类型	描述
data	JSON	访客数据

访客数据

字段名	类型	备注
applyId	String	申请ID
name	String	姓名
communityName	String	小区名
houseName	String	房屋名
type	Integer	访客类型
phone	String	手机号（脱敏）
idCard	String	身份证号（脱敏）
identityId	String	提交者身份Id
visitorId	String	访客身份ID

gender	Integer	性别
plateNum	String	车牌
startTime	String	到访时间
endTime	String	结束时间
checkStatus	Integer	验证状态0未到访，1已到访，2废弃

1.5 批量查询访客

Path路径	/solution/community/house/visitor/apply/query
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
startTime	String	是	开始时间
endTime	String	是	结束时间
houseId	String	否	房屋ID
pageNo	Integer	是	页号
pageSize	Integer	是	页大小

返回参数

参数	类型	描述
pageNo	Integer	页号
pageSize	Integer	页大小
total	Integer	总数据量
data	JSONArray	访客数据列表

访客数据

字段名	类型	备注
applyId	String	申请ID
name	String	姓名
communityName	String	小区名

houseName	String	房屋名
type	Integer	访客类型
phone	String	手机号（脱敏）
idCard	String	身份证号（脱敏）
identityId	String	提交者身份id
gender	Integer	性别
plateNum	String	车牌
startTime	String	到访时间
endTime	String	结束时间
checkStatus	Integer	验证状态0未到访，1已到访，2废弃

2.7.10. 车辆管理

1.1 添加车辆

Path路径	/solution/community/household/user/vehicle/add
请求方式	post
接口版本	1.0.1

请求参数

字段名	类型	必填	备注
identityId	String	是	用户Id
vehicleList	JSONArray	是	车辆信息列表（每人在同一个小区内最多10辆车）

vehicleList车辆信息

字段名	类型	必填	备注
plateNumber	String	是	车牌号
brand	String	否	车辆品牌
color	Integer	否	车辆颜色：0其他；1白； 2灰； 3黄； 4粉； 5红； 6紫； 7绿； 8蓝； 9棕； 10黑；11 青

返回结果

字段名	类型	备注

data	JSONArray	车辆id列表，每个列表元素类型为Long

1.2 修改车辆

Path路径	/solution/community/household/user/vehicle/update
请求方式	post
接口版本	1.0.1

请求参数

字段名	类型	必填	备注
vehicleList	JSONArray	是	车辆信息列表

vehicleList车辆信息（最多10个）

字段名	类型	必填	备注
id	Long	是	车辆ID
plateNumber	String	否	车牌号
brand	String	否	车辆品牌
color	Integer	否	车辆颜色：0其他；1白； 2灰； 3黄； 4粉； 5红； 6紫； 7绿； 8蓝； 9棕； 10黑；11 青

返回结果

字段名	类型	备注
data	JSONArray	车辆id列表，每个列表元素类型为Long

1.3 查询车辆

Path路径	/solution/community/household/user/vehicle/query
请求方式	post
接口版本	1.0.1

请求参数

字段名	类型	必填	备注
identityId	String	否	用户Id
plateNumber	String	否	车牌号，和用户id二选一

返回结果

字段名	类型	备注
data	JSONArray	详见DigitalCommunityVehicleDTO定义

DigitalCommunityVehicleDTO

字段名	类型	备注
id	Long	车辆ID
identityId	String	用户Id
plateNumber	String	车牌号
brand	String	车辆品牌
color	Integer	车辆颜色：0其他；1白； 2灰； 3黄； 4粉； 5红； 6紫； 7绿； 8蓝； 9棕； 10黑；11 青

1.4 删除车辆

Path路径	/solution/community/household/user/vehicle/delete
请求方式	post
接口版本	1.0.1

请求参数

字段名	类型	必填	备注
vehicleList	JSONArray	是	车辆信息列表（上限10个）

vehicleList车辆信息

字段名	类型	必填	备注
id	Long	是	车辆ID

返回结果

通用网关消息体，无业务字段

2.7.11. 物业员工管理

1.1 创建员工信息

Path路径	/solution/community/org/employee/create
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述

mobile	String	是	手机号，最大长度20
name	String	是	名称，最大长度12
faceImgUploadId	String	否	智能识别图片上传文件ID 上传bizCode=HOUSEHOLD_FACE

返回参数

参数	类型	描述
identityId	String	员工ID

1.2 修改员工信息

Path路径	/solution/community/org/employee/modify
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
identityId	String	是	用户ID
mobile	String	否	手机号，为空时表示不修改字段，最大长度20
name	String	否	姓名，为空时表示修改字段，最大长度12
faceImgUploadId	String	否	智能识别图片上传文件ID 上传bizCode=HOUSEHOLD_FACE

返回参数

参数	类型	描述
无		

1.3 删除员工信息

Path路径	/solution/community/org/employee/delete
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述

identityId	String	是	员工ID

返回参数

参数	类型	描述
无		

1.4 分页查询员工

Path路径	/solution/community/org/employee/query
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
pageNo	Integer	是	页码，不小于0
pageSize	Integer	是	页记录大小，范围[1, 100]
name	String	否	非空时匹配名称
nameLike	String	否	非空时模糊查询
mobile	String	否	非空时匹配手机号
orderBy	JSONArray	否	排序策略，默认按创建时间倒序
column	String	是	排序列
order	String	否	DESC 倒序，ASC正序

返回参数

参数	类型	描述
total	Integer	消息总数
pageNo	Integer	页码
pageSize	Integer	页面条数
data	JSONArray	数据列表
identityId	String	员工的用户ID
name	String	姓名
mobile	String	手机号

faceImgUrl	String	智能识别图片URL
faceId	String	智能识别ID，这个字段会在上传智能识别图片后自动填充

请求示例：

```
{
  "pageNo": 1,
  "pageSize": 20,
  "nameLike": "张"
  "orderBy": {
    "column": "NAME",
    "order": "DESC"
  }
}
```

1.5 获取员工信息

Path路径	/solution/community/org/employee/get
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
identityId	String	是	用户ID

返回参数

参数	类型	描述
identityId	String	用户ID
name	String	姓名
mobile	String	手机号
faceImgUrl	String	智能识别图片URL
faceId	String	智能识别ID，这个字段会在上传智能识别图片后自动填充

1.6 添加员工到部门

Path路径	/solution/community/department/employee/add
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述

identityId	String	是	手机号
depId	String	是	部门ID

返回参数

参数	类型	描述
无		

1.7 从部门移除员工

Path路径	/solution/community/department/employee/remove
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
identityId	String	是	手机号
depId	String	是	组织节点

返回参数

参数	类型	描述
无		

1.8 分页查询部门的员工

Path路径	/solution/community/department/employee/query
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
depId	String	是	部门ID
pageNo	Integer	是	从返回结果中的第几页开始显示，不小于1
pageSize	Integer	是	返回结果中每页显示的记录数量，范围[1, 100]

返回参数

参数	类型	描述

total	Integer	总数量
pageNo	Integer	同请求参数
pageSize	Integer	同请求参数
data	JSONArray<Object>	列表
identityId	String	用户ID
mobile	String	手机号
name	String	名称
faceImgUrl	String	智能识别图片地址
faceId	String	智能识别ID

1.9 查询员工的部门

Path路径	/solution/community/employee/department/list
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
identityId	String	是	员工ID

返回参数

数据部data是JSONArray类型

参数	类型	描述
data	JSONArray<Object>	列表
depId	String	部门ID
name	String	部门名称
desc	String	说明
parentId	String	父部门ID
pathIds	JSONArray<String>	路径ID列表（包含当前部门）
pathNames	JSONArray<String>	路径名称列表（包含当前部门）

2.7.12. 物业组织管理

1.1 创建组织节点

Path路径	/solution/community/org/department/create
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
parentDepId	String	否	公司根节点时为空
name	String	是	节点名，64长
desc	String	否	描述，最长256

返回参数

参数	类型	描述
data	String	部门ID

1.2 修改组织节点

Path路径	/solution/community/org/department/modify
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
depId	String	是	组织节点ID
name	String	是	节点名，64长
desc	String	否	描述，最长256

返回参数

参数	类型	描述
无		

1.3 删除组织节点

Path路径	/solution/community/org/department/delete
请求方式	post

接口版本	1.0.0

请求参数

参数	类型	必填	描述
depId	String	是	组织节点ID

返回参数

参数	类型	描述
无		

1.4 查询子组织节点

Path路径	/solution/community/org/department/query
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
parentDepId	String	否	组织节点，查询根节点的子节点时为空
name	String	否	组织名称，精确匹配
pageNo	Integer	是	从返回结果中的第几页开始显示，默认值 1。
pageSize	Integer	是	返回结果中每页显示的记录数量，最大取值100，默认值是20。

返回参数

参数	类型	描述
total	Integer	总数量
pageNo	Integer	同请求参数
pageSize	Integer	同请求参数
list	JSONArray<Object>	列表
depId	String	节点ID
name	String	节点名
desc	String	描述

2.7.13. 设备管理

注意：设备请先在应用服务平台添加到项目，再添加到小区

1.1 添加设备到小区

path	版本	描述
/solution/community/paas/device/add	1.0.2	添加设备到小区

请求参数

入参名称	数据类型	是否必须	入参描述
deviceIdentity	JsonObject	是	详见DeviceIdentityDTO
nickName	String	否	设备别名

DeviceIdentityDTO

字段名	类型	必填	备注
type	String	是	IOT_ID/PK_AND_DN
iotId	String	否	设备id
productKey	String	否	产品PK
deviceName	String	否	产品DN

返回结果使用通用结果类型，不使用data域。

1.2 从小区删除设备

path	版本	描述
/solution/community/paas/device/remove	1.0.1	从小区删除设备

请求参数

入参名称	数据类型	是否必须	入参描述
deviceIdentity	JsonObject	是	详见DeviceIdentityDTO

返回结果使用通用结果类型，不使用data域。

1.3 小区设备列表

path	版本	描述
/solution/community/paas/device/query	1.0.2	小区设备列表

请求参数

入参名称	数据类型	是否必须	入参描述
pageNo	整型	是	当前页，从1开始

pageSize	整型	是	每页大小，最大100
categoryKey	字符串	否	设备的品类。例如梯控设备的品类是ElevatorController

返回结果

字段名	类型	备注
data	JSON	JsonArray 详见DeviceDTO

1.4 租户设备列表

path	版本	描述
/solution/community/paas/tenant/device/query	1.0.1	租户设备列表

请求参数

入参名称	数据类型	是否必须	入参描述
queryType	String	是	查询类型OWNED本租户设备，AUTHORIZED 授权设备
pageNo	整型	是	当前页，从1开始
pageSize	整型	是	每页大小，最大100

返回结果

字段名	类型	备注
data	JSON	JsonArray 详见DeviceDTO

DeviceDTO

参数名称	数据类型	参数描述
iotId	String	设备Id
productImage	String	设备的产品图片
deviceName	String	设备名称
nickName	String	设备昵称
productKey	String	产品PK
productName	String	产品名称
nodeType	String	0:DEVICE 1:GATEWAY
status	Integer	0 离线 1在线 -1未激活
categoryName	String	设备所属品类名称

categoryKey	String	设备所属品类key
thingType	String	设备的类型：VIRTUAL、WEB、APP、DEVICE
lastOnlineTime	String	设备最后上线时间

1.5 查询推流地址

获取摄像头实时视频流服务

该接口不需要cloudtoken

path	版本	描述
/vision/stream/query	2.1.0	查询推流地址

请求参数

入参名称	数据类型	是否必须	入参描述
iotId	字符串	是	设备id
streamType	整型	否	流类型，0表示主码流，1表示辅码流
encrypted	布尔	否	是否加密
encryptType	整型	否	加密类型，0表示关键帧加密
forceIFrame	布尔	否	强制I帧
scheme	字符串	否	协议类型，支持 rtmp/hls/flv

返回结果

参数名称	数据类型	参数描述
data	JSON	视频流

视频流结构

参数名称	数据类型	参数描述
path	字符串	直播地址
decryptKey	JSON	解密密钥
iv	字符串	向量base64值
key	字符串	密钥base64值

1.6 获取录像记录

获取NVR上的录像记录，用于1.11.4播放

该接口不需要cloudtoken

path	版本	描述

/vision/record/query	2.0.0	获取录像记录

请求参数

入参名称	数据类型	是否必须	入参描述
iotId	字符串	是	设备ID
streamType	整型	否	码流类型，0表示主码流，1表示辅码流
beginTime	整型	是	查询开始时间，1970年1月1日0点开始的秒数
endTime	整型	是	查询结束时间，1970年1月1日0点开始的秒数
recordType	整型	否	录像类型，0表示计划录像，1表示报警录像，2表示主动录像，3表示边看边录，99表示所有录像
pageNo	整型	否	起始页数，从0开始
pageSize	整型	否	分页大小
needSnapshot	布尔	否	是否需要录像缩略图

返回结果

参数名称	数据类型	参数描述
data	JSON	录像结果

录像结果

参数名称	数据类型	参数描述
recordFileList	JSON	录像文件列表
beginTime	字符串	文件开始时间，格式yyyy-MM-dd HH:mm:ss
endTime	字符串	文件结束时间，格式yyyy-MM-dd HH:mm:ss
recordType	整型	录像类型
streamType	整型	码流类型
fileName	字符串	文件名
fileSize	整型	文件大小
snapshotUrl	字符串	录像缩略图URL
nextValid	布尔	下一页是否有效

nextBeginTime	整型	下一页起始时间

1.7 获取历史文件播放地址

获取录像记录的播放地址

该接口不需要cloudtoken

path	版本	描述
/vision/vod/getbyfilename	2.0.0	获取录像记录播放地址

请求参数

入参名称	数据类型	是否必须	入参描述
iotId	字符串	是	设备ID
fileName	字符串	是	文件名，录像查询接口返回

返回结果

参数名称	数据类型	参数描述
data	JSON	响应结果，结果请查看下一列
vodUrl	字符串	点播地址，带鉴权信息，无法重复点播使用

1.8 摄像头信息维护

录入接口

https://help.aliyun.com/document_detail/140317.html?spm=a2c4g.11174283.6.656.3eb152d0h4ulQY

API名称	API描述	API Path	API 版本
新增/查询数据	基于已经创建且被授权写入的模型，进行数据的新增。	/data/model/data/insert /data/model/data/query	0.0.3

模型ID	版本
·VideoDeviceInfo	1.1

属性名称	属性标识符	数据类型	数据描述	是否必须
视频设备Id	deviceId	String	视频设备Id	是
设备别名	name	String	设备别名	是
品牌	brand	String	品牌	否
产品型号	productModel	String	产品型号	否
分辨率	resolution	String	分辨率	否
IP地址	ip	String	IP地址	否

属性名称	属性标识符	数据类型	数据描述	是否必须
通道号	channel	String	通道号	否
分组名称	groupName	String	分组名称	否
播放地址	url	String	播放地址	否
状态	status	Enum	状态	否
位置地址	address	String	位置地址	否
位置经度	longitude	Double	位置经度-高德坐标系	否
位置经度	latitude	Double	位置经度-高德坐标系	否
备注	remark	String	备注	否

1.9 SI项目下绑定设备

该接口不需要cloudtoken

path	版本	描述
/project/device/bind	1.0.0	绑定项目

请求参数

入参名称	数据类型	是否必须	入参描述
projectId	String	是	SI中的项目id
iotIds	Array	是	设备的iotid列表，["TEHlmHRWktnr0QA1zwOk000000","lAAkfniNdZWECLttFN1h000000"]

返回结果

参数名称	数据类型	参数描述
data	JSON	无

1.10 SI项目下解绑设备

该接口不需要cloudtoken

path	版本	描述
/project/device/unbind	1.0.0	解绑项目

请求参数

入参名称	数据类型	是否必须	入参描述
projectId	String	是	SI中的项目id

入参名称	数据类型	是否必须	入参描述
iotIds	Array	是	设备的iotid列表，["TEHlmHRWktnr0QA1zwOk000000","lAAkfniNdZWECLttFN1h000000"]

返回结果

参数名称	数据类型	参数描述
data	JSON	无

1.11 修改设备别名

path	版本	描述
/solution/community/device/nickname/update	1.0.0	修改设备别名

请求参数

入参名称	数据类型	是否必须	入参描述
deviceIdentity	JsonObject	是	详见DeviceIdentityDTO
nickName	String	是	设备别名

DeviceIdentityDTO

字段名	类型	必填	备注
type	String	是	IOT_ID/PK_AND_DN
iotId	String	否	设备id
productKey	String	否	产品PK
deviceName	String	否	产品DN

返回结果使用通用结果类型，不使用data域。

2.7.14. 门禁管理服务

1.1 门禁智能通行记录

模型ID	版本
iotx_face_record_model	1.6

属性列表

属性名称	属性标识符	数据类型	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，
设备id	iotId	String	设备id
用户信息扩展信息	userExtInfo	String	用户扩展信息
设备扩展信息	deviceExtInfo	String	设备扩展信息
通行方式	mediaType	String	开门方式，例如face
抓拍图片	pictureUrl	String	抓拍图片URL

1.2 门禁传统通行记录

模型ID	版本
iot_entrance_event_model	1.1

属性列表

属性名称	属性标识符	数据类型	数据描述	必须
用户id	identityId	String	用户id	是
用户姓名	name	String	用户姓名	是
事件时间	time	Date	事件时间	是
设备id	iotId	String	设备id	是
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	否
门禁卡号	cardId	String	门禁卡号，只有type是CARD的时候有效	否
通行方向	direction	String	通行方向，ENTRANCE 进，EXIT 出	否
用户扩展信息	userExtInfo	String	用户扩展信息	否
设备扩展信息	devExtInfo	String	设备扩展信息	否

1.3 生成二维码

生成访客二维码，首先需要在用户组中添加访客组，并且在组内添加相应的人员信息获取到identityId。

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

请求参数

参数	类型	是否必填	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16

返回结果

字段名	类型	备注
data	String	二维码内容字符串

1.4 二维码权限配置

将二维码绑定到门禁设备上，系统会根据绑定关系下发到指定设备中。

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

请求参数

参数	类型	是否必填	备注
qrCode	String	是	二维码
iotIds	JSONArray	是	待配置权限的门禁设备iotId，批量最多10个
identityId	String	是	申请二维码人员的统一身份ID
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 不能超过当前时间
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

返回结果

字段名	类型	备注
data	JSON	
code	Integer	返回码
message	String	错误消息
iotId	String	设备iotId

1.5 二维码权限删除

将二维码从门禁设备中删除。

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

请求参数

参数	类型	是否必填	备注
qrCode	String	是	二维码
iotIds	JSONArray	是	待配置权限的门禁设备iotId，批量最多10个
identityId	String	是	申请二维码人员的统一身份ID

返回结果

字段名	类型	备注
data	JSON	
code	Integer	返回码
message	String	错误消息
iotId	String	设备iotId

1.6 用户权限配置

将已保存的用户图片下发到设备端，使设备有权限识别对应的用户。支持从设备维度和空间维度进行下发。

path	版本	是否需要登录
/face/paas/permission/add	1.0.0	否

请求参数

参数	类型	是否必填	备注
userType	String	是	默认填写IDENTITY
userIdList	JSONArray	是	用户ID列表，批量最多10个
scopeType	String	是	默认填写IOT_ID
scopeIdList	JSONArray	是	信息下发的设备列表，批量最多10个

返回结果

返回结果使用通用结果类型，不使用data域。

1.7 用户权限删除

将已保存的用户图片从设备端删除，使设备无权限识别对应的用户。支持从设备维度和空间维度进行删除。

path	版本	是否需要登录
/face/paas/permission/delete	1.0.0	否

请求参数

参数	类型	是否必填	备注
userType	String	是	默认填写IDENTITY
userIdList	JSONArray	是	用户ID列表，批量最多10个
scopeType	String	是	默认填写IOT_ID
scopeIdList	JSONArray	是	信息下发的设备列表，批量最多10个

返回结果

返回结果使用通用结果类型，不使用data域。

1.8 查询用户和设备的权限关系

根据用户id和用户类型查询图片信息及其下发的设备列表（含下发状态）。

path	版本	是否需要登录
/face/paas/permission/querybyuser	1.0.4	否

请求参数

参数	类型	是否必填	备注
userType	String	是	默认填写IDENTITY
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	JsonArray	否	状态列表： 详见下发状态定义

下发状态定义

状态	定义
toBeTransferred	等待下发
transferring	下发中
deviceOffline	设备离线或者设备异常
transferred	下发完成
faceCheckTimeout	下发失败，提取特征值超时

状态	定义
faceDlError	下发失败，下载图片失败
faceCheckError	下发失败，提取特征值失败或其他未知原因
facePushError	下发失败，推送到终端设备出错
unknownError	下发失败，设备端未返回结果，原因未知
transferDeleted	删除成功
deleteFailed	删除失败
transferTimeout	下发超时
deleteTimeout	删除超时

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	默认填写IDENTITY
userId	String	用户ID
userName	String	用户姓名，不超过64字符
expiredTime	String	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
iotId	String	设备Id
syncTime	String	下发时间
status	String	详见下发状态定义
deviceNickName	String	设备昵称
cron	String	有效期表达式

1.9 根据设备查询用户权限

根据用户id和用户类型查询图片信息及其下发的设备列表（含下发状态）。

path	版本	是否需要登录
/face/paas/permission/querybydevice	1.0.4	否

请求参数

参数	类型	是否必填	备注
iotId	String	是	设备iotId
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	JsonArray	否	状态列表： 详见下发状态定义

返回结果

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	Long	下发到该设备的用户图片总数
pageNo	Integer	请求页码
pageSize	Integer	请求页大小
data	JsonArray	用户列表
userType	String	默认填写IDENTITY
userId	String	用户ID
userName	String	用户姓名，不超过64字符
expiredTime	String	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
syncTime	String	图片下发时间
status	String	下发状态： 详见下发状态定义
cron	String	有效期表达式

1.10 远程开门服务

根据用户id和用户类型查询图片信息及其下发的设备列表（含下发状态）。

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

请求参数

参数	类型	是否必填	备注
iotId	String	是	门禁设备的iotId
identifier	String	是	固定值“remoteOpen”
args	JSNOObject	是	固定值{}

返回结果

返回结果使用通用结果类型，data域为空。

1.11生成加密二维码

通过1.3生成的非加密二维码进行RSA密钥加密，生成88位的加密二维码，该二维码用于展示后给设备端扫码使用

path	版本	是否需要登录
/entrance/paas/perm/qrcode/encrypt	1.0.0	否

请求参数

参数	类型	是否必填	备注
identityId	String	是	二维码对应用户id
qrCode	String	是	二维码字符串
expiryTime	String	是	二维码失效时间，格式： yyyy-MM-dd HH:mm:ss

返回结果

字段名	类型	备注
data	String	加密二维码字符串

二维码人员信息录入

二维码通行需要使用本接口将用户信息绑定到平台

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。

字段	类型	是否必传	备注
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
}
```

二维码人员删除人员信息

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
}
```

2.7.15. AI布防服务

1. AIBOX用户组布防

用户组布防，是将对应的用户组智能识别同步至边缘AIBOX中，由AIBOX进行人员比对后形成人行数据，后续用户组的用户信息进行变更时。

Path路径	/solution/community/paas/user/group/deploy
请求方式	post

接口版本	1.0.0

请求参数

字段名	类型	必填	备注
iotId	String	是	布控设备(AI-BOX)ID
userGroupId	String	是	用户组id

返回结果

通用网关消息体，无业务字段

2. AIBOX用户组撤防

用户组撤防，是将之前同步至线下的用户组智能识别进行删除操作，撤防后人行轨迹数据将不再上报。

Path路径	/solution/community/paas/user/group/undeploy
请求方式	post
接口版本	1.0.0

请求参数

字段名	类型	必填	备注
iotId	String	是	布控设备(AI-BOX)ID
userGroupId	String	是	用户组ID

返回结果

通用网关消息体，无业务字段

3. 小区布防记录查询

对当前布防记录进行查询，实时了解当前布防情况。

Path路径	/solution/community/paas/user/group/deploy/query
请求方式	post
接口版本	1.0.0

请求参数

字段名	类型	必填	备注
pageSize	Integer	否	页面大小，最大100，默认20
pageNo	Integer	否	页码，默认1
userGroupId	String	否	用户组ID
iotId	String	否	布控设备ID

返回结果

字段名	类型	备注

data	JSONArray	布控信息

布防信息

字段名	类型	备注
deployTime	Long	最新部署时间
userGroupId	String	用户组id
userGroupName	String	组名称
userGroupType	String	用户组类型
iotId	String	布控设备ID
deployStatus	String	布控结果
schedule	Integer	布控进度 0-100

4. 边缘AI事件

模型ID	版本
IntelligentEventData	1.0

属性列表

字段	类型	备注	是否必填
eventCode	String	事件代码	是
pictureFileKey	String	事件图片	是
rect	String	检测	否
pictureContent	Integer	图片分析内容	否，人员聚集事件时反馈人数，区域入侵时反馈： 0-- person 1-- car 2-- bus 3-- truck 4-- motorcycle
iotId	String	设备ID	是
eventTime	Long	事件抓拍时间	是

事件Code

字段	类型	备注	是否必填
EventCode	String	事件代码	是

EventName	String	事件名称	是
EventDescription	String	事件描述	否

事件定义

EventName	EventCode	EventDescription
CrowdGathered	11012	人员聚集
RegionalInvasion	11001	区域入侵
FireChannelOccupancy	13002	消防通道占用
illegalParking	10001	违章停车
GarbageExposure	12002	垃圾满溢
AbsentDetect	11004	离岗检测
KeyAreaOccupation	10012	重点区域占用
CameraObscureDetect	10010	摄像头遮挡检测

5. 人行接口详情

5.1 人行轨迹检索

人行轨迹检索能力是通过用户faceId在人行数据集中进行数据检索，用户可以通过返回的轨迹结构化数据，进行轨迹视图的拼装，由于所有设备的经纬度都采用的是高德坐标系，建议使用高德的轨迹还原服务，进行目前轨迹数据如果分页返回可能会导致轨迹绘制问题，目前是会采取全量返回的策略，最大10w条，默认返回最近的数据。

Path路径	/home/face/user/trace/query
请求方式	post
接口版本	1.0.0

请求参数

字段名	类型	必填	备注
startTM	Long	是	开始时间，毫秒
endTM	Long	是	结束时间，毫秒
faceId	String	是	智能识别id，可以是系统同录入智能识别的faceId，也可以是陌生人的faceId

返回结果

字段名	类型	备注
code	Integer	200-成功
message	String	success

localizedMsg	String	
data	JSON	
faceId	String	智能识别ID
timeSeries	String	样例："{\"1578648767325\":\"1\", // 时间戳： 设备序号 \"1578648767953\":\"1\",\"1578648768029\":\"1\"}"
positionMap	String	样例："{\"1\": // 设备序号 {\"iotId\":\"rDLZdBN1QsERXtr8biXX000100\", //设备id\"latitude\":\"30.132176\", // 设备经度 \"longitude\":\"120.080502\" // 设备纬度}}"

5.2 人行数据订阅

实时订阅人行数据，用于业务层做流程组装。

模型ID	版本
· CustomerFlowBasicFaceData	1.1

人行数据

字段名	类型	备注
scope_id	String	隔离Id，如项目Id
iotId	String	摄像头iotId
productKey	String	设备pk
deviceName	String	设备dn
blur_score	Double	智能识别模糊度得分
pose_score	Double	智能识别姿态得分
age	Integer	年龄
gender	String	male/female/unknown
image	String	智能识别图文件名称
landmarks	String	[[0.1, 0.1], [0.1, 0.1],[0.1, 0.1],[0.1, 0.1],[0.1, 0.1]]
groupId	String	非必填，人员分组ID
faceId	String	识别人员的faceId
faceIdEffectiveTime	Date	faceId有效时间
recordTime	Date	记录时间unix timestamp

2.7.16. 人行管理服务

1.1 人行轨迹检索

人行轨迹检索能力是通过用户faceId在人行数据集中进行数据检索，用户可以通过返回的轨迹结构化数据，进行轨迹视图的拼装，由于所有设备的经纬度都采用的是高德坐标系，建议使用高德的轨迹还原服务，进行目前轨迹数据如果分页返回可能会导致轨迹绘制问题，目前是会采取全量返回的策略，最大10w条，默认返回最近的数据。

Path路径	/home/face/user/trace/query
请求方式	post
接口版本	1.0.0

请求参数

字段名	类型	必填	备注
startTM	Long	是	开始时间，毫秒
endTM	Long	是	结束时间，毫秒
faceId	String	是	智能识别id，可以是系统同录入智能识别的faceId，也可以是陌生人的faceId

返回结果

字段名	类型	备注
code	Integer	200-成功
message	String	success
localizedMsg	String	
data	JSON	
faceId	String	智能识别ID
timeSeries	String	样例： "{ \"1578648767325\":\"1\", // 时间戳：设备序号 \"1578648767953\":\"1\", \"1578648768029\":\"1\" }"

positionMap	String	样例：
		"{
		\"1\": // 设备序号
		{
		\"iotId\":\"rDLZdBN1QsERXtr8biXX000100\", //设备id
		\"latitude\":\"30.132176\", // 设备经度
		\"longitude\":\"120.080502\" // 设备纬度
		}
		}"

1.2 人行数据订阅

实时订阅人行数据，用于业务层做流程组装。

模型ID	版本
· CustomerFlowBasicFaceData	1.1

人行数据

字段名	类型	备注
scope_id	String	隔离Id，如项目Id
iotId	String	摄像头iotId
productKey	String	设备pk
deviceName	String	设备dn
blur_score	Double	智能识别模糊度得分
pose_score	Double	智能识别姿态得分
age	Integer	年龄
gender	String	male/female/unknown
image	String	智能识别图文件名称
landmarks	String	[[0.1, 0.1], [0.1, 0.1],[0.1, 0.1],[0.1, 0.1],[0.1, 0.1]]
groupId	String	非必填，人员分组ID
faceId	String	识别人员的faceId
faceIdEffectiveTime	Date	faceId有效时间
recordTime	Date	记录时间unix timestamp

2.7.17. 车行管理服务

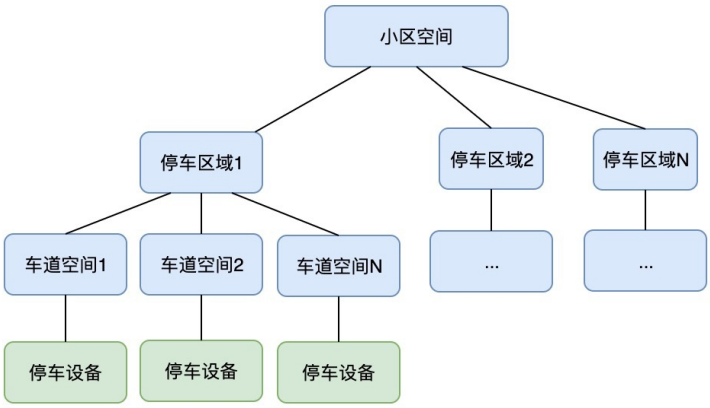
1 车行空间与设备

车行空间是对小区空间下的车辆出入口、停放区域的空间抽象，主要用于车辆权限分发，以及停车设备管理。

数字社区车行空间包括：

- 停车区域（parking_area）：一个独立的停车场空间，有单独的进出口通道；
- 车道空间（parking_space_channel）：车辆进出口空间，每个车道绑定一个停车设备。

一个小区可以划分多个停车区域，一个停车区域可以包含多个车道，每个车道绑定一个停车设备，如下图所示：



2 车辆通行记录

实时订阅车行数据，用于业务层做流程组装。

模型ID	版本
iot_park_record_v2_model	1.6

车行数据

字段名	类型	备注
direction	Integer	车辆通行的方向：0：进场，1：出场
openType	Integer	抬杆类型：0：正常抬杆，1：异常抬杆，2：手动抬杆
typePermission	Integer	车辆的进出权限类型：0：普通车辆，1：临时车辆，2：白名单，3：黑名单，4：无牌车
tollProductKey	String	车辆道闸设备productKey
tollDeviceName	String	车辆道闸设备deviceName
plateNumber	String	车牌号
areaSpaceId	String	停车区域空间ID
iotId	String	设备iotId
plateNumberType	String	机动车号牌种类代码：01-大型汽车号牌；02-小型汽车号牌；03-使馆汽车号牌；04-领馆汽车号牌；05-境外汽车号牌；06-外籍汽车号牌；07-普通摩托车号牌；08轻便摩托车号牌；09-使馆摩托车号牌；51-大型新能源车牌汽车； 52-小型新能源车牌车；99-其他
carColorNumber	String	机动车颜色代码：A-白；B-灰；C-黄；D-粉；E-红；F-紫；G-绿；H-蓝；I-棕；J-黑；K-橙；L-青；M-银；N-藏青；Z-其他
hasPlate	Integer	有无车牌：0：无车牌，1：有车牌
plateColorNumber	Integer	机动车号牌颜色：0-白色、1-黄色、2-蓝色、3-黑色，4-其他，5-绿色、6-红色、7-黄绿双色， 8-渐变绿色，9-未识别

plateRecognitionReliability	Double	车牌识别可信度：0~1浮点数
carLogoBrand	String	车辆品牌标志
carEigenvaluesInfo	String	车辆特征值信息，JSON字符串
carExtraInfo	String	车辆扩展信息，JSON字符串
userExtraInfo	String	用户扩展信息，JSON字符串
deviceExtraInfo	String	设备扩展信息，JSON字符串
carImageKey	String	车辆全景图像ID
channelSpaceId	String	车道空间ID（出入口）

3 车行空间管理

3.1 新建车行空间

见“新建小区下区域”接口：

- 新建停车区域：选择空间类型代码“parking_area”；
- 新建车道空间：选择空间类型代码“parking_space_channel”，并指定车道方向（direction）。

3.2 修改车行空间

见“修改小区下区域”接口。

3.3 删除车行空间

见“删除小区下区域”接口。

3.4 查询车行空间

见“层级查询子空间”接口。

见“平铺查询子空间”接口

见“查询空间详情”接口。

4 车辆权限管理

4.1 添加（更新）车辆权限

Path路径	/parking/paas/vehicle/permission/add
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
permList	JSONArray	是	
- spaceId	String	是	空间ID
- plateNumber	String	是	车牌号
- effectiveDate	String	是	车辆有效期的开始时间：yyyy-MM-dd HH:mm:ss

- expiryDate	String	是	车辆有效期的终止时间：yyyy-MM-dd HH:mm:ss

返回参数

参数	类型	描述
无		

4.2 查询车辆权限

Path路径	/parking/paas/vehicle/permission/query
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
spaceIds	List<String>	是	空间ID
plateNumbers	List<String>	否	车牌号
status	String	否	权限同步状态： • toBeTransferred 等待同步 • transferring 同步中 • transferred 同步成功 • deviceOffline 设备离线 • transferTimeout 同步超时 • unknownError 同步失败
pageNo	Integer	是	页码
pageSize	Integer	是	页面大小

返回参数

参数	类型	描述
pageNo	Integer	页码
pageSize	Integer	页面大小
total	Integer	总数
data	JSONArray	权限列表
- plateNumber	String	车牌号
- effectiveDate	String	车辆有效期的开始时间：yyyy-MM-dd HH:mm:ss
- expiryDate	String	车辆有效期的终止时间：yyyy-MM-dd HH:mm:ss

- spaceId	String	空间ID
- spaceName	String	空间名称
- status	String	权限同步状态： • toBeTransferred 等待同步 • transferring 同步中 • transferred 同步成功 • deviceOffline 设备离线 • transferTimeout 同步超时 • unknownError 同步失败

4.3 删除车辆权限

Path路径	/parking/paas/vehicle/permission/delete
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
permList	JSONArray	是	
-spaceId	String	是	空间ID
-plateNumber	String	是	车牌号

返回参数

参数	类型	描述
无		

5 黑白名单管理

5.1 添加（更新）黑白名单

Path路径	/parking/paas/vehicle/special/concerns/add
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
permList	JSONArray	是	
- spaceId	String	是	空间ID
- plateNumber	String	是	车牌号

- type	String	是	车辆类型： • WHITELIST 白名单 • BLACKLIST 黑名单
- brand	String	否	车辆品牌
- color	Integer	否	车身颜色： 0-其他 1-白 2-灰 3-黄 4-粉 5-红 6-紫 7-绿 8-蓝 9-棕 10-黑 11-青

返回参数

参数	类型	描述
无		

5.2 删除黑白名单

Path路径	/parking/paas/vehicle/special/concerns/delete
请求方式	post
接口版本	1.0.1

请求参数

参数	类型	必填	描述
permList	JSONArray	是	
- spaceId	String	是	空间ID
- plateNumber	String	是	车牌号

返回参数

参数	类型	描述
无		

5.3 查询黑白名单

Path路径	/parking/paas/vehicle/special/concerns/query
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
spaceIds	List<String>	是	空间ID
plateNumbers	List<String>	否	车牌号

type	String	是	车辆类型： • WHITELIST 白名单 • BLACKLIST 黑名单
pageNo	Integer	是	页码
pageSize	Integer	是	页面大小

返回参数

参数	类型	描述
pageNo	Integer	页码
pageSize	Integer	页面大小
total	Integer	总数
data	JSONArray	权限列表
- plateNumber	String	车牌号
- brand	String	车辆品牌
- color	Integer	车身颜色： 0-其他 1-白 2-灰 3-黄 4-粉 5-红 6-紫 7-绿 8-蓝 9-棕 10-黑 11-青
- spaceId	String	空间ID
- status	String	黑白名单同步状态： • toBeTransferred 等待同步 • transferring 同步中 • transferred 同步成功 • deviceOffline 设备离线 • transferTimeout 同步超时 • unknownError 同步失败

6 停车设备管理

6.1 绑定停车设备

Path路径	/parking/paas/toll/bind
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
iotId	String	是	设备唯一标识
spaceId	String	是	车道空间ID

返回参数

参数	类型	描述
无		

6.2 解绑停车设备

Path路径	/parking/paas/toll/unbind
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
iotId	String	是	设备唯一标识

返回参数

参数	类型	描述
无		

6.3 查询停车设备

Path路径	/parking/paas/toll/query
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
spaceId	String	是	车道空间ID
getDeviceStatus	Boolean	否	是否查询设备状态： • true 查询设备状态 • false 不查询设备状态（默认）

返回参数

参数	类型	描述
iotId	String	设备iotId
name	String	设备名称
productKey	String	设备productKey

status	Integer	设备状态： • 0 未激活 • 1 在线 • 3 离线 • 8 禁用

7 远程开关道闸

见“触发物的服务”接口。

请求参数

参数	类型	必填	备注
iotId	String	是	设备唯一标识
args	map	是	{"OperatType": 1} • 0 关闸 • 1 开闸 • 2 长开
identifier	String	是	固定值：OperatGate

返回结果

字段名	类型	备注
无	无	无

8 车道视频直播

见“查询推流地址”接口。

9 LED内容下发

9.1 LED内容下发模式

见“获取物的属性”接口，属性标识为“LedNoticeMode”，支持以下属性值：

- 0：设备支持分离下发模式，可分别下发提示语音、提示显示、默认显示；
- 1：设备仅支持混合下发提示信息，屏幕不支持单行刷新。

9.2 提示语音下发

见“触发物的服务”接口。

请求参数

参数	类型	必填	备注
iotId	String	是	设备唯一标识
args	Map	是	{"VoiceContent": "语音提示信息","PlaySpeed":1}
-VoiceContent	String	是	语音播报内容，数据长度2048
-PlaySpeed	Integer	否	语音播报速度，取值范围：1 - 10，步长：1
identifier	String	是	固定值：NoticeBroadcast

返回结果

字段名	类型	备注
无	无	无

9.3 提示显示下发

见“触发物的服务”接口。

请求参数

参数	类型	必填	备注
iotId	String	是	设备唯一标识
args	Map	是	{"DisplayContent": "提示显示信息","RowNumber":1}
-DisplayContent	String	是	显示内容，数据长度2048
-RowNumber	Integer	是	屏幕行号，取值范围：1 - 100
-MovingSpeed	Integer	否	文字移动速度，取值范围：1 - 10，步长：1
-DisplayTime	Integer	否	显示时长，取值范围：1 - 3600，单位：秒
-MovingType	Integer	否	文字移动方式： • 0 禁止（默认） • 1 从右到左 • 2 从左到右 • 3 从下到上 • 4 从上到下
identifier	String	是	固定值：NoticeDisplay

返回结果

字段名	类型	备注
无	无	无

9.4 默认显示下发

见“触发物的服务”接口。

请求参数

参数	类型	是否必填	备注
iotId	String	是	设备唯一标识
args	Map	是	{"DisplayContent": "提示显示信息", "RowNumber":1, "DefaultSituation":0}
-DisplayContent	String	是	显示内容，数据长度2048
-RowNumber	Integer	是	屏幕行号，取值范围：1 - 100

-DefaultSituation	Integer	是	默认显示场景： • 0 空闲 • 1 普通权限车辆、白名单车辆过车 • 2 黑名单车辆过车 • 3 临时车量过车 • 4 无牌车辆过车
identifier	String	是	固定值：DefaultDisplay

返回结果

字段名	类型	备注
无	无	无

9.5 LED内容混合下发

不支持“提示语音”、“提示显示”分离下发的设备，使用以下接口混合下发提示信息。

Path路径	/parking/paas/led/content/download
请求方式	post
接口版本	1.0.0

请求参数

参数	类型	必填	描述
iotId	String	是	车行摄像头的设备ID
type	Integer	是	下发场景类型： • 0 固定场景 • 1 临时场景 • 2 个性化场景
situation	Integer	否	固定场景种类： • 0 空闲（默认） • 1 普通权限车辆、白名单车辆过车 • 2 黑名单车辆过车 • 3 临时车辆过车 • 4 无牌车辆过车
showTime	Integer	否	内容持续显示时长，单位：秒，默认值：10
audioContextInfo	String	否	自定义语音播报内容
screenLineNum	Integer	是	屏幕显示行数，默认值：4
contextList	JSONArray	是	屏幕显示内容，每个元素表示一行

- runType	Integer	否	文字滚动方式： • 0 静止 （默认） • 1 从右到左 • 2 从左到右 • 3 从下到上 • 4 从上到下
- colorType	Integer	否	文字颜色： • 1 红色 • 2 绿色（默认）
- customContext	String	是	自定义内容，长度0-128。内容支持以下通配符： • #T 当前时间 • #t 车辆到期时间 • #p 车牌号码
- audio	Integer	否	是否进行语音播报： • 0 不播报（默认） • 1 播报

返回参数

参数	类型	描述
无		

10 手动触发车牌识别

见“触发物的服务”接口。

请求参数

参数	类型	是否必填	备注
iotId	String	是	设备唯一标识
args	Map	是	{}
identifier	String	是	固定值：TriggerRecognition

返回结果

字段名	类型	备注
无	无	无

11 二维码下发

11.1 是否支持二维码

见“获取物的属性”接口，属性标识为“IsSurportQrCode”，支持以下属性值：

- 0：不支持
- 1：支持

11.2 二维码下发

见“触发物的服务”接口。

请求参数

参数	类型	是否必填	备注
iotId	String	是	设备唯一标识
args	Map	是	{"QrCode":"二维码文本信息"}
identifier	String	是	固定值：QrCodeDisplay

返回结果

字段名	类型	备注
无	无	无

2.7.18. 公告服务

1.1添加消息

Path路径	/solution/community/notice/add
请求方式	Post
接口版本	1.0.3

参数	类型	必填	描述
title	String	是	公告标题。最大长度为20。只允许数字、中英文字符。
publications	JSONArray	是	发布消息中的元素分组列表。每个元素分组最少包含一条文本，可以包含一个图片。最多5组。
text	String	是	单条公告文本内容。最大长度1000。
uploadId	String	否	公告图片上传到OSS后的uploadId
type	Integer	是	通知类型：1社区公告、2账单通知、3缴费结果、4业务受理、5业务办结、6小区通知、7活动推荐、8物业服务、9便民生活
isTop	Integer	是	是否置顶：1是，0否
communityId	Long	否	小区ID
publishType	Integer	是	发送类型，1-支付宝生活号，2-支付宝小程序通知，4-天猫精灵
spaceIds	List<String>	否	目标空间ID列表

消息ID。数据类型为String。

1.2更新消息

Path路径	/solution/community/notice/update

请求方式	post
接口版本	1.0.0

参数	类型	必填	描述
messageId	Long	是	消息ID
communityId	Long	否	小区ID
isTop	Integer	是	是否置顶：1是，0否

无

1.3查询消息

消息按照添加时间倒叙排列，isTop设置为1的数据排序优于isTop设置为0的数据

Path路径	/solution/community/notice/query
请求方式	post
接口版本	1.0.9

参数	类型	必填	描述
communityId	Long	否	小区ID
pageNo	Integer	是	页码
pageSize	Integer	是	页面条数
publishType	Integer	否	按照发送类型查询，0-所有公告消息，1-支付宝生活号，2-支付宝小程序通知，4-天猫精灵
type	Integer	否	通知类型：1社区公告、2账单通知、3缴费结果、4业务受理、5业务办结、6小区通知、7活动推荐、8物业服务、9便民生活
topFirst	Integer	否	返回结果是否以发送参数“isTop”排序。1以isTop排序，0不以isTop排序。缺省为1
spaceId	String	否	空间ID

参数	类型	描述
total	Integer	消息总数
pageNo	Integer	页码
pageSize	Integer	页面条数

data	JSONArray	数据列表
messageId	Long	消息Id
title	String	消息名称
publications	JSONArray	发布的元素分组列表
text	String	消息文本内容
pictureUrl	String	消息图片url
type	Integer	通知类型：1社区公告、2账单通知、3缴费结果、4业务受理、5业务办结、6小区通知、7活动推荐、8物业服务、9便民生活
isTop	Integer	是否置顶：1是，0否
gmtCreate	Long	创建时间
publishType	Integer	发送类型，1-支付宝生活号，2-支付宝小程序通知，4-天猫精灵。支持发送类型的按位或。
publishRange	Integer	发布类型。0-到整个小区；1-到指定的空间
spaceIds	List<String>	发送的目标空间ID列表。publishRange=0时有效

1.4删除消息

Path路径	/solution/community/notice/delete
请求方式	post
接口版本	1.0.0

参数	类型	必填	描述
messageIds	List<Integer>	是	消息ID列表
communityId	Long	否	小区ID

无

1.5获取消息详情

Path路径	/solution/community/notice/get
请求方式	post
接口版本	1.0.6

参数	类型	必填	描述
communityId	Long	否	小区ID

messageId	Long	是	小区公告ID

参数	类型	描述
messageId	Long	消息Id
title	String	消息名称
publications	JSONArray	发布的元素分组列表
text	String	消息文本内容
pictureUrl	String	消息图片url
type	Integer	通知类型：1社区公告、2账单通知、3缴费结果、4业务受理、5业务办结、6小区通知、7活动推荐、8物业服务、9便民生活
isTop	Integer	是否置顶：1是，0否
gmtCreate	Long	创建时间
publishRange	Integer	发布类型。0-到整个小区；1-到指定的空间
spaceIds	JSONArray	发送的目标空间ID列表。publishRange=0时有效

2.7.19. 变更推送及订阅

1.变更推送订阅

推送topic定义

各类数据变更使用不同的topic进行定义，便于ISV订阅不同的业务消息，以及做权限控制。/sys/community/${appkey}/${biz}/change，其中${appkey}需要替换成你的Appkey，${biz}的定义如下

业务类型

数据类型	topic	说明
空间	/sys/appkey/${appkey}/community/space/change	当空间信息发生变化时推送
公告	/sys/appkey/${appkey}/community/message/change	当小区公告有新增发布时推送
用户	/sys/appkey/${appkey}/community/user/change	当新增用户，修改用户时推送

格式示例

```
{
  "communityId":123,
  "ids":["XXXXXXX"],
  "type":"insert/delete/update/publish"
}
```

使用限制

只能订阅到Appkey有权限的小区数据，有权限的前提为，小区是使用Appkey创建的，或者小区购买了Appkey对应的应用

订阅示例

```
String appKey = "your_appkey";
String appSecret = "your_appkey_secrect";

Profile profile = Profile.getAppKeyProfile(${httpsUrl}, appKey, appSecret);

MessageClient client = MessageClientFactory.messageClient(profile);
client.connect(messageToken -> {
    System.out.println(messageToken.getMessage());
    return MessageCallback.Action.CommitSuccess;
});
```

SDK引入

```
<dependency>
    <groupId>com.aliyun.openservices</groupId>
    <artifactId>iot-client-message</artifactId>
    <version>1.1.3</version>
</dependency>

<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>3.7.1</version>
</dependency>
```

2.DOP数据订阅

前置条件

1.产生事件的设备已经添加到小区，并添加到了应用服务平台

2.订阅数据的Appkey，必须是申请小区的Appkey

使用限制

1.当存在多台机器使用同一个Appkey订阅时，推送服务将以随机的方式推送到订阅的机器上。改方式有利于集群模式部署和订阅。但如果有本地调试的机器也使用同样Appkey订阅，会导致数据被本地调试机器订阅走，请一定注意订阅的环境。

数据模型订阅接口及代码请参考

https://help.aliyun.com/document_detail/140317.html?spm=a2c4g.11174283.6.654.3eb152d0LWfvKO

3.设备数据订阅

前置条件

1.产生事件的设备已经添加到小区，并添加到了应用服务平台

使用限制

1.当存在多台机器使用同一个Appkey订阅时，推送服务将以随机的方式推送到订阅的机器上。改方式有利于集群模式部署和订阅。但如果有本地调试的机器也使用同样Appkey订阅，会导致数据被本地调试机器订阅走，请一定注意订阅的环境。

设备数据订阅代码请参考

https://help.aliyun.com/document_detail/145112.html?spm=a2c4g.11186623.6.656.1a8c907fuDGWJx

2.7.20. 可视对讲服务

1. 服务器管理

1.1 查询服务器列表

接口定义

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

入参

无

出参

字段	类型	备注
code	Integer	返回码，200为成功

message	String	返回信息
data	JSONArray	
-> serverId	String	服务器id
-> hostIp	String	服务器地址

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": {
    [
      {
        "serverId": "xxx",
        "hostIp": "1.1.1.1"
      },
      {
        "serverId": "xxx",
        "hostIp": "1.1.1.2"
      }
    ]
}
```

1.2 绑定服务器

接口定义

path	版本	是否需要登录
/entrance/paas/sipserver/bind	1.0.0	否

入参

字段	类型	是否必传	备注
serverId	String	是	服务器id

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息

请求示例

```
{
  "serverId": "xxx"
}
```

返回示例

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

1.3 查询当前绑定SIP服务器

接口定义

path	版本	是否需要登录
/entrance/paas/sipserver/getbind	1.0.0	否

入参

无

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息
data	JSONObject	
-> serverId	String	服务器id
-> hostIp	String	服务器地址

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": {
      {
        "serverId": "xxx",
        "hostIp": "1.1.1.1"
      }
  }
}
```

2. 对讲服务

2.1 创建SIP账号

需要指定type，1为门禁机类型，用于门禁机，当一路APP正在调看门禁机时，若发起第二路APP调看，则会加入会话，可以实现多个APP同时调看同一台门禁机；0为APP类型，用于APP，当SIP号在通话时，则无法加入。

接口定义

path	版本	是否需要登录
/entrance/paas/sipservice/sip/add	1.0.0	否

入参

字段	类型	是否必传	备注
count	Integer	是	创建数量，最大100
type	Integer	是	类型，0 - APP， 1 - 门禁机

出参

字段	类型	备注
code	Integer	返回码，200为成功

message	String	返回信息
data	JSONArray	数据列表
->sipNumber	String	SIP号，长度不超过50个字符
->sipPassword	String	SIP密码，长度不超过50个字符
->sipHost	String	SIP服务器地址
->sipPort	Integer	SIP服务器端口
->type	Integer	类型，0 - APP， 1 - 门禁机
->createTime	Long	创建时间戳（单位毫秒）

请求示例

```
{
  "count": 2,
  "type": 1
}
```

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": [
    {
      "sipNumber": "obog97v34g",
      "sipPassword": "o3yira",
      "sipHost": "sip.iot.aliyun.com",
      "sipPort": 6050,
      "type": 1,
      "createTime":1577085030000
    }
  ]
}
```

2.2 删除SIP账号

需要指定type，1为门禁机类型，用于门禁机，当一路APP正在调看门禁机时，若发起第二路APP调看，则会加入会话，可以实现多个APP同时调看同一台门禁机；0为APP类型，用于APP，当SIP号在通话时，则无法加入。

接口定义

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

入参

字段	类型	是否必传	备注
sipNumberList	JASONArray<String>	是	待删除SIP号列表

出参

字段	类型	备注
code	Integer	返回码，200为成功

message	String	返回信息

请求示例

```
{
  "sipNumberList": [
    "obog97v34g",
    "vbog97v34g"
  ]
}
```

返回示例

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

2.3 查询SIP账号详情

若返回错误为SYS_0002，表示SIP号无效或不存在。

接口定义

path	版本	是否需要登录
/entrance/paas/sipservice/sip/detail/get	1.0.0	否

入参

字段	类型	是否必传	备注
sipNumber	String	是	SIP号

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息
data	JSONObject	数据信息
->sipNumber	String	SIP号，长度不超过50个字符
->sipPassword	String	SIP密码，长度不超过50个字符
->sipHost	String	SIP服务器地址
->sipPort	Integer	SIP服务器端口
->type	Integer	类型，0 - APP， 1 - 门禁机
->createTime	Long	创建时间戳（单位毫秒）

请求示例

```
{
  "sipNumber": "obog97v34g"
}
```

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": {
    "sipNumber": "obog97v34g",
    "sipPassword": "o3yira",
    "sipHost": "sip.iot.aliyun.com",
    "sipPort": 6050,
    "type": 1,
    "createTime":1577085030000
  }
}
```

2.4 更新SIP账号密码

目前一个SIP账号同时只允许在一个客户端登录，因此针对可能在多端同时登录的SIP账号，请在登录SIP账号前，调用该接口更新密码，并返回新密码，则之前的旧密码失效。

接口定义

path	版本	是否需要登录
/entrance/paas/sipservice/sip/password/update	1.0.0	否

入参

字段	类型	是否必传	备注
sipNumber	String	是	SIP号

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息
data	JSONObject	数据信息
->sipPassword	String	SIP密码，长度不超过50个字符

请求示例

```
{
  "sipNumber": "obog97v34g"
}
```

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": {
    "sipPassword": "12345678"
  }
}
```

2.5 创建SIP组

接口定义

path	版本	是否需要登录

/entrance/paas/sipservice/group/add	1.0.0	否

入参

字段	类型	是否必传	备注
count	Integer	是	创建数量，最大100

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息
data	JSONArray	数据信息
->groupID	String	SIP组ID，长度不超过50个字符
->createTime	Long	创建时间戳（单位毫秒）

请求示例

```
{
  "count": 2
}
```

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": [
      {
      "groupID": "obog97v34g",
      "createTime":1577085030000
      },
      {
      "groupID": "abcg97v34g",
      "createTime":1577085030000
      }
  ]
}
```

2.6 删除SIP组

接口定义

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

入参

字段	类型	是否必传	备注
groupList	JASONArray<String>	是	待删除SIP组号列表

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息

请求示例

```
{
  "groupList": [
    "obog97v34g",
    "vbog97v34g"
  ]
}
```

返回示例

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

2.7 SIP账号分组

只有SIP账号type为0，即APP端使用的SIP账号，才能支持分组。

接口定义

path	版本	是否需要登录
/entrance/paas/sipservice/group/sip/add	1.0.0	否

入参

字段	类型	是否必传	备注
groupID	String	是	SIP组ID
sipNumberList	JASONArray<String>	是	SIP号列表

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息

请求示例

```
{
  "groupID": "11111",
  "sipNumberList": [
    "obog97v34g",
    "tb2yz0ca17"
  ]
}
```

返回示例

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

2.8 SIP账号从组里删除

接口定义

path	版本	是否需要登录
/entrance/paas/sipservice/group/sip/delete	1.0.0	否

入参

字段	类型	是否必传	备注
groupID	String	是	SIP组ID
sipNumberList	JASONArray<String>	是	SIP号列表

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息

请求示例

```
{
  "groupID": "11111",
  "sipNumberList": [
    "obog97v34g",
    "tb2yz0ca17"
  ]
}
```

返回示例

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

2.9 查看组下SIP账号列表

接口定义

path	版本	是否需要登录
/entrance/paas/sipservice/group/sip/list	1.0.0	否

入参

字段	类型	是否必传	备注
groupID	String	是	SIP组ID

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息

data	JASONArray	SIP账号列表
->sipNumber	String	SIP号，长度不超过50个字符
->sipPassword	String	SIP密码，长度不超过50个字符
->sipHost	String	SIP服务器地址
->sipPort	Integer	SIP服务器端口
->type	Integer	类型，0 - APP， 1 - 门禁机
->createTime	Long	创建时间戳（单位毫秒）

请求示例

```
{
  "groupID": "11111"
}
```

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": [{
      "sipNumber": "obog97v34g",
      Introduction
      10
      "sipPassword": "o3yira",
      "sipHost": "sip.iot.aliyun.com",
      "serverPort": 6050,
      "type": 0,
      "createTime":1577085030000
    },
    {
      "sipNumber": "tb2yz0ca17",
      "sipPassword": "p14lvq",
      "sipHost": "sip.iot.aliyun.com",
      "serverPort": 6050,
      "type": 0,
      "createTime":1577085030000
    }]
}
```

3. 设备与房间权限

3.1 颁发门禁机设备SIP Number

接口定义

path	版本	是否需要登录
/entrance/paas/videointercom/sipnumber/sync	1.0.0	否

入参

字段	类型	是否必传	备注
iotId	String	是	门禁对讲机iotId
sipNumber	String	是	SIP Number
password	String	是	password

host	String	是	SIP 服务器地址
port	String	是	SIP 服务器端口

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息

请求示例

```
{
  "iotId": "xxx",
  "sipNumber": "xxx",
  "host": "x.x.x.x",
  "port": "8080"
}
```

返回示例

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

3.2 查询门禁机设备SIP Number

接口定义

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

入参

字段	类型	是否必传	备注
iotId	String	是	门禁对讲机iotId

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息
data	String	SIP Number
->sipNumber	String	SIP Number
->host	String	SIP 服务器地址
->port	String	SIP 服务器端口

请求示例

```
{
  "iotId": "xxx"
}
```

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": {
    "sipNumber": "xxxx",
    "host": "x.x.x.x",
    "port": "8080"
  }
}
```

3.3 添加SIP组与房间编号映射关系

接口定义

path	版本	是否需要登录
/entrance/paas/videointercom/map/add	1.0.0	否

入参

字段	类型	是否必传	备注
iotId	String	是	门禁对讲机iotId
maps	JSONArray	是	映射关系列表
-> roomId	String	是	房间ID
-> groupId	String	是	SIP组ID

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息

请求示例

```
{
  "iotId": "xxx",
  "maps": [
    {
      "roomId": "room1",
      "groupId": "group1"
    },
    {
      "roomId": "room2",
      "groupId": "group2"
    }
  ]
}
```

返回示例

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

3.4 删除SIP组与房间编号映射关系

接口定义

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

入参

字段	类型	是否必传	备注
iotId	String	是	门禁对讲机iotId
maps	JSONArray	是	映射关系列表
-> roomId	String	是	房间ID
-> groupId	String	是	SIP组ID

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息

请求示例

```
{
  "iotId": "xxx",
  "maps": [
    {
      "roomId": "room1",
      "groupId": "group1"
    },
    {
      "roomId": "room2",
      "groupId": "group2"
    }
  ]
}
```

返回示例

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

3.5 查询SIP组与房间编号映射关系

接口定义

path	版本	是否需要登录
/entrance/paas/videointercom/map/query	1.0.1	否

入参

字段	类型	是否必传	备注
iotId	String	是	门禁对讲机iotId
roomId	String	否	房间ID
groupId	String	否	SIP组ID
pageNo	Integer	是	请求的页码，详细见“分页参数”定义。
pageSize	Integer	是	每页的记录数，详细见“分页参数”定义。

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息
data	JSONObject	
-> total	Integer	总数
-> pageNo	Integer	页码
-> pageSize	Integer	页码大小
-> data	JSONArray	映射关系列表
--> roomId	String	房间ID
--> groupId	String	SIP组ID

请求示例

```
{
  "iotId": "xxx",
  "pageNo":1,
  "pageSize":20
}
```

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": {
    "total": 2,
    "pageNo": 1,
    "pageSize": 20,
    "data": [
      {
        "roomId": "room1",
        "groupId": "group1"
      },
      {
        "roomId": "room2",
        "groupId": "group2"
      }
      ]
  }
}
```

3.6 SIP呼叫事件通知订阅

呼叫事件topic

/broadcast/{aliyunUid}/vinotify

字段	类型	是否必传	备注
eventCode	String	是	事件编号，1：呼叫事件
eventTime	Long	是	事件发生时间戳（单位毫秒）
source	String	是	事件发起方，如呼叫事件，则为门禁机SIP号
targetType	String	是	目标类型，0 - SIP号；1 - SIP组
target	String	是	事件目标方，如呼叫事件，则为组ID

样例：

```
{
  "eventCode": "xxx",
  "eventTime": 1577085030000,
  "source": "22222",
  "targetType": 1,
  "target": "11111"
}
```

数据格式数据订阅参考

https://gaic.alicdn.com/doc/homelink-aquarius2/yoqgpf.html

3.7 添加/更新房屋空间

接口定义

path	版本	是否需要登录
/entrance/paas/videointercom/space/add	1.0.0	否

入参

字段	类型	是否必传	备注
spaceId	String	是	房屋空间
groupId	String	是	SIP组

roomId	String	是	房间ID

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息

请求示例

```
{
  "spaceId": "xxx",
  "groupId": "xxx",
  "roomId": "xxx"
}
```

返回示例

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

3.8 删除房屋空间

接口定义

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

入参

字段	类型	是否必传	备注
spaceId	String	是	房屋空间

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息

请求示例

```
{
  "spaceId": "xxx"
}
```

返回示例

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

3.9 查询房屋空间

接口定义

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

入参

字段	类型	是否必传	备注
spaceId	String	否	房屋空间（与SIP组二选一）
groupId	String	否	SIP组（与房屋空间二选一）

出参

字段	类型	备注
code	Integer	返回码，200为成功
message	String	返回信息
data	JSONObject	
-spaceId	String	房屋空间
-groupId	String	SIP组
-roomId	String	房间ID

请求示例

```
{
  "spaceId": "xxx"
}
```

返回示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": {
    "spaceId": "xxx",
    "groupId": "xxx",
    "roomid": "xxx"
  }
}
```

4. 数据模型定义

模型ID：iot_entrance_videointercom_record_model

模型名称：可视对讲通行记录

模型版本：1.1

属性列表：

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设备标示	iotId	String	对讲机IoT设备标示，如果为固话对讲则为空	否

源类型	sourceType	String	0 - 云对讲	是
事件信息	eventInfo	String	类型为云对讲：{ "sipNumber": "xxx", "groupId": "xxx", "roomId": "xxx", "spaceId": "xxx"}	是

2.7.21. 应用权限管理

1.1 获取用户在每个小区的应用权限列表

path	版本	描述
/solution/community/paas/app/community/auth/list	1.0.0	获取用户在每个小区的应用权限列表

请求参数

入参名称	数据类型	是否必须	入参描述
identityId	String	是	用户ID
appKey	String	是	应用appKey

返回结果

字段名	类型	备注
code	Integer	返回码
message	String	错误消息
data	JSONArray	在各个小区的应用权限列表
communityId	Long	小区ID
actionCodes	List<String>	权限点名称列表。例如：["MANAGE", "USE"]

1.2 获取用户的应用权限列表

path	版本	描述
/solution/community/paas/app/perm/query	1.0.0	获取用户的应用权限列表

请求参数

入参名称	数据类型	是否必须	入参描述
identityId	String	是	用户ID
appId	String	是	应用appId

返回结果

字段名	类型	备注

code	Integer	返回码
message	String	错误消息
data	JSONArray	应用权限列表
actionCodes	JSONArray	权限点名称列表。
actionCode	String	权限点名称，MANAGE/USE

3.物业管理一体机

3.1. 什么是物业管理一体机

一句话介绍

物业管理一体机是面向园区、社区、校园等多种业态空间，为其解决物业管理的边缘计算产品，重点提供云边协同的能力，实现门禁、停车、安防、楼宇BA等设备标准化接入上云，并将设备和子系统能力抽象到云端，通过云端统一的开放体系进行开放，赋能合作伙伴搭建智慧通行、云对讲、设备运维管理等解决方案。

产品架构



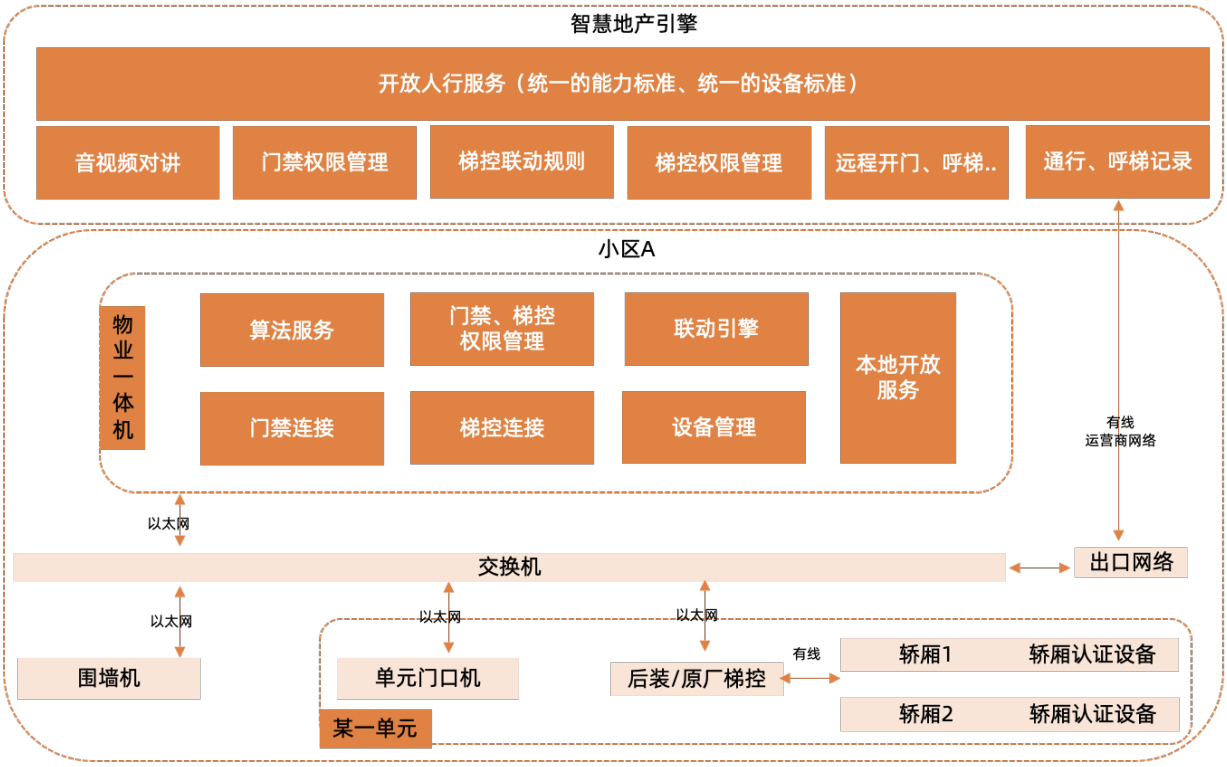
产品优势

- 场景全面，空间公区里的核心硬件全覆盖，并定义了人行场景、车行场景、EBA场景、安防场景、梯控场景的标准一体机，提供即开即用服务，通过配置来完成原有的IT和OT融合实施工作（仅限于已经完成对接的生态应用型号）
- 生态优势，产品通过生态优势，逐步丰富各场景接入的生态应用品牌和型号，已经覆盖300+设备型号，并在持续增长，减少项目对接开发成本
- 云边一体，通过AIoT的云边协同技术，将云和边进行分离计算，保证边缘离线可持续运行，同时可以发挥云在数据和服务归一的优势
- 远程运维，具备远程运维应用能力，无需跑现场，节省维护成本
- 形态开放，针对特殊场景，产品预留了多种开放能力
  - 设备接入的开放，用户可以使用物业一体机完成场景中没有覆盖的特殊设备接入，产品提供了一整套设备接入、管理、运维的服务体系
  - 边缘应用的开放，用户根据自身场景要求，在物业一体机上部署自行开发本地应用，产品提供了一整套应用开发、管理、运维的服务体系
  - 云边协同的开放，用户可以根据云边协同场景的需要，边缘应用可以通过云边协同中服务和数据的通道来完成和云上应用的协同

3.2. 功能介绍

3.2.1. 人行版一体机

产品架构



服务能力

人行版一体机可提供以下服务内容

设备接入能力

- 支持主流品牌的门禁、电梯设备、云对讲等设备接入管理，同时可以通过云端管控平台，实时下发协议、设备配置以及通行规则，可以实现人员在空间的多种通行规则，例如单元门禁自动呼梯功能。

通行服务

- 智能通行，支持智能门禁机接入和云端原始库管理，保障边缘可断网继续向本地提供智能通行服务
- 二维码通行，支持二维码门禁机接入和云端二维码管理，并通过加密二维码方式提高安全性的同时也满足断网持续运行能力，支持2W加密码管理，性能达到200次/分钟
- IC卡及蓝牙通行，支持云端IC卡卡号和蓝牙ID管理，支持2W容量IC卡号管理。
- 支持200个门禁设备接入和管理。其中人脸识别通行，支持2W人脸底库管理，性能达到200次/分钟比对，比对准确率99%以上

对讲服务

- 支持云端SIP可视对讲服务对接，对门禁和室内机或移动终端设备间，完成音视频对讲权限管理，设备间通过云对讲服务进行音视频对讲

梯控服务

支持云端呼叫电梯至指定楼层、释放楼层权限、查询电梯状态

人梯联动

提供边缘联动引擎，构建门禁机与电梯的联动规则，规则中支持人员通行后呼叫电梯至门禁所在楼层，并释放人员所有权限的楼层

硬件规格

产品型号	备注
处理器	支持Intel® Core™ i5-6400，LGA1151
内存	8GB DDR4
磁盘	1 * 128GB SSD ， 1 * HDD Bay(可选),
网口	2 * Intel i210 Gbe LAN controller

USB	4 * USB3.0/2.0/1.1, 4 * USB2.0/1.1
箱体结构	铝型材外壳，具有良好的抗震性和抗冲击性
安装方式	桌面安装，壁挂安装
散热系统	鳍状铝型材散热
电源输入	标配DC 12V输入，具有过流、过压、反接保护措施
总体尺寸	240mm × 240mm × 95mm
整机重量	3KG
工作温度	-20 ~ 60° C (-4°F~140°F) (使用宽温等级固态硬盘); 0 ~ 45° C (32°F~113°F) (使用普通温度等级机械硬盘/固态硬盘)
功耗	待机：27Watt，满载：55Watt
系统	Linux/Link Edge

云端API

- 人员账号服务API：人员账号服务API
- 传统门禁服务API：传统门禁服务API
- 人脸门禁服务API：人脸门禁服务API
- 对讲服务请查阅：可视对讲服务API
- 梯控服务API：梯控联动服务API

设备接入

- 可视对讲设备端SDK:

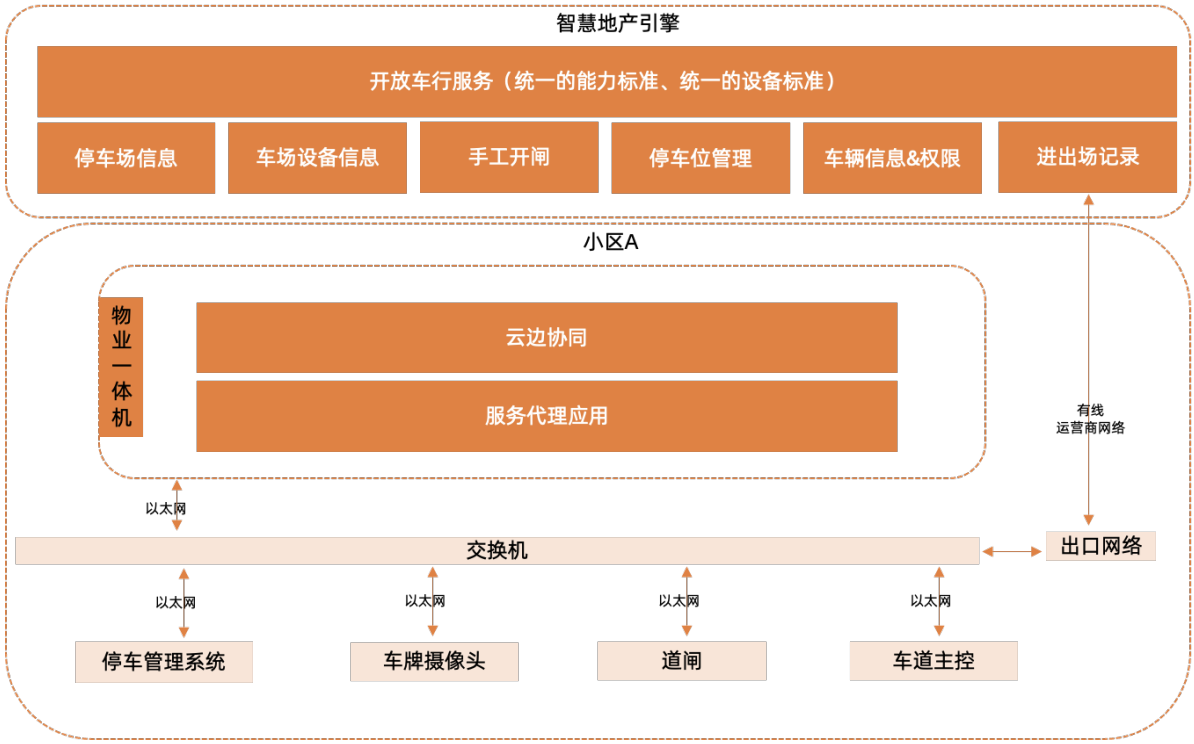
可视对讲移动端SDK接口说明-[iOS]

可视对讲移动端SDK接口说明[Andriod]

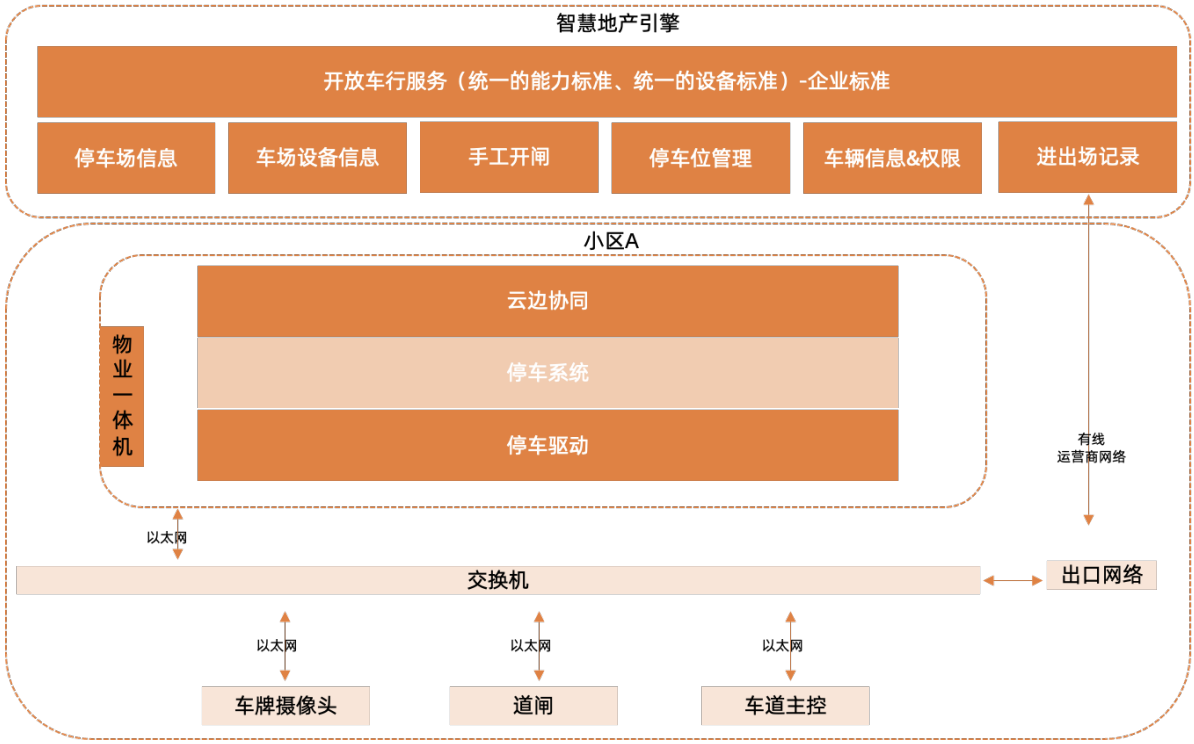
3.2.2. 车行版一体机

产品架构

1. 车行版（系统方案），保留原有本地停车系统，通过物业一体机中的停车驱动和本地停车管理软件对接，将系统服务能力抽象至云端，为云上SaaS提供服务



2. 车行版（设备方案），取消原有本地停车系统，保留停车硬件，通过物业一体机中的停车驱动和停车硬件进行对接，数据由部署在物业一体机中的定制开发停车系统消费，并通过云边协同数据通道和云上应用进行数据交换



方案推荐

1. 车行版（系统方案）采用保留原有系统的模式，将服务代理上云，方案适合于集中化弱管控的方式，这种方式可以将车场信息、车辆信息、进出行为做云上统一收口，但是无法将各厂商的差异化定价、收费逻辑做统一
2. 车行版（设备方案）采用保留原有硬件的模式，将边缘停车应用进行重新定义，方案适合于集中化强管控，这种方式可以通过本地和云上停车应用之间的协同，将全部停车数据进行管理

服务能力

支持功能	支持本地停车管理系统部署，服务8车道，4进4出的设备，支持跨品牌停车硬件接入，同时支持一体机通过云控制台快速替换
对接品牌	捷顺，富士，邦道，科拓，艾科，立方等品牌，详细型号及具体型号接入的功能范围见已对接设备清单，部分品牌的SDK授权需要客户和原厂进行申请
扩展服务	支持用户基于标准扩展框架自行对不同品牌停车硬件SDK进行二次开发，将停车硬件对接至本地停车管理系统

硬件规格

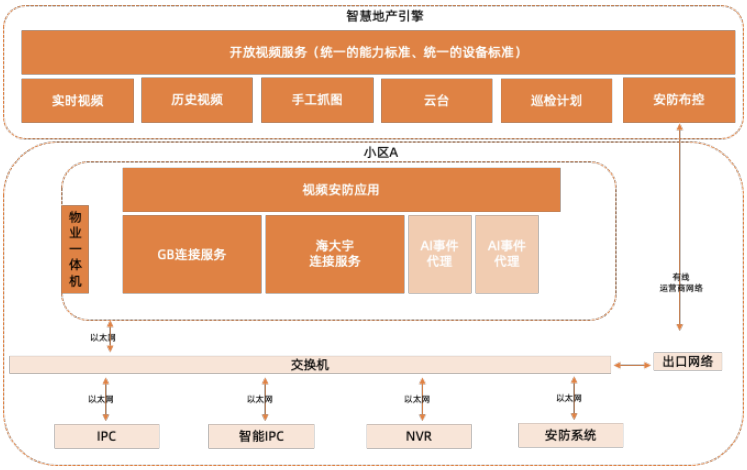
产品型号	备注
处理器	支持Intel® Core™ i5-6400，LGA1151
内存	8GB DDR4
磁盘	1 * 128GB SSD ， 1 * HDD Bay(可选),
网口	2 * Intel i210 Gbe LAN controller
USB	4 * USB3.0/2.0/1.1, 4 * USB2.0/1.1
箱体结构	铝型材外壳，具有良好的抗震性和抗冲击性
安装方式	桌面安装，壁挂安装
散热系统	鳍状铝型材散热
电源输入	标配DC 12V输入，具有过流、过压、反接保护措施
总体尺寸	240mm × 240mm × 95mm
整机重量	3KG
工作温度	-20 ~ 60° C (-4°F~140°F) (使用宽温等级固态硬盘); 0 ~ 45° C (32°F~113°F) (使用普通温度等级机械硬盘/固态硬盘)
功耗	待机：27Watt，满载：55Watt
系统	Linux/Link Edge

接入指引

1. 车行版（系统方案）
    i. 云端服务API：停车场系统领域服务-服务模型定义
    ii. 云端数据API：停车场系统领域服务-数据模型定义
2. 车行版（设备方案）
    i. 边缘设备接口

3.2.3. 视频版一体机

产品架构



服务能力

视频功能	支持通过本地IP网络对数字化NVR和IPC进行上云进行摄像头管理，包含H264视频实时、历史（需对接NVR）视频拉流、云台（需设备支持GB协议上云台功能）、截图功能
性能	支持1000路摄像头接入，同时支持10路1080P视频上云，满载运行需要规划40M带宽
对接品牌	支持通过GB协议对接海康、大华、宇视、天地伟业的主流IPC和NVR
扩展服务	支持用户基于标准扩展框架自行对安防AI设备事件进行二次开发，将AI事件上云

硬件规格

功能参数	描述
处理器	Intel Bay Trail Celeron J1900
内存	8GB DDR3
存储	256GB SSD
网口	2 * 10/100/1000Mbps 以太网，板载WIFI、4G模块可扩展
串口	1 * RS232
电源接口	DC12/24
总体尺寸	145mm*91.5mm*35.5mm
重量	≤0.7KG
散热系统	整机无风扇设计
安装方式	DIN标准轨道安装
工作温度	-20~65℃
电源输入电压范围	12-24V

功耗	典型功耗8W，最大功耗：20W
系统	Linux/Link IoT Edge

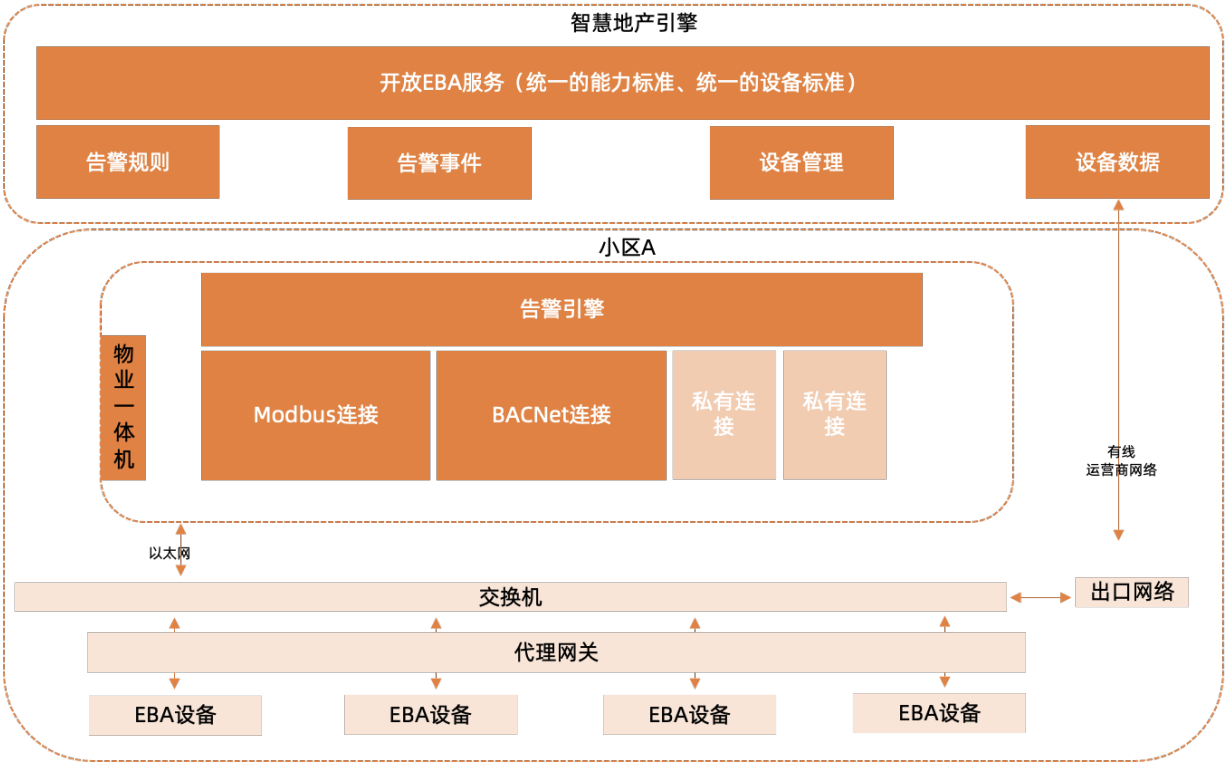
接入指引

视频API：视频监控服务API

智能安防API：智能安防服务API

3.2.4. EBA版一体机

产品架构



服务能力

支持功能	支持本地基于Modbus、BACNet协议接入的EBA设备连接及数据上云，支持云端配置边缘报警规则，并将告警事件上云
设备连接	支持Modbus、BACNet两种标准协议的可视化配置接入，同时支持用户基于标准扩展框架自行对其他非标协议进行二次开发
设备规模	• 支持1000个BA设备的接入和管理，支持每个设备200个点位管理 • 支持标准协议每分钟1w点位数据上报
告警服务	支持指定时间范围内对设备属性数值（大于、小于、等于、不等于、范围内、范围外）、持续时间、持续次数进行多条件判断，当满足条件时触发报警事件
开发服务	支持云端通过接口配置边缘告警规则，同时支持云边两端订阅告警事件

硬件规格

功能参数	描述

处理器	Intel Bay Trail Celeron J1900
内存	8GB DDR3
存储	256GB SSD
网口	2 * 10/100/1000Mbps 以太网，板载WIFI、4G模块可扩展
串口	1 * RS232
电源接口	DC12/24
总体尺寸	145mm*91.5mm*35.5mm
重量	≤0.7KG
散热系统	整机无风扇设计
安装方式	DIN标准轨道安装
工作温度	-20~65℃
电源输入电压范围	12-24V
功耗	典型功耗8W，最大功耗：20W
系统	Linux/Link IoT Edge

接入指引

- 云端API请查阅：设备告警开放服务API

3.2.5. 通行版一体机

产品架构

通行版一体机，采用阿里云边缘容器底座，能够将CPU/内存资源虚拟化，支持CPU 1/1000核和内存MB级别的容器级别调度，集成了人行版和车行版的软件能力，可以支撑人行和车行的场景。

服务能力

- 采用1U机架式智能散热系统设计，区域调控温度，充分降低整机功耗；主板集成1Gb双口网卡，并支持低成本的板载直连存储和RAID存储。
- 采用阿里云边缘容器底座，能够将CPU/内存资源虚拟化，支持CPU 1/1000核和内存MB级别的容器级别调度，稳定可靠的支撑边缘SaaS
- 通过云端管控平台，实时下发协议和设备配置，支持主流品牌的门禁设备和停车场设备接入和管理
- 支持4进4出的停车场管理，同时处理每5秒进出一辆车的系统请求。
- 支持云端导入人脸图片，下发边缘建立人脸库
- 支持200个门禁设备接入和管理。其中人脸识别通行，支持2W人脸底库管理，性能达到200次/分钟比对，比对准确率99%以上；其中二维码通行，支持2W加密码管理，性能达到200次/分钟；其中还支持2W容量IC卡号管理。
- 支持断网，不影响人员通行、车辆通行
- 预装阿里云IoT云边协同组件，可以远程SSH容器级运维及主机，提供安全通道由云端直达边缘停车SaaS和主机进行运维管理。

硬件规格

功能参数	描述
处理器	海光3185：8核/16线程 _3.0GHz

内存	32GB DDR4
存储	480GB SSD，2T HDD
网口	板载集成千兆双口（Intel I210 AT） _
电源	单电源250W
电源电压	200-240V/50Hz
机箱尺寸	1U机架：554mm（深度）x 432mm（宽度）x 43.5mm（高度）
重量	最大18千克（不含导轨）
工作温度	工作时5℃～40℃(41℉～104℉) 存储-40℃～60℃(-40℉～140℉)
操作系统	Linux/Link IoT Edge

人员通行开发指引

- 人员账号服务API：人员账号服务API
- 传统门禁服务API：传统门禁服务API
- 人脸门禁服务API：人脸门禁服务API
- 对讲服务请查阅：可视对讲服务API
- 梯控服务API：梯控联动服务API
- 可视对讲设备端SDK:

可视对讲移动端SDK接口说明-[iOS]

可视对讲移动端SDK接口说明[Andriod]

车辆通行开发指引

1. 车行版（系统方案）
    i. 云端服务API：停车场系统领域服务-服务模型定义
    ii. 云端数据API：停车场系统领域服务-数据模型定义
2. 车行版（设备方案）
    i. 边缘设备接口

3.3. 实施配置

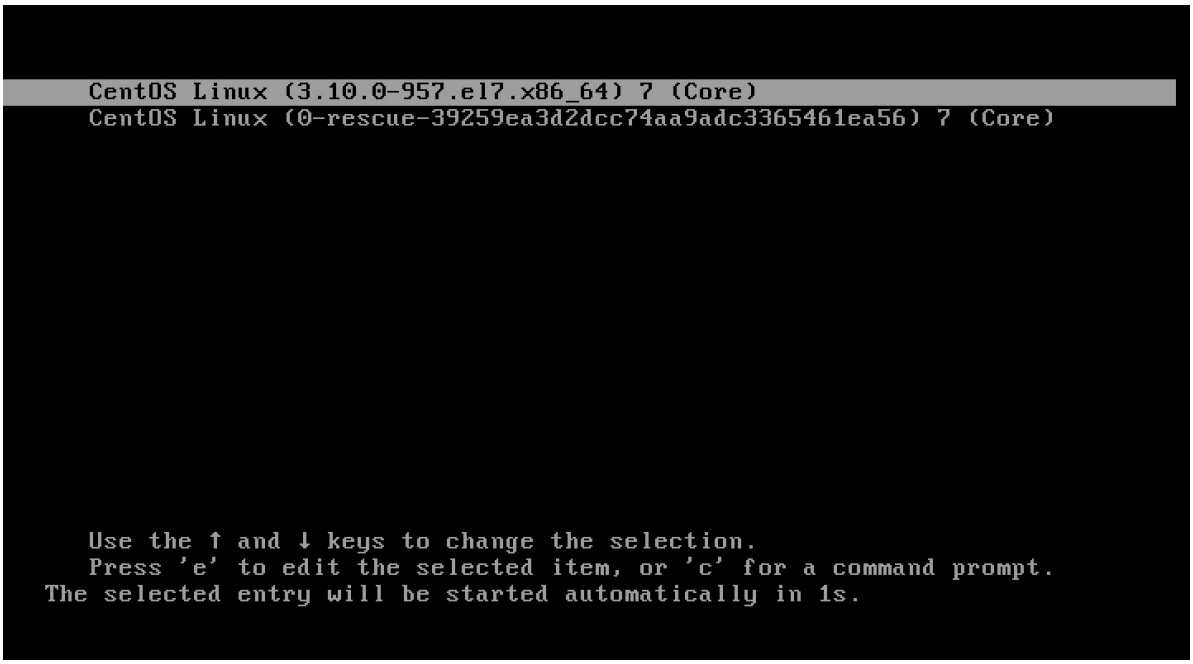
3.3.1. 部署初始化

本文为物业一体机部署实施手册，主要包括物业一体机的网络配置，边缘集群创建，人行/车行/安防/EBA四类场景的组件安装部署，以及设备绑定空间操作流程。人行/车行/安防/EBA四类设备的部署方法，请参见与本手册配套的相应的设备实施手册。

物业一体机和相关设备的部署实施，大部分操作都可以通过登录阿里巴巴的**物联网应用服务平台远程**进行操作，极少部分的操作，需要在项目**现场**进行。下面手册内容，如果在章节名称前有**[远程]**，则说明该章节及下面的所有子章节的部署操作，都可以通过远程进行；如果在章节名称前有**[现场]**，则说明该章节及下面的所有子章节的部署操作，都需要在现场进行。

> **说明**
>
> 请特别注意，该实施手册只适用于物业一体机。如果项目落地的设备不是物业一体机，是AIBOX产品时，请参考AIBOX实施手册。物业一体机对应的产品型号有下面几种。具体型号信息查看设备的背面产线信息标牌
>
> - 物业管理一体机-楼宇版：LE-A-B001-SC001
> - 物业管理一体机-人行版：LE-A-B002-SC001
> - 物业管理一体机-停车版：LE-A-B011-SC001
> - 物业管理一体机-通行版：LE-A-S001-SC001
> - 物业管理一体机-综合版：LE-A-S006-SC001

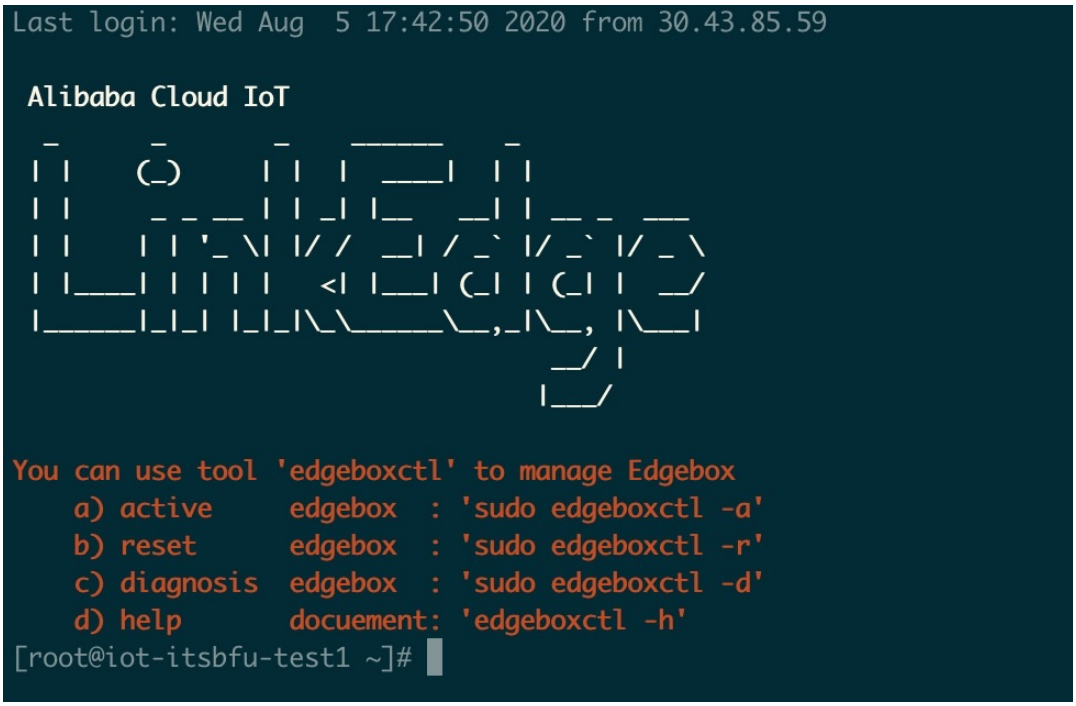
1. [现场]IP配置及网路测试，边缘一体机开机后，选择第一个选项：CentOS_Linux (3.10.XXXX)



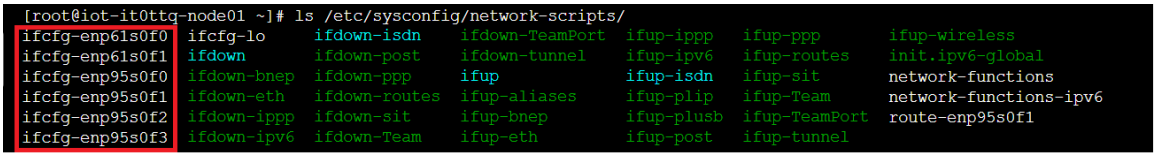
i. 系统启动完成后，需要输入用户名和密码。**请输入：用户名： owner 密码：changepassword**



ii. 登录成功后，显示下面的界面：



iii. 输入命令：ls /etc/sysconfig/network-scripts



上面示例，显示有6个网卡配置文件（ifcfg-enp61s0f0、ifcfg-enp61s0f1、ifcfg-enp95s0f0、ifcfg-enp95s0f1、ifcfg-enp95s0f2、ifcfg-enp95s0f3），以上网卡配置文件名称和数目，不同的机型也存在变化，请根据设备的实际情况进行选择。

2. 说明

基础信息对齐

i. 概念介绍

a. VLAN虚拟局域网

b. 网关(Gateway)网间连接器或协议转换器

ii. 进场条件

a. 项目提供网络规划表，参考如下：

应用分类	IP 段规划	备注与示例	VLAN
网络设备管理地址	10.249.0.0/24	网关：10.249.0.1，256 个地址	Vlan100
网络设备互联地址	10.249.1.0/24	网关：10.249.1.1，256 个地址	Vlan101
智慧社区人行设备	10.249.32.0/21	网关：10.249.32.1，2048 个地址	Vlan108
智慧社区车行设备	10.249.12.0/24	网关：10.249.12.1，256 个地址	Vlan112
智慧社区设备设施	10.249.16.0/24	网关：10.249.16.1，256 个地址	Vlan116
智慧社区安防设备	10.249.20.0/22	网关：10.249.20.1，1024 个地址	Vlan120
智慧社区服务器	10.249.24.0/24	网关：10.249.24.1，256 个地址	Vlan124

要求：

1. 设备地址，开始 10 个和最后 15 个预留给网络使用。例如安防网段，从 10.249.20.11 开始使用，用到 10.249.23.240。
2. 系统专属服务器，从规划的地址段后面开始往前用。例如安防系统的录像机，从 10.249.23.240 开始往前使用，10.249.23.239，10.249.23.238……
3. 智慧社区平台一级的服务器设备，按规划的网段使用，10.249.24.11 开始往后。

b. 现场交换机的VLAN配置完成，并且所有VLAN通外网

3. 网卡网络配置，项目标准方案为一个网口连接内网和外网，默认是使用**单网卡方案**做配置；（注：只有现场评估短时间无法解决单网卡连接内网和外网情况下，紧急情况才可以使用双网卡）

BOOTPROTO	IP地址获取方式，static为静态地址，DHCP为动态获取地址。要求都配置为static
IPADDR	IP地址
NETMASK	子网掩码
GATEWAY	网关地址
DNS1	DNS服务器地址
DEFROUTE	默认路由，no为此网卡非默认路由，yes为此网卡默认路由。要求：外网网卡需设置为yes，内网网卡需设置为no
ONBOOT	开机是否自动运行，no为开机不自启动，yes为开机自启动。要求：对应的外网和内网网卡都设置成yes
PEERDNS	若该选项指定为yes，则表示网络启动后加载的dns服务器是从/etc/resolv.conf读取 若该选项指定为no，则表示网络启动后加载的dns服务器为该配置文件内的DNS参数，而不是/etc/resolv.conf文件。要求：配置成no

i. 单网卡方案配置(优先)，单网卡方案即只有一根网线可以同时连接外网和内网。项目施工优先选择该方案，配置简单。注意：单网卡方案，请不要在一个网卡上配置多个IP，例如通过虚拟网卡的方式多添加一个IP。如果的确有多个IP的需求，请采用双网卡方案

a. IP地址配置，通过vi修改并保存文件：sudo vi /etc/sysconfig/network-scripts/ifcfg-enp95s0f0（enp95s0f0 为对应网卡名称，10.249.24.20为物业一体机IP地址。实际填写内容根据项目情况调整），**注意：配置单网卡时，需要确认其他网卡配置的DEFROUTE=no或者不配置DEFROUTE才可以！**

```
BOOTPROTO=static
IPADDR=10.249.24.20
NETMASK=255.255.255.0
GATEWAY=10.249.24.1
DNS1=223.5.5.5
PEERDNS=no
DEFROUTE=yes
ONBOOT=yes
DEVICE="enp95s0f0"
```

ii. 双网卡方案配置(可选)，双网卡方案即需要两跟网线，一根网线连接外网，另外一根连接内网。从左算第一个网口连接外网，第二个网口连接内网。

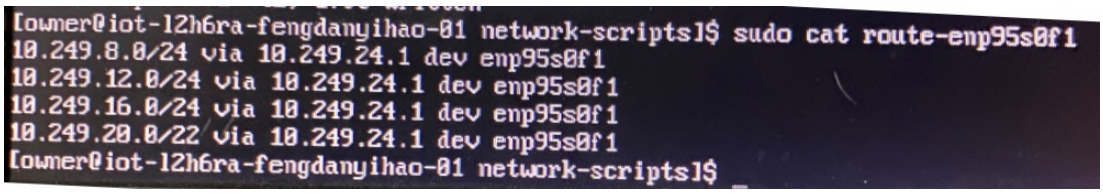
a. IP地址配置-第一个网口，通过vi修改文件：sudo vi /etc/sysconfig/network-scripts/ifcfg-enp95s0f0（enp95s0f0 为对应网卡名称，10.249.24.20为物业一体机IP地址。实际填写内容根据项目情况调整）

```
BOOTPROTO=static
IPADDR=10.249.24.20
NETMASK=255.255.255.0
GATEWAY=10.249.24.1
DNS1=223.5.5.5
PEERDNS=no
DEFROUTE=yes
ONBOOT=yes
DEVICE="enp95s0f0"
```

b. IP地址配置-第二个网口，通过vi修改文件：sudo vi /etc/sysconfig/network-scripts/ifcfg-enp95s0f1（enp95s0f0 为对应网卡名称，10.249.24.20为物业一体机IP地址。实际填写内容根据项目情况调整）

```
BOOTPROTO=static
IPADDR=10.249.24.20
NETMASK=255.255.255.0
GATEWAY=10.249.24.1
DEFROUTE=no
ONBOOT=yes
DEVICE="enp95s0f1"
```

c. 配置静态路由-第二个网口，需要为内网网口配置联调四个子系统（人行、车行、安防、EBA）所在vlan的静态路由通过vi新建文件：sudo vi /etc/sysconfig/network-scripts/route-enp95s0f1（enp95s0f1 为对应网卡名称，以设备实际名称为准）为每一个目标vlan配置静态路由，其中10.249.24.1是物业一体机IP对应的Gateway地址。



a. 10.249.8.0/24 via 10.249.24.1 dev enp95s0f1

b. 10.249.12.0/24 via 10.249.24.1 dev enp95s0f1

c. 10.249.16.0/24 via 10.249.24.1 dev enp95s0f1

d. 10.249.20.0/22 via 10.249.24.1 dev enp95s0f1

e. 10.249.32.0/21 via 10.249.24.1 dev enp95s0f1

iii. 重新启动系统，配置完成后需要重新启动系统。使网关配置生效，然后进行网络测试（重启命令：sudo reboot）

4. 网络配置诊断

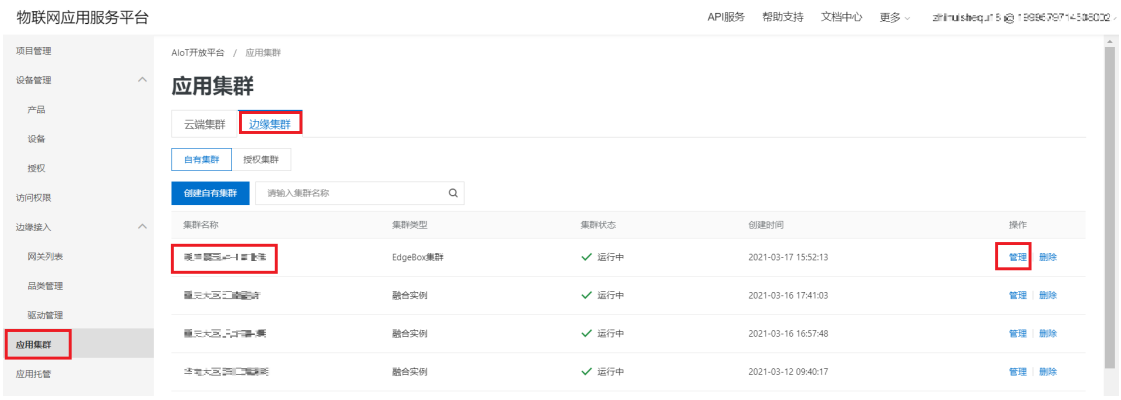
i. 登录物业一体机：有两种方法登录物业一体机

a. 现场项目，直接开机登录

b. 当物业一体机已经激活，可以远程登录：

登录阿里云物联网平台，对该项目对应的边缘实例进行操作，具体步骤如下：

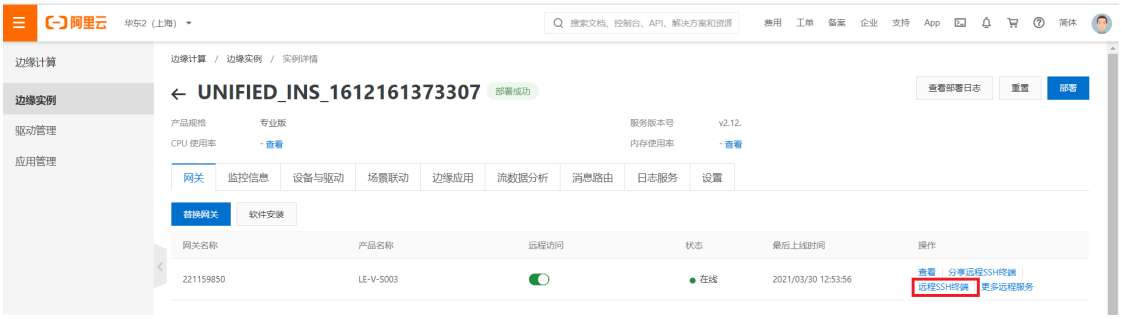
点击"应用集群-边缘集群"，点击对应项目的边缘集群右侧的**管理**。如图所示



点击**组件管理**，找到Link Edge边缘设备接入组件，点击**实例**。如下图所示。如果是融合实例，点击右侧的**前往实例**



在打开的实例详情页面，点击**远程SSH终端**，如下图所示



在弹出的页面，输入用户名和密码：owner，changepassword



ii. 运行网络配置诊断软件：登录到物业一体机后，输入下面命令，直接回车即可运行网络配置诊断软件

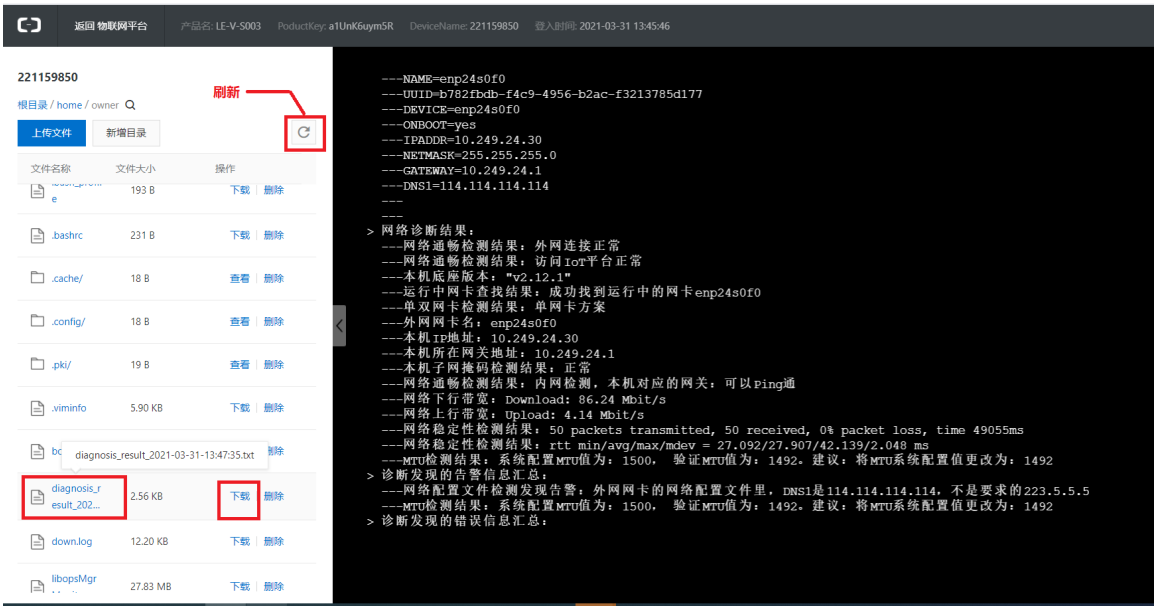
curl -s https://face-edge-test.oss-cn-beijing.aliyuncs.com/network_diagnosis/network_diagnosis.sh | bash

诊断软件运行大概需要2～3分钟，运行完后诊断结果将在屏幕显示，诊断出的告警信息和错误信息也将汇总展示，如下图示例：



iii. 网络配置诊断结果下载存档保留：诊断软件会将有效的网卡配置文件，路由配置文件，诊断结果信息保存到运行诊断软件的当前目录，文件名以diagnosis_result_开头，后面为诊断进行的时间，例如：diagnosis_result_2021-03-31-13:47:35.txt。这个文件需要下载存档保留。

诊断软件运行完，先点击左侧**刷新**按钮，然后找到诊断结果文件，点击该文件右侧的**下载**，则该文件将下载到个人电脑里。



iv. 诊断出的问题处理建议

- 如果诊断发现外网不通，请参照下一节**边缘服务器无法访问外网**进行排查
- 如果本机对应的网关无法ping通，请参照下一节**边缘服务器无法连接某个设备**进行排查
- vlan对应的网关如果无法ping通：

(1) 单网卡方案：证明交换机VLAN划分未完成

(2) 双网卡方案：先使用sudo route命令排查静态路由表是否与**双网卡方案配置**里静态路由配置一致，如果路由表正确证明交换机VLAN划分未完成

5. 网络异常问题排查方法

i. 服务器无法访问外网

a. 单网卡方案

a. ping 223.5.5.5，如果ping的通，证明网络可以通外网，只是域名无法解析，说明本机DNS配置错误，请检查网卡的DNS配置是否与1.1.3.1一致，排查结束；如果ping不通，继续步骤2；

b. ping 10.249.24.1，如果ping的通，证明本机的IP配置正确，请通知项目侧检查外网不通，排查结束；如果ping不通，继续步骤3；

c. 检查本机网卡的IP地址和GATEWAY配置是否与1.1.3.1一致，如果检查无异常，请通知现场检查网线是否插入，网线是否插入到交换机的正确端口上；

b. 双网卡方案

a. ping 223.5.5.5，如果ping的通，证明网络可以通外网，只是域名无法解析，说明本机DNS配置错误，请检查网卡的DNS配置是否与1.1.3.2一致，排查结束；如果ping不通，继续步骤2；

b. 通过sudo route命令检查default路由项是否在网卡1上，如果没有在网卡1，请检查两个网卡的默认路由配置是否与1.1.3.2一致，排查结束；如果在网卡1，请继续步骤3；

```
[sudo] password for owner:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         gateway         0.0.0.0         UG    101    0        0 enp95s0f0
10.0.0.0        0.0.0.0         255.0.0.0       U     100    0        0 enp95s0f1
10.78.128.0     0.0.0.0         255.255.255.128 U     0      0        0 cni0
10.249.8.0      10.249.24.1     255.255.255.0   UG    100    0        0 enp95s0f1
10.249.12.0     10.249.24.1     255.255.255.0   UG    100    0        0 enp95s0f1
10.249.16.0     10.249.24.1     255.255.255.0   UG    100    0        0 enp95s0f1
10.249.20.0     10.249.24.1     255.255.252.0   UG    100    0        0 enp95s0f1
10.249.32.0     10.249.24.1     255.255.248.0   UG    100    0        0 enp95s0f1
169.254.123.0   0.0.0.0         255.255.255.0   U     0      0        0 docker0
192.168.58.0    0.0.0.0         255.255.255.0   U     101    0        0 enp95s0f0
[owner@iot-it0ttq-node01 network-scripts]$
```

c. ping外网网卡的路由器ip地址，如果ping的通，证明本机的IP配置正确，请通知项目侧检查外网不通，排查结束；如果ping不通，继续步骤4；

d. 检查网卡1的IP地址和GATEWAY配置是否与1.1.3.2一致，如果检查无异常，请通知现场检查网线是否插入，网线是否插入到交换机的正确端口上；

ii. 服务器无法连接某个设备

a. 单网卡方案

a. ping设备所在的VLAN网关IP，如果ping的通，证明是设备不存在或者IP地址不对，请通知项目组检查设备IP配置，排查结束；如果ping不通，继续步骤2；

b. ping 10.249.24.1，如果ping的通，证明本机的IP配置正确，请通知项目侧检查交换机网络配置，排查结束；如果ping不通继续步骤3

c. 检查本机网卡的IP地址和GATEWAY配置是否与1.1.3.1配置一致，如果检查无异常，请通知现场检查网线是否插入，网线是否插入到交换机的正确端口上

b. 双网卡方案

a. ping设备所在的VLAN网关IP，如果ping的通，证明是本地IP和交换机配置正确，继续排查本机的静态路由表的子网掩码是否与1.1.3.2一致，如果静态路由表正确，证明是设备不存在或者IP配置错误，请通知项目组检查设备IP配置，排查结束；如果ping不通VLAN网关IP，继续步骤2；

b. 通过sudo route命令检查静态路由表项是否有对应网段在网卡1上，如果没有，请检查静态路由表是否配置是否与1.1.3.2一致，排查结束；如果静态路由正常，请继续步骤3；

```
[sudo] password for owner:
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         gateway         0.0.0.0         UG    101    0        0 enp95s0f0
10.0.0.0        0.0.0.0         255.0.0.0       U     100    0        0 enp95s0f1
10.78.128.0     0.0.0.0         255.255.255.128 U     0      0        0 cni0
10.249.8.0      10.249.24.1     255.255.255.0   UG    100    0        0 enp95s0f1
10.249.12.0     10.249.24.1     255.255.255.0   UG    100    0        0 enp95s0f1
10.249.16.0     10.249.24.1     255.255.255.0   UG    100    0        0 enp95s0f1
10.249.20.0     10.249.24.1     255.255.252.0   UG    100    0        0 enp95s0f1
10.249.32.0     10.249.24.1     255.255.248.0   UG    100    0        0 enp95s0f1
169.254.123.0   0.0.0.0         255.255.255.0   U     0      0        0 docker0
192.168.58.0    0.0.0.0         255.255.255.0   U     101    0        0 enp95s0f0
[owner@iot-it0ttq-node01 network-scripts]$
```

c. ping 10.249.24.1，如果ping的通，证明本机的IP配置正确，请通知项目侧检查交换机网络配置，排查结束；如果ping不通，继续步骤4；

d. 检查网卡2的IP地址和GATEWAY配置是否与1.1.3.2一致，如果检查无异常，请通知现场检查网线是否插入，网线是否插入到交换机的正确端口上

iii. 设备无法连接服务器

a. 从设备ping 10.249.24.1，如果ping不通，证明问题跟边缘服务器无关，请通知项目组检查设备或者网络，如果ping的通请继续步骤2

b. 检查边缘服务器的IP地址是否配置正确

3.3.2. 激活一体机

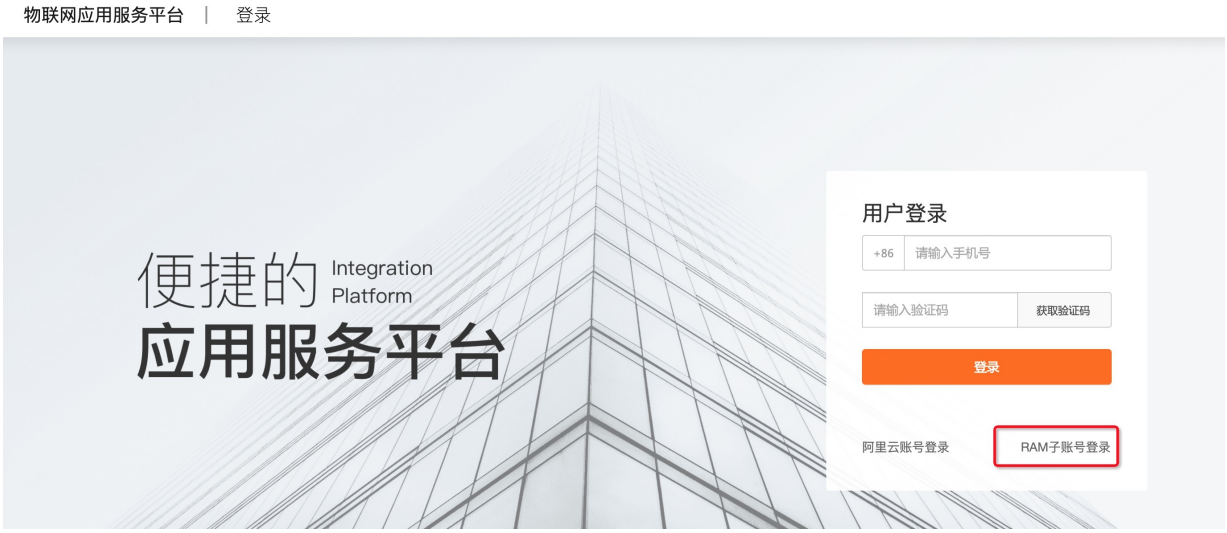
本章介绍如何将一体机做激活操作

1 登录阿里云应用服务平台

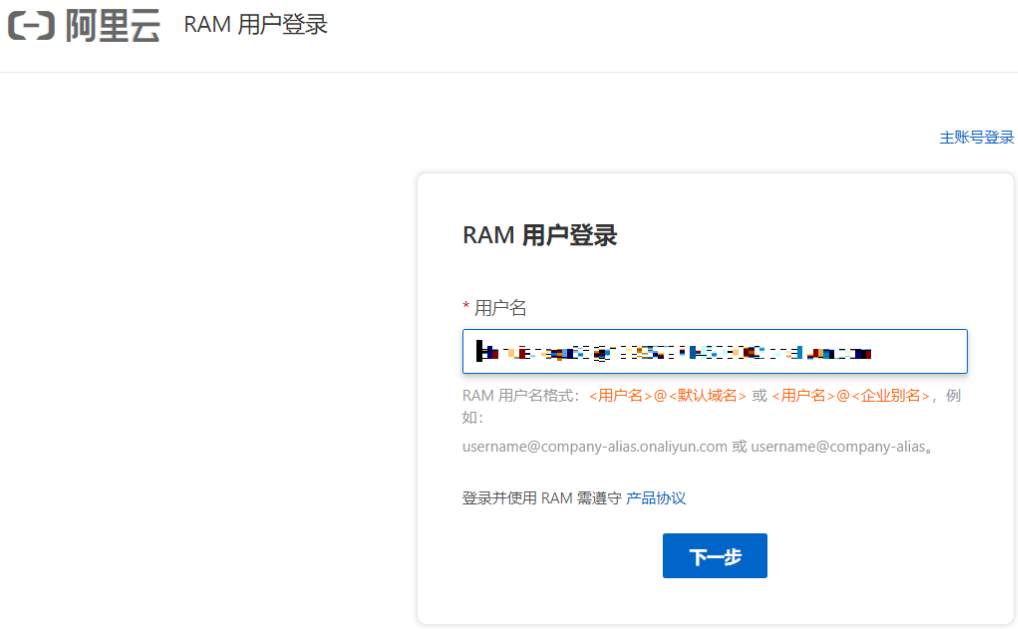
使用阿里云账号登录阿里物联网应用服务平台。登录账号请联系项目负责人获取。

登录步骤如下：

1、输入si.iot.aliyun.com，展示登录界面，选择RAM子账号登录。

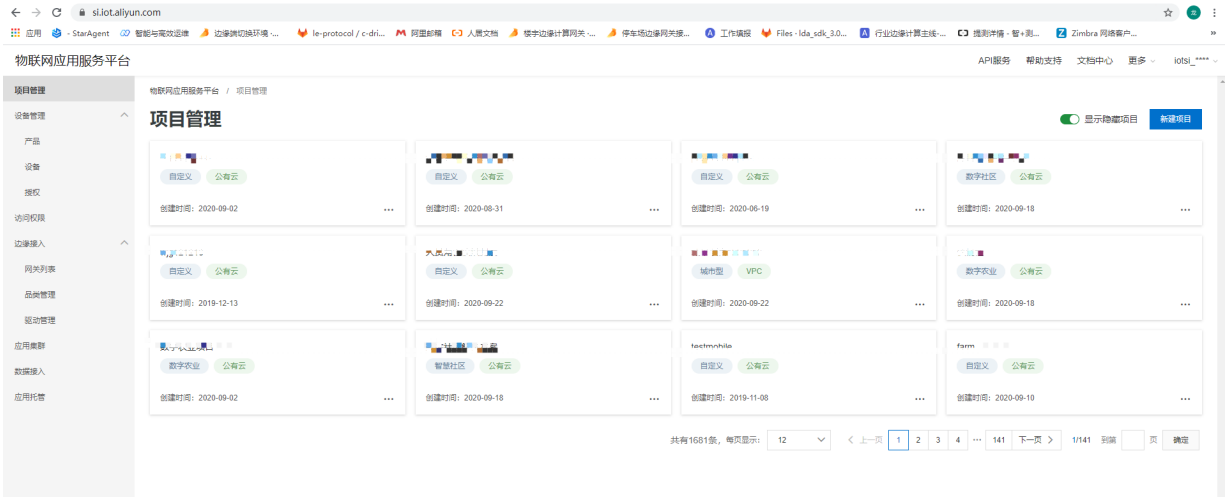


2、输入账号名和密码





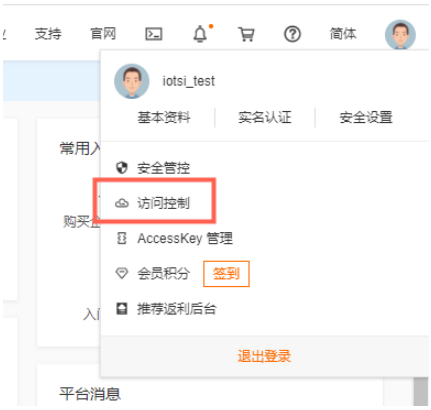
如果登录成功之后，可以看到如下界面，则接下来可以按照下一节内容进行操作。



- 如果没有出现以上界面而出现企业入驻页面，请参考如下处理方式。
- 如果子帐号登录后出现企业入驻页面，需按照提示填写内容，由阿里后台审核通过后才可使用SI平台。
- 另外还需要主账号为子账号开通AliyunIoTFullAccess权限。主账号登录

  https://iot.console.aliyun.com/

  平台后，在右上角点击访问控制，如下图所示：



- 进入RAM控制页面对子账号进行授权操作，如下图所示：



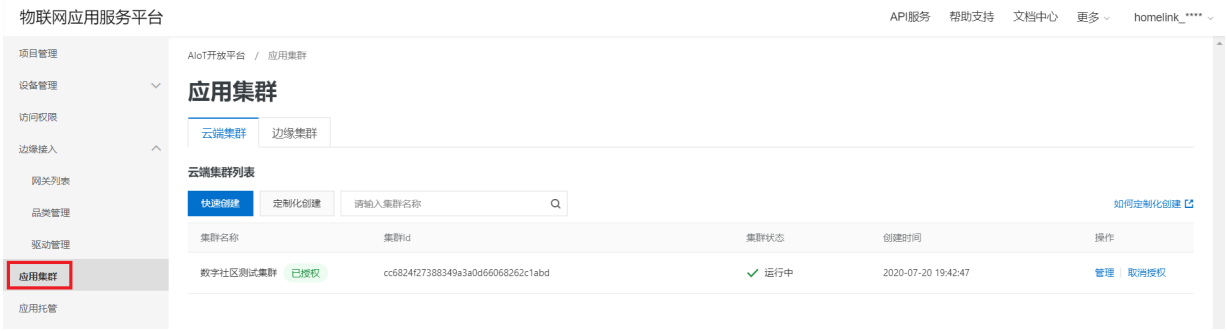
授权主体：即为待授权的RAM子账号

权限策略名称：选择**AliyunIOTFullAccess**即可

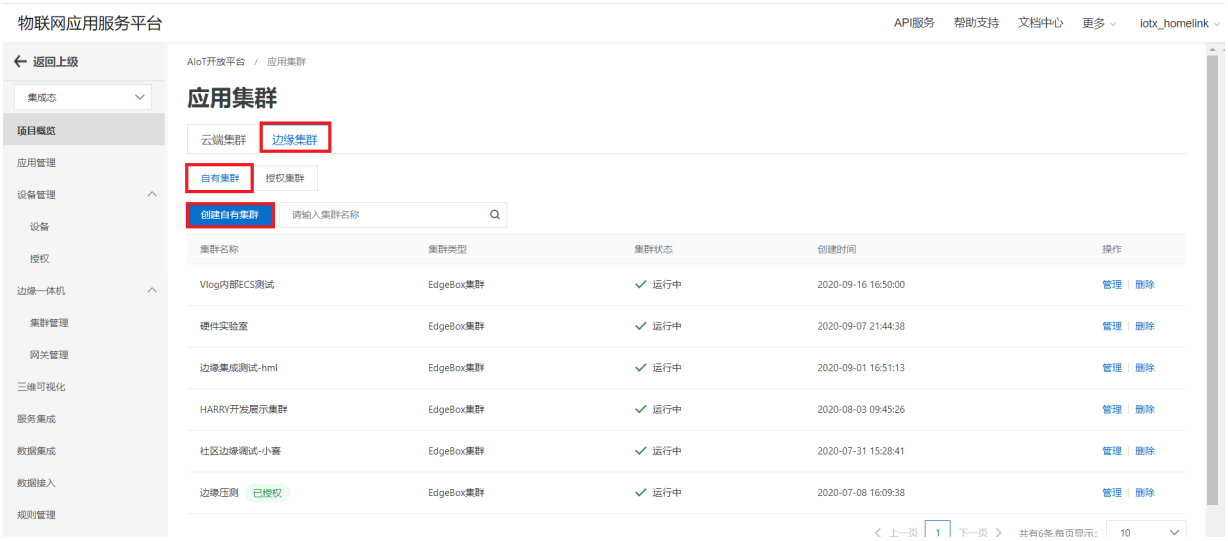
- 完成企业入驻审批及AliyunIoTFullAccess权限授权后可以使用SI平台

2 创建边缘集群

集群是负责管理边缘资源，完成项目整个实施，需要先创建集群。在物联网应用服务平台的顶级页面，点击应用集群。如下图所示：



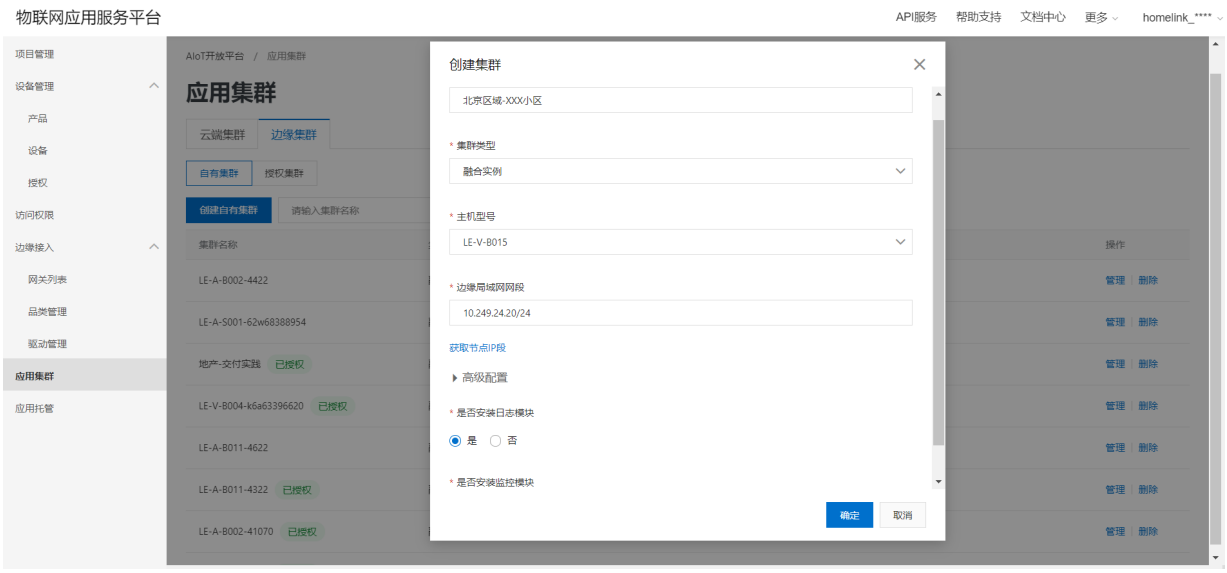
进入**边缘集群-自有集群**，点击**创建自有集群**，如下图所示



创建集群方法如下：

集群名称：集群命名规则为“区域名称-小区名称”，例如“北京区域-XXX小区”。具体名称联系项目组确定

集群类型：选择**融合实例**



主机型号：根据物业一体机设备上贴的标签里的**型号**信息进行选择，如下图示例

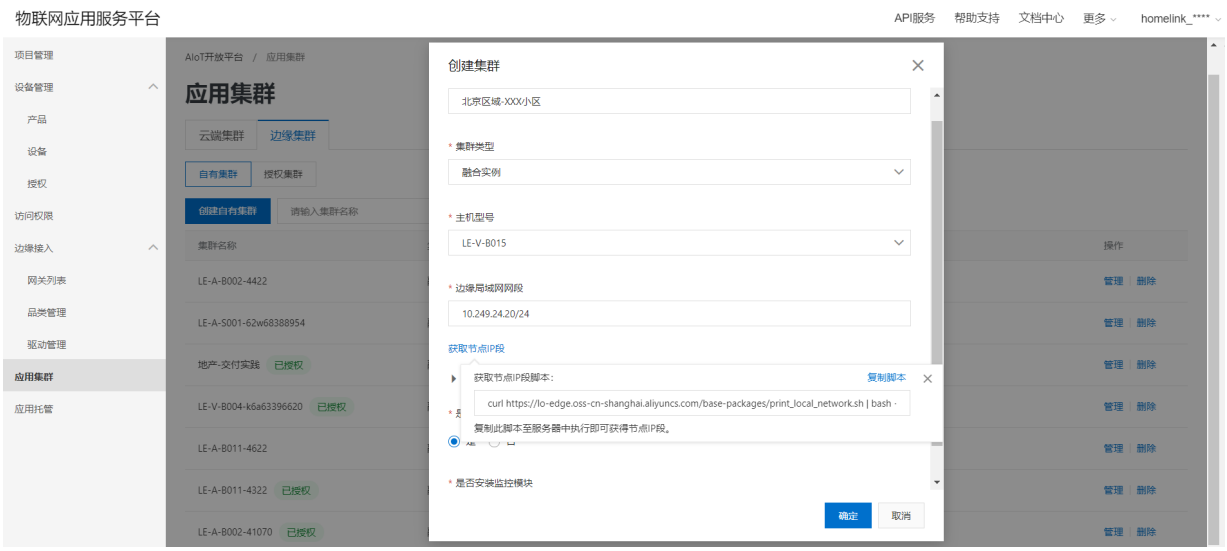


边缘局域网网段：边缘服务器地址固定为10.249.24.20，边缘局域网网段固定为10.249.24.20/24。如果边缘服务器地址有调整，点击**"获取节点IP段"**，复制脚本。将脚本在边缘服务器上运行，得到边缘局域网网段。

高级配置：不需要操作

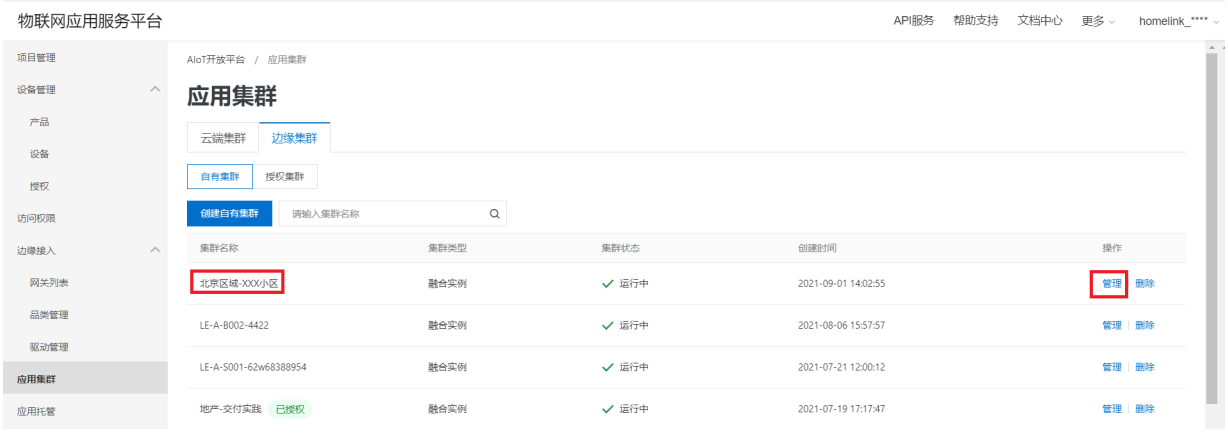
是否安装日志模块：是

是否安装监控模块：是



3 集群中添加融合实例节点

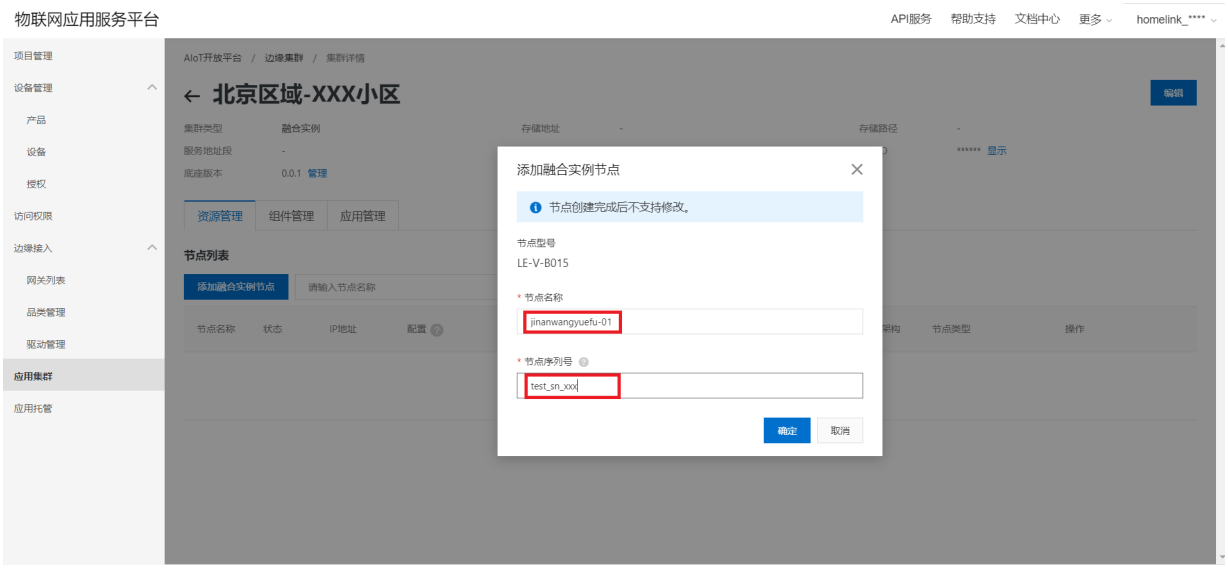
创建完成后的边缘集群后，点击对应集群的右侧**管理**，进入集群管理页面。



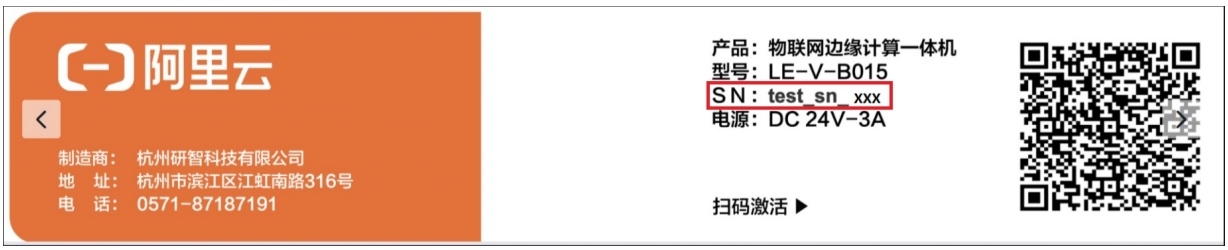
点击**添加融合实例节点**，如下图所示。



节点名称：命名要求：只支持数字、小写英文、短划线，不能以短划线开头和结尾，长度限制4-30。节点名称命名规则：“小区名称全拼-编号”，比如“jinanwangyuefu-01”



节点序列号：查看物业一体机设备上贴的标签，请将SN对应的序号填入，如下图所示：



点击确定，然后可以看见刚创建的节点，并处于未激活状态，如下图所示。

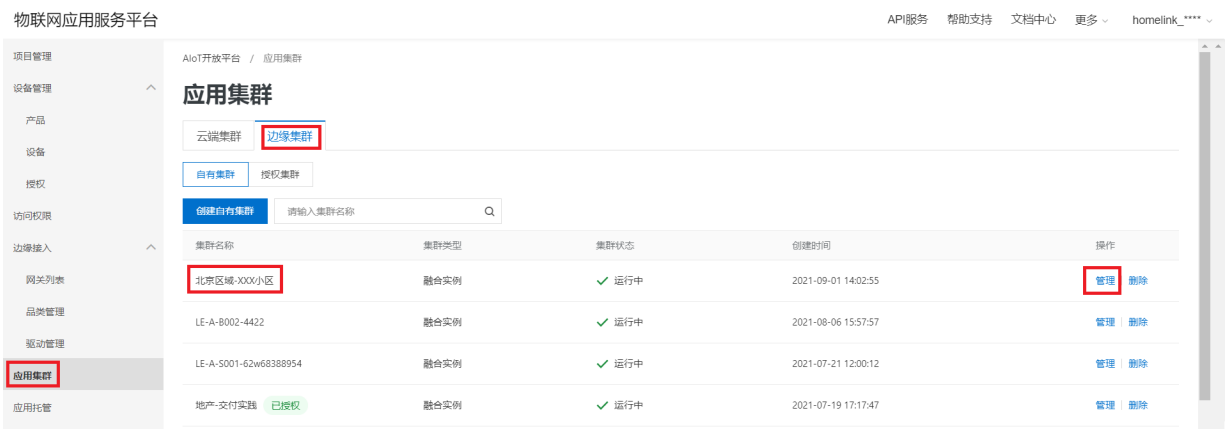


大概过5分钟，该节点将自动激活，状态处于运行中，如下图所示。可以刷新页面观察状态变化

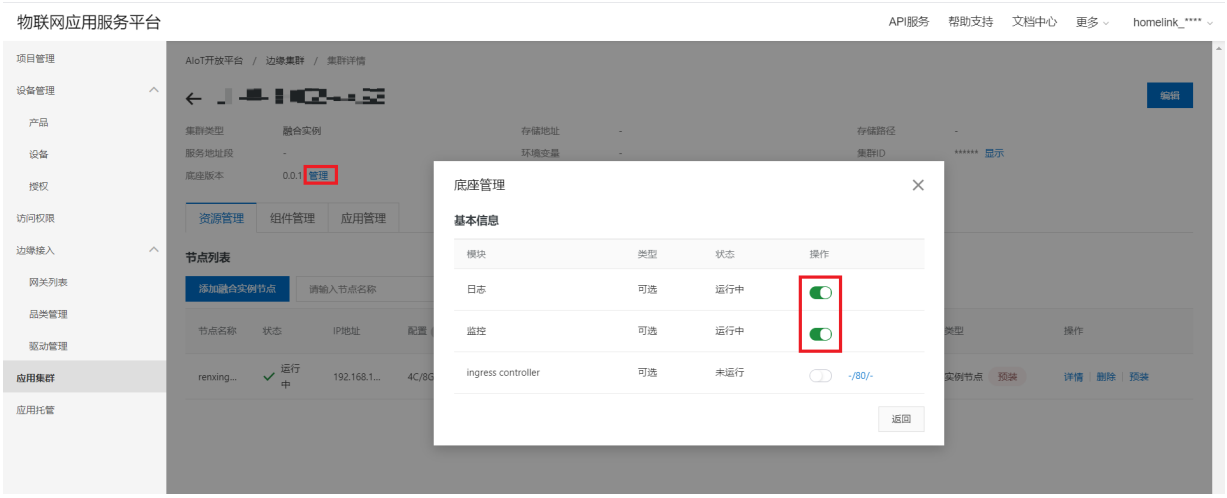


4 打开底座管理的日志和监控开关

在**应用集群>边缘集群**中查看之前创建的集群，点击集群的右侧**管理**，如下图所示



在打开的边缘集群里，点击底座版本右侧**管理**，在弹出的底座管理页面，点击打开日志和监控，如下图所示：



3.3.3. 部署运行组件

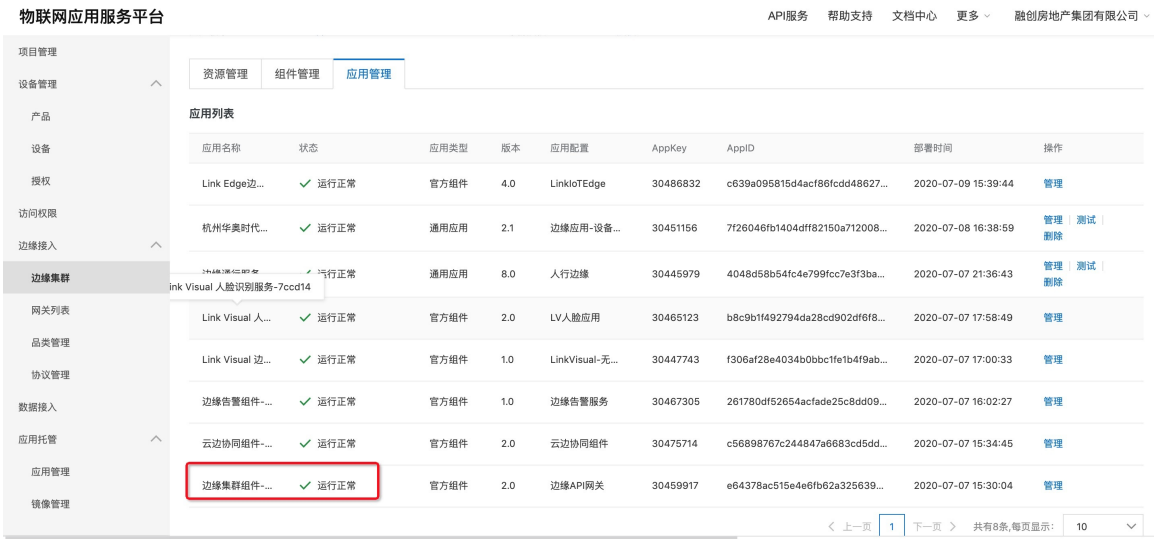
3.3.3.1. 部署基础组件

1. [远程]部署基础组件，边缘集群组件和LinkEdge边缘设备接入组件是基础组件，每个项目都会使用的。请按照下面说明进行部署。**不要部署非必要组件，否则会因为一体机资源不足导致必要组件部署失败**

i. 登录阿里云应用服务平台，进行下面操作：选择**应用集群**，点击对应集群右侧**管理**进入所选集群，选择**组件管理**。找到边缘集群组件，点击安装进行边缘集群组件的部署，如所示。



ii. 当部署完成之后，可以到**边缘集群>集群管理>应用管理**，查看组件是否部署成功，如下图所示：



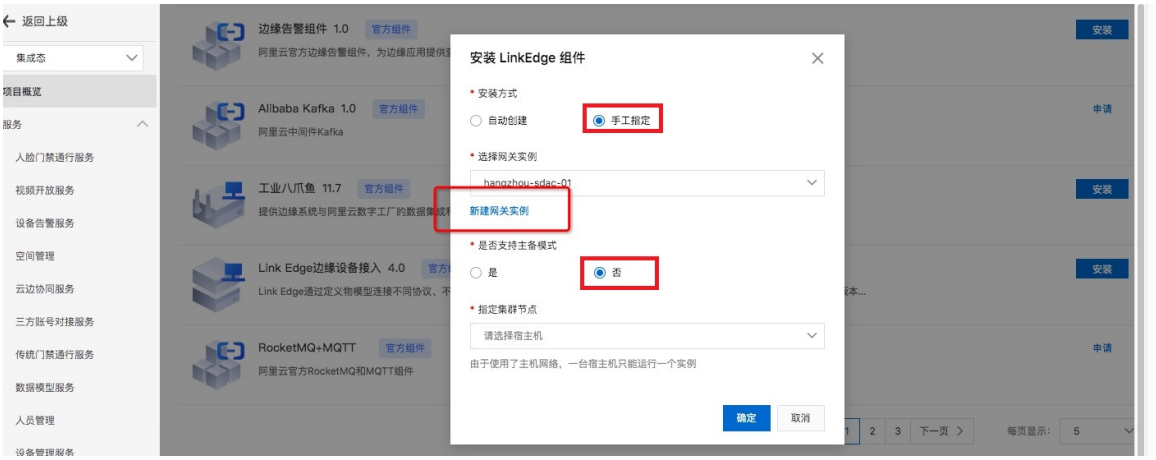
iii. 部署LinkEdge边缘设备接入组件，请完全参照本节说明进行部署。组件管理里的**LinkEdge边缘设备接入组件**右侧显示为**安装**，如下图所示：



iv. 选择**应用集群**，进入相应的集群，在**组件管理**里，选择Link Edge边缘设备组件的右侧**部署**，如所示。

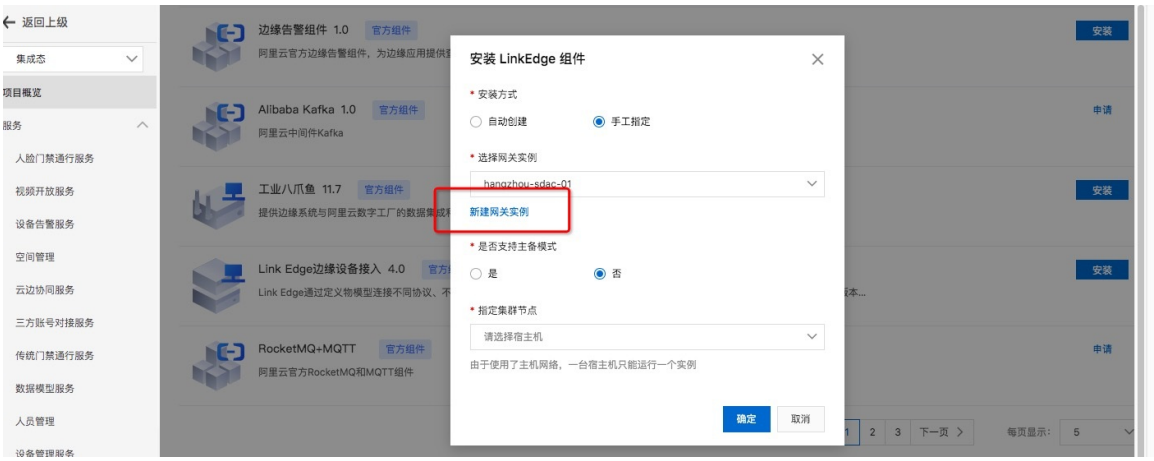


v. 如下图所示，进行LinkIoT Edge边缘设备组件信息的填写。

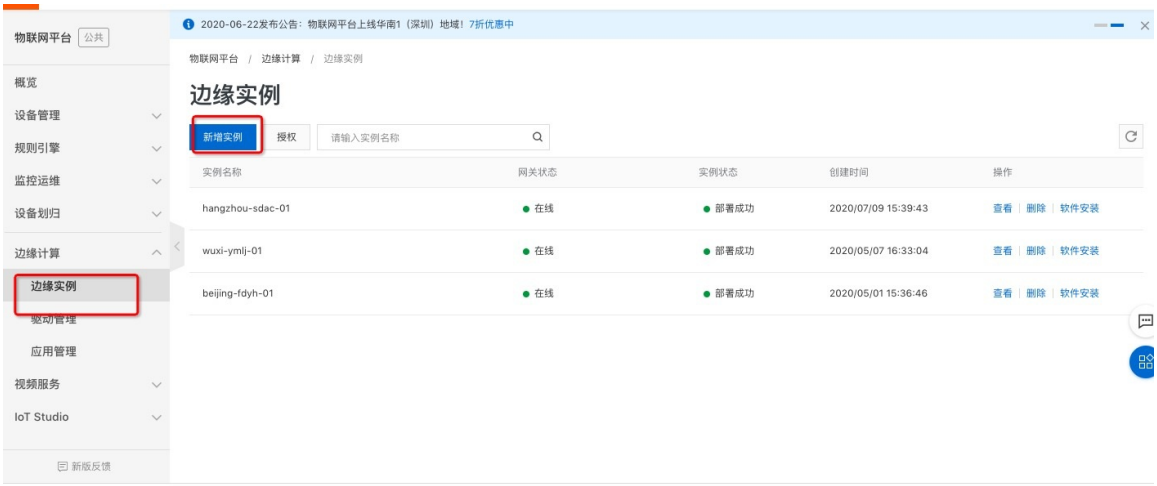


名词解释	参数说明
安装方式	选择手工指定
是否支持主备模式	选择否
指定集群节点	边缘服务器中节点的名称，选择某个节点代表该组件部署在这个节点中。一般一个集群只有一个服务节点，选择默认节点就好

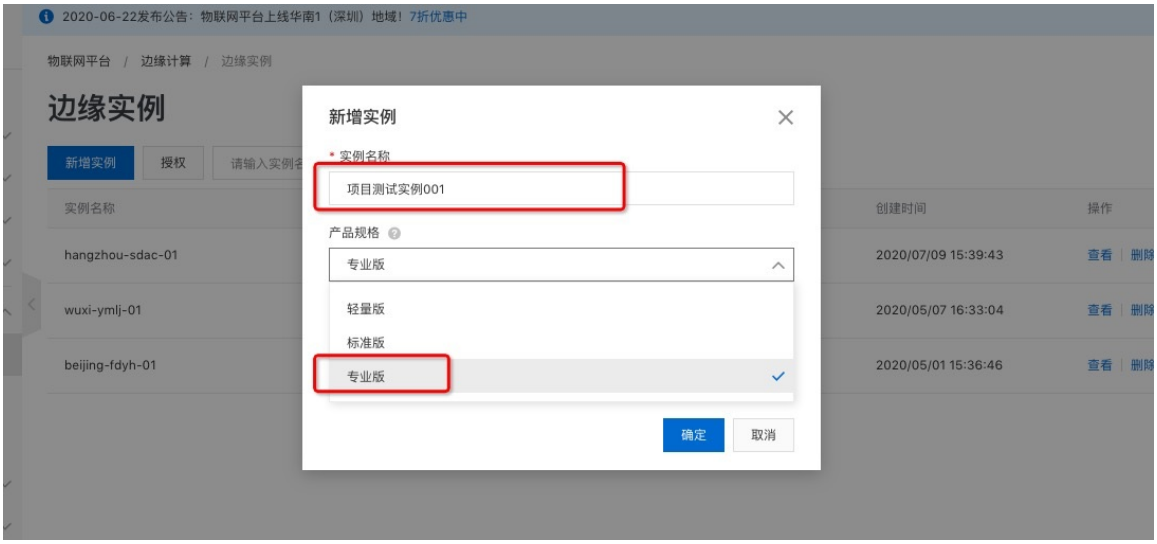
vi. 选择安装方式为手工指定，即为手动选择项目下的网关实例。所以这里需要创建属于项目的网关实例。如下图所示。



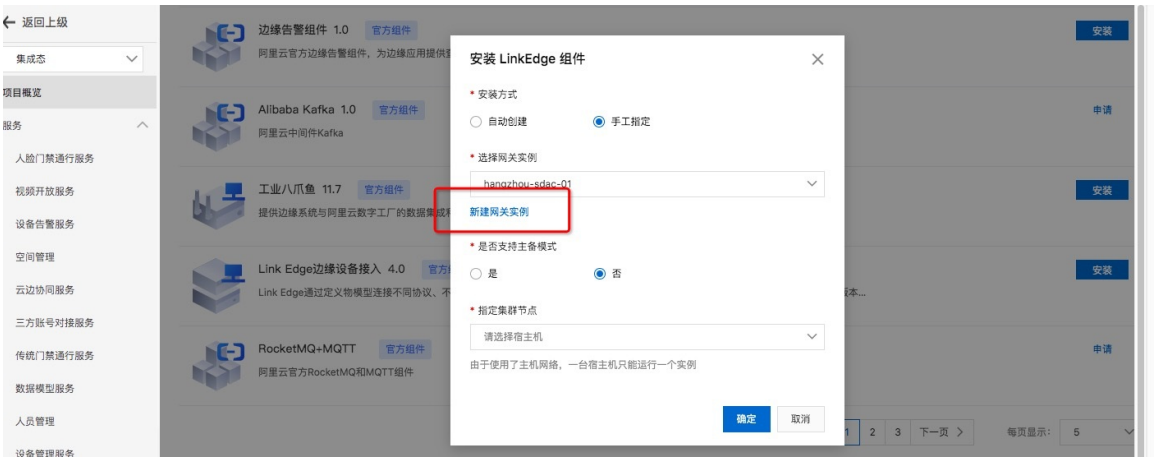
vii. 点击新建网关实例，会跳转到物联网平台，进行网关实例的创建。如下图所示。



viii. 填写网关实例名称，选择专业版，点击确定。如下图所示。



ix. 完成上述操作后，选择刚创建的网关实例，点击确定，完成LinkEdge边缘设备接入组件的安装，如下图所示。



x. 部署成功后，点击实例，查看边缘实例状态。如下图所示。



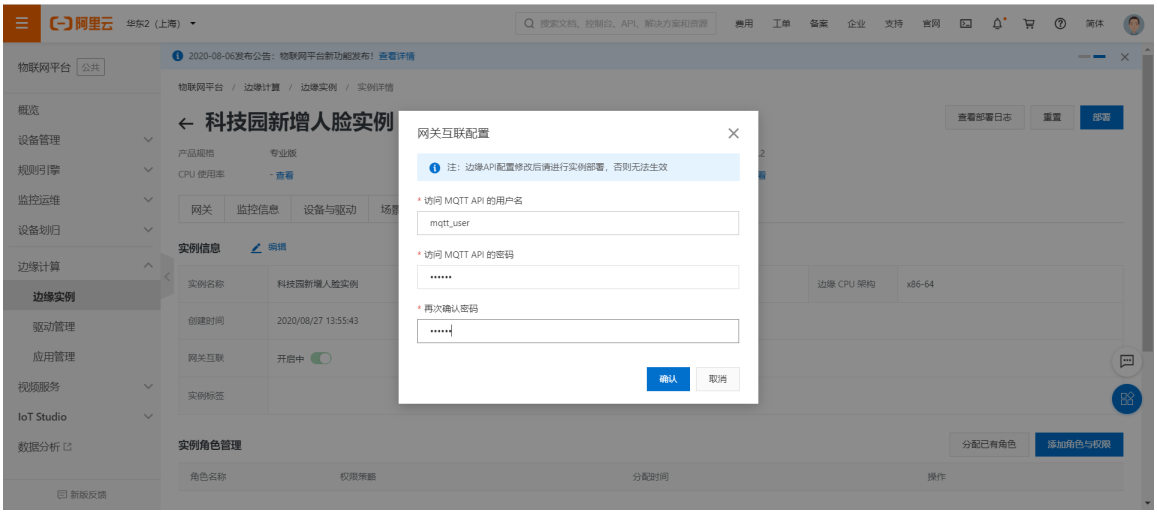
xi. 点击**实例**后，会看到边缘实例-网关为在线状态，代表该组件部署成功。



xii. 查看边缘设备接入组件状态里，点击**设置**，将网关互联开关打开。



xiii. 打开网关互联，需要输入用户名和密码。这个开关打开后，允许边缘应用订阅边缘系统通过MQTT发出的事件和告警。这个开关打开后，如果关闭后重新打开，将清除掉之前的用户名和密码，需要重新输入。**用户名**：固定输入：mqtt_user **密码**：固定输入：123456



xiv. 网关互联打开后，需要点击右上角的**部署**，该配置才能推送到边缘端生效，如下图所示。



3.3.3.2. 部署人行设备接入所需组件

人行设备接入，除了需要部署基础组件外，还需要部署云边协同组件、边缘通行服务组件、LinkVisual人脸识别服务组件。请按照下面说明进行部署安装。

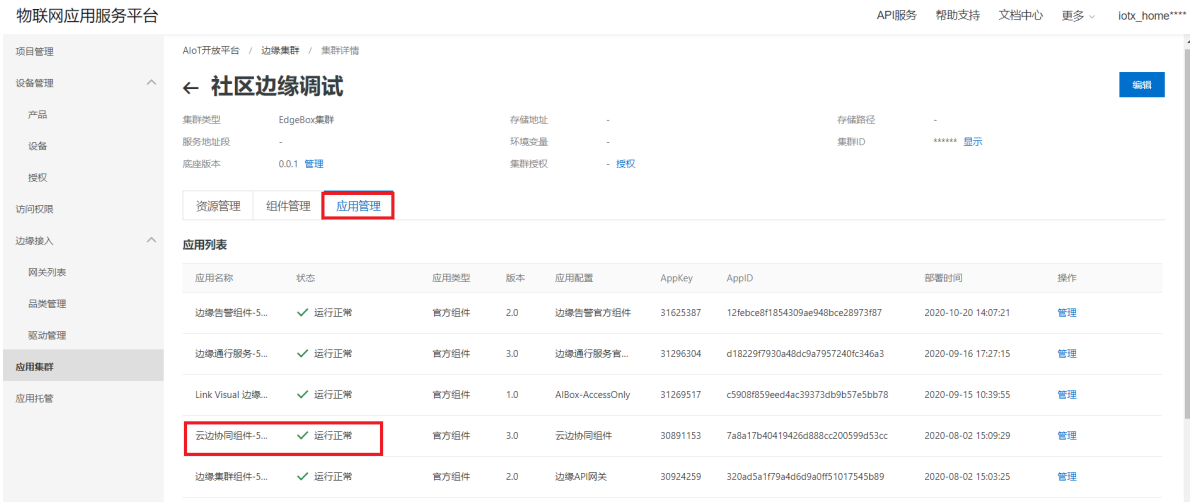
1. 选择**应用集群**，进入相应的集群，在**组件管理**里，选择**云边协同组件**的右侧**安装**，如所示。



2. 使用云边协同功能，账号是否由本地管理，一般选择否，如下图所示。



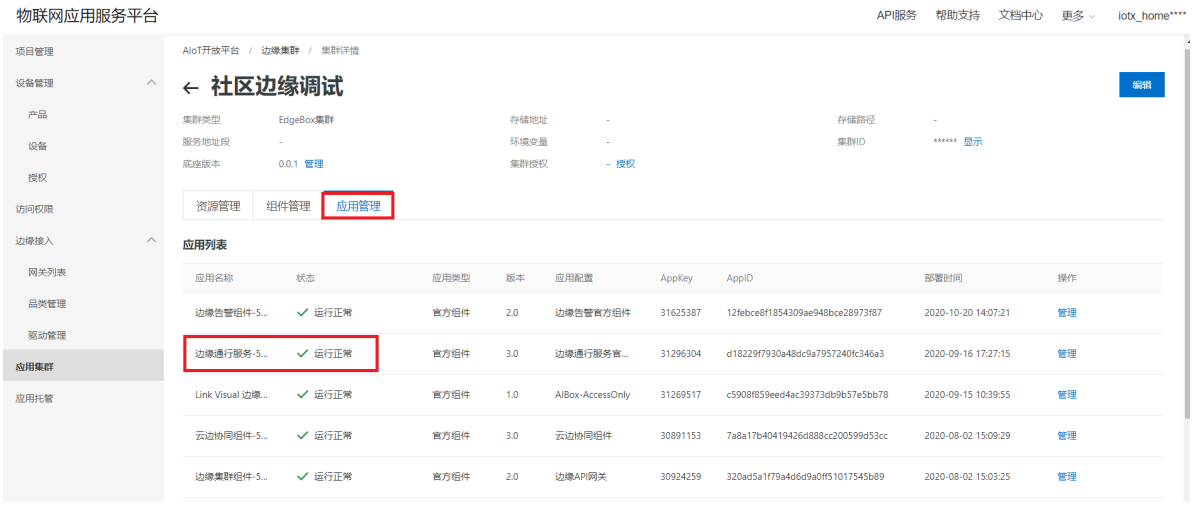
3. 当部署完成之后，可以进入该集群的**应用管理**，查看组件是否部署成功，如下图所示：



4. 选择**应用集群**，进入相应的集群，在**组件管理**里，选择**边缘通行服务**组件的右侧**安装**，如所示。



5. 当部署完成之后，可以进入该集群的**应用管理**，查看组件是否部署成功，如下图所示：

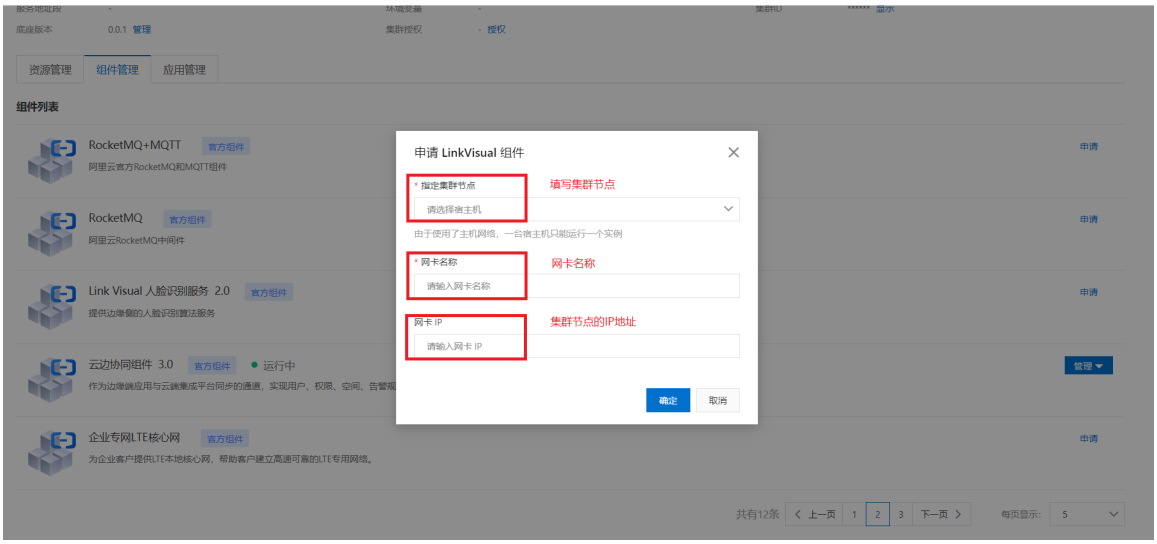


6. 项目里如果需要接入**人行的人脸识别**门禁设备，而且该设备的人脸识别方式是边侧识别，则需要部署Link Visual人脸识别服务组件。可以根据项目需要对接的人脸门禁设备品牌xxx，找到对应的<xxx 人行设备接入>手册，手册里会说明该设备的人脸识别方式是边侧识别还是端侧识别。

   i. 登录阿里云应用服务平台，登录成功后“边缘集群 - 组件管理- 部署组件”，找到**Link Visual 人脸识别服务**，点击右侧**申请**。



ii. 弹出下面对话框



名词解释	参数说明
集群节点	边缘服务器中节点的名称，选择某个节点代表该组件部署在这个节点中。（项目中一般一个集群只有一个服务节点，选择默认节点就好
网卡名称	边缘集群的运行节点所在网卡名称，可以在主机上查看，使用命令：ipconfig 查看网卡名称和IP。选择该集群节点所在的IP所对应的网卡名称
网卡IP	边缘集群运行的节点IP，可以在主机上查看，使用命令：ipconfig 查看网卡名称和IP。选择该集群节点所在的IP

iii. 网卡名称就是在1.1.3节网络配置里选择的网卡名称，如下图所示。如果选择的网络配置文件为**ifcfg-enp61s0f0**，网卡名称即为enp61s0f0.

```
[root@iot-it0ttg-node01 ~]# ls /etc/sysconfig/network-scripts/
ifcfg-enp61s0f0  ifcfg-lo      ifdown-isdn     ifdown-TeamPort  ifup-ippp     ifup-ppp       ifup-wireless
ifcfg-enp61s0f1  ifdown        ifdown-post     ifdown-tunnel    ifup-ipv6     ifup-routes    init.ipv6-global
ifcfg-enp95s0f0  ifdown-bnep   ifdown-ppp      ifup             ifup-isdn     ifup-sit       network-functions
ifcfg-enp95s0f1  ifdown-eth    ifdown-routes   ifup-aliases     ifup-plip     ifup-Team      network-functions-ipv6
ifcfg-enp95s0f2  ifdown-ippp   ifdown-sit      ifup-bnep        ifup-plusb    ifup-TeamPort  route-enp95s0f1
ifcfg-enp95s0f3  ifdown-ipv6   ifdown-Team     ifup-eth         ifup-post     ifup-tunnel
```

iv. 完成Link Visual 人脸识别服务组件的申请，请联系项目组请阿里小二后台进行审批。审批通过后，阿里小二会进行部署，当部署完成之后，可以进入该集群的**应用管理**，查看组件是否部署成功。

3.3.3.3. 部署车行设备接入所需组件

车行设备接入，除了需要部署1.4节里提到的基础组件外，还需要部署云边协同组件、边缘通行服务组件。如果在人行设备接入里已经部署过了这两个组件，请忽略。

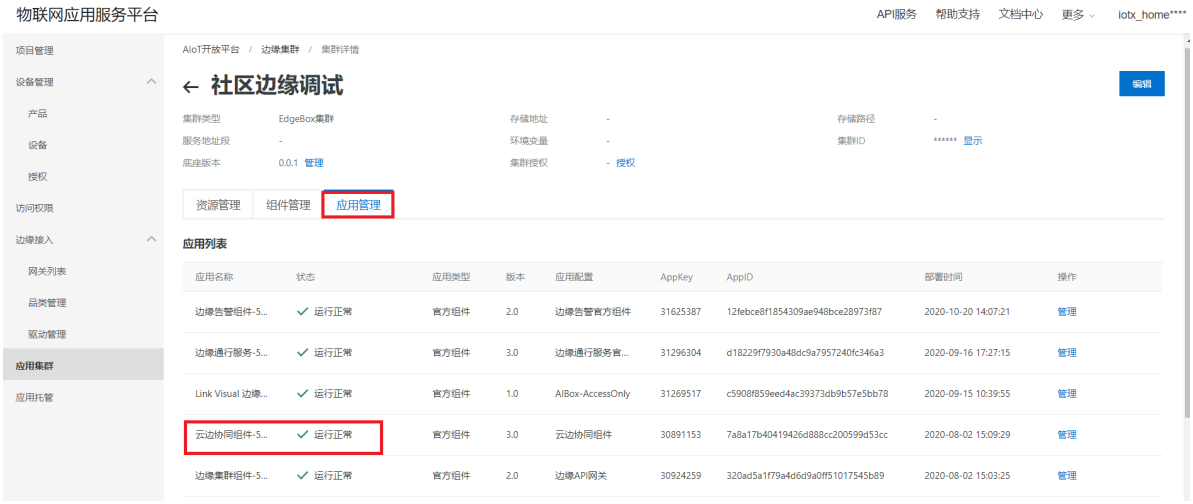
1. 选择**应用集群**，进入相应的集群，在**组件管理**里，选择**云边协同组件**的右侧**安装**，如所示。请注意：如果项目使用的是邦道车行设备，请选择**云边协同组件-车行版**组件进行安装。



2. 使用云边协调功能，账号是否由本地管理，视项目具体需求。是表示账号有本地产生和管理，否表示从云端同步SI账号到边缘侧。如下图所示。



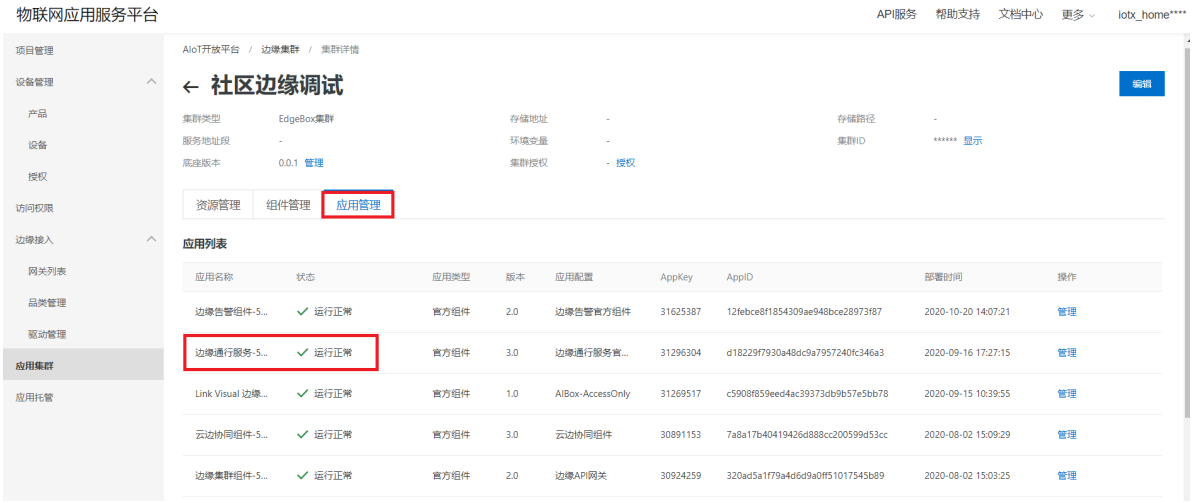
3. 当部署完成之后，可以进入该集群的**应用管理**，查看组件是否部署成功，如下图所示：



4. 选择**应用集群**，进入相应的集群，在**组件管理**里，选择**边缘通行服务**组件的右侧**安装**，如所示。请注意：如果项目使用的是邦道车行设备，请选择**边缘通行服务-车行版**组件进行安装。



5. 当部署完成之后，可以进入该集群的**应用管理**，查看组件是否部署成功，如下图所示：



3.3.3.4. 部署安防设备接入所需组件

安防设备接入，除了需要部署基础组件外，还需要部署LinkVisual边缘视频接入组件。

请按照下面说明进行部署安装。请特别注意，该实施手册只适用于物业一体机设备。如果是AIBOX设备，请参考对应的手册进行部署安装。项目里如果需要部署安防设备，例如IPC摄像头或者NVR设备，需要部署LinkVisual边缘视频接入组件。

1. 登录阿里云应用服务平台，进行下面操作：登录成功后，选择**应用集群**，进入相应的集群，在**组件管理**里，选择LinkVisual边缘视频的右侧**申请**，如图所示：

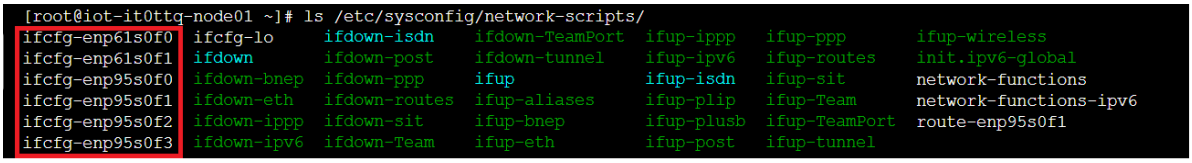


2. 弹出的申请页面如下图所示：

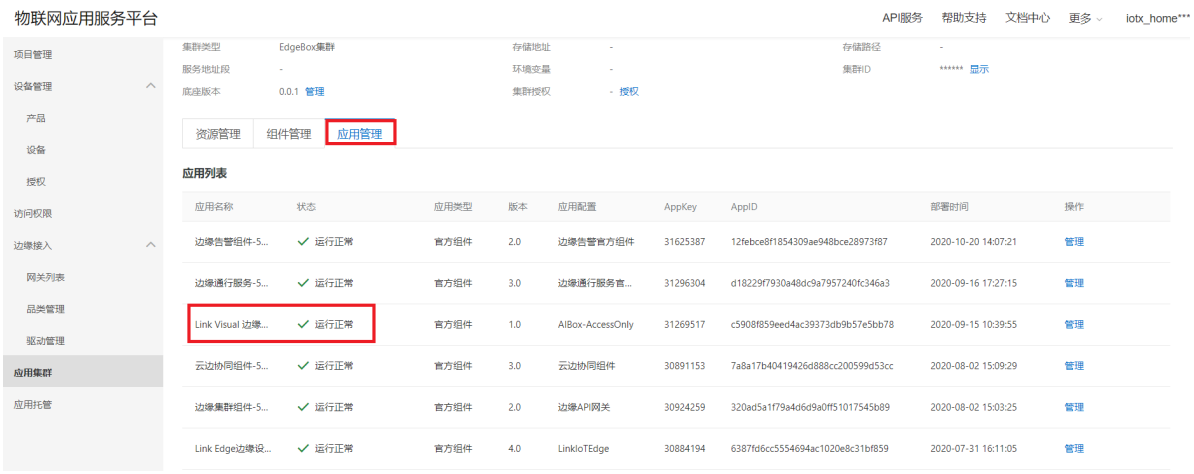


名词解释	参数说明
集群节点	边缘服务器中节点的名称，选择某个节点代表该组件部署在这个节点中。（一般一个集群只有一个服务节点，选择默认节点就好
网卡名称	是边缘集群的运行节点所在网卡名称，可以在主机上查看，使用命令：ipconfig 查看网卡名称和IP。选择该集群节点所在的IP所对应的网卡名称
网卡IP	边缘集群运行的节点IP，可以在主机上查看，使用命令：ipconfig 查看网卡名称和IP。选择该集群节点所在的IP
网关产品	不需要任何操作

3. 网卡名称就是在1.1.3节网络配置里选择的网卡名称，如下图所示。如果选择的网络配置文件为**ifcfg-enp61s0f0**，网卡名称即为enp61s0f0.



4. 完成Link Visual 边缘视频组件的申请，会有阿里小二后台进行审批。审批通过后，阿里小二会在后台进行部署。部署完成之后，可以进入该集群的**应用管理**，查看组件是否部署成功，如下图所示：

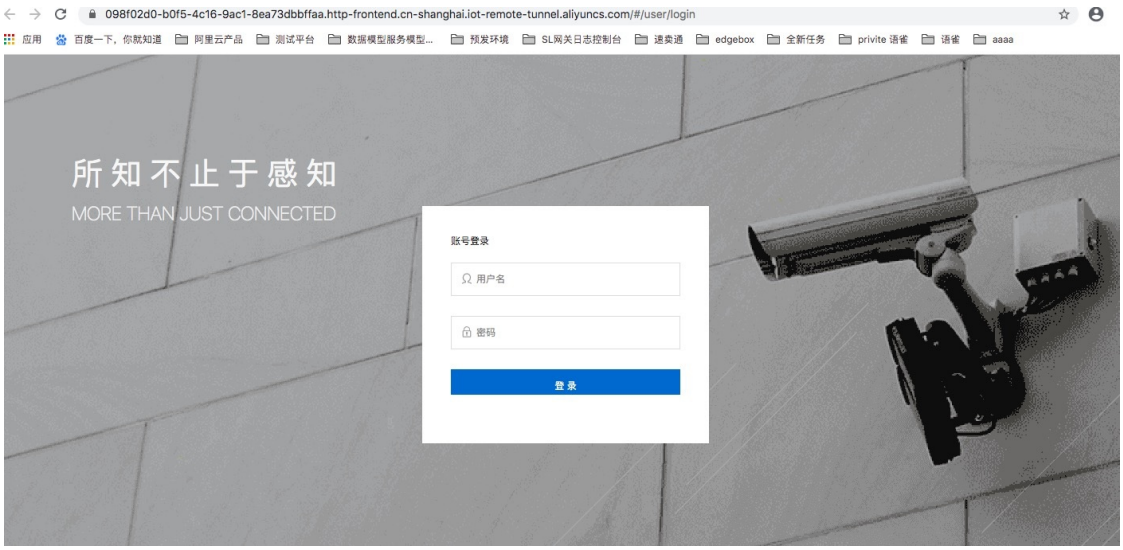


i. **访问电脑和边缘集群在同一个网段内**

a. 部署成功后，可以查看边缘视频组件的本地控制台页面。如下图所示。将IP和端口输入浏览器即可。



b. 下图为登录页面：用户名和密码都为admin和admin。



c. 登录成功页面如下图所示。到此处，Link Visual 边缘视频组件安装成功。



ii. **访问电脑和边缘集群不在同一个网段内**

a. 部署成功后，需要登录到应用服务平台，通过Link Edge边缘设备接入组件访问进行本地边缘视频组件的控制台。

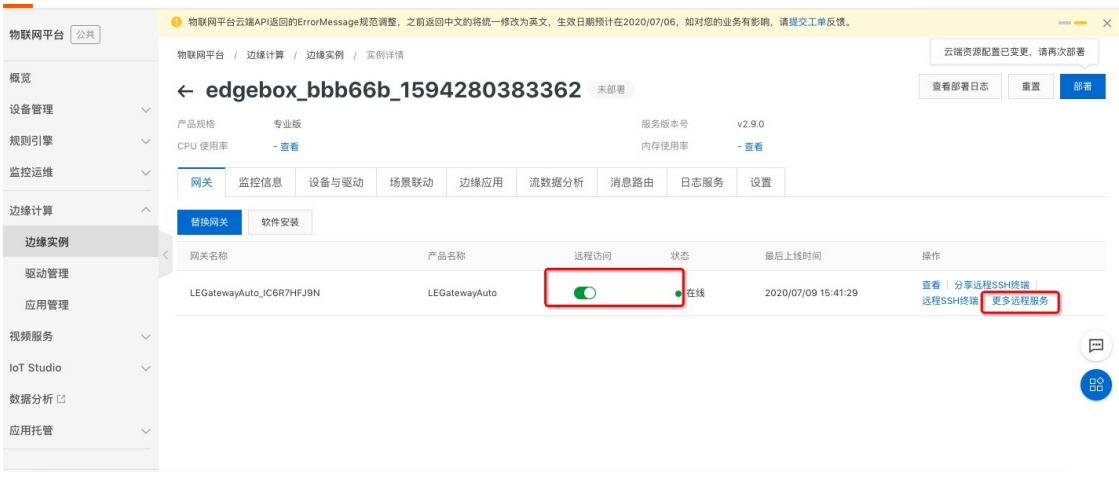
b. 点击Link Edge边缘设备接入组件的实例。如下图所示。



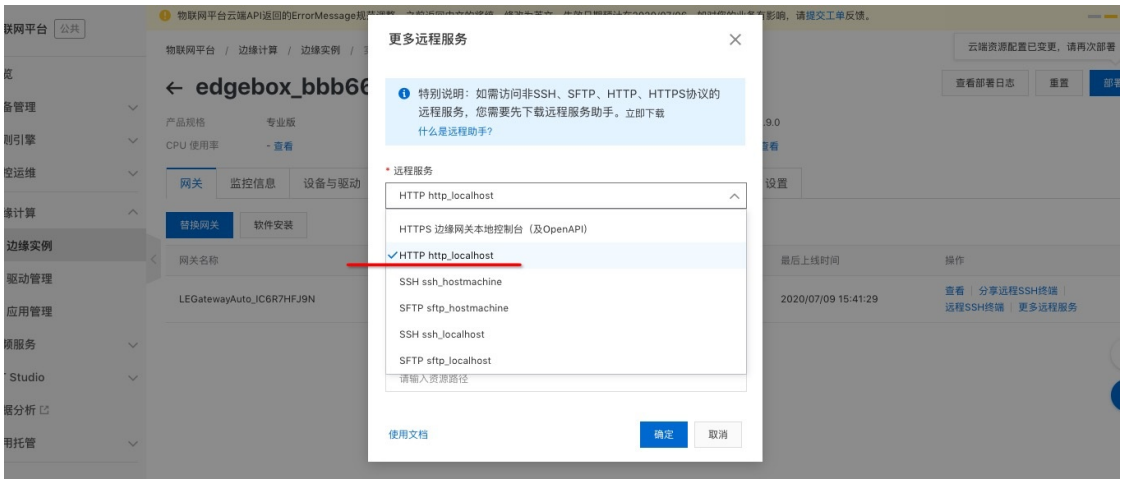
c. 跳转到物联网平台页面，点击开启远程访问。



d. 远程访问开启成功后。如下图所示。点击更多远程服务。



e. 远程服务协议选择：Http_http_localhsot 如下图所示。



f. 点击确定后，边缘视频组件的本地控制台页面。如下图所示。用户名和密码都为admin和admin。



g. 登录成功页面如下图所示。到此处，Link Visual 边缘视频组件安装成功。



3.3.3.5. 部署EBA设备接入所需组件

EBA设备接入，只有完成部署基础组件就可以了，不需要再额外部署其他组件

3.3.3.6. 部署边缘告警组件

边缘规则引擎官方组件是边缘告警组件的依赖组件，安装边缘告警组件之前需要先安装该组件。

> **警告**
>
> 边缘告警组件耗费的资源比较多。安装请与相关人员确定项目上是否使用到了边缘告警组件。安装顺序：必须先部署边缘规则引擎官方组件，然后再部署边缘告警组件

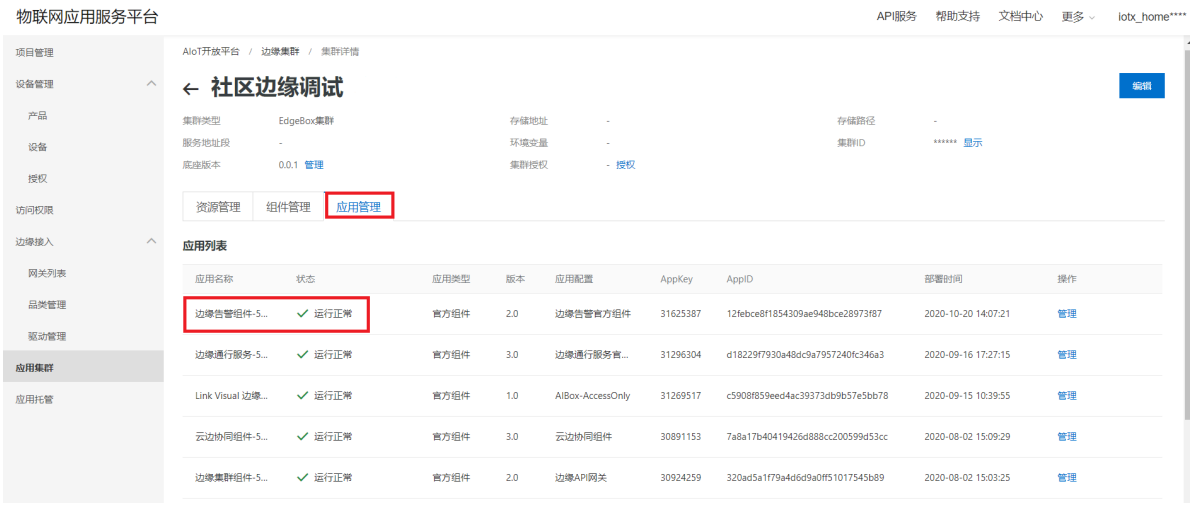
1. 选择**应用集群**，进入相应的集群，在**组件管理**里，选择边缘规则引擎官方组件右侧的**安装**，如下图所示



2. 当部署完成之后，可以进入该集群的**应用管理**，查看组件是否部署成功。选择**应用集群**，进入相应的集群，在**组件管理**里，选择边缘告警组件右侧的**安装**，如下图所示。



3. 当部署完成之后，可以进入该集群的**应用管理**，查看组件是否部署成功，如下图所示：



3.3.4. 一体机关联项目

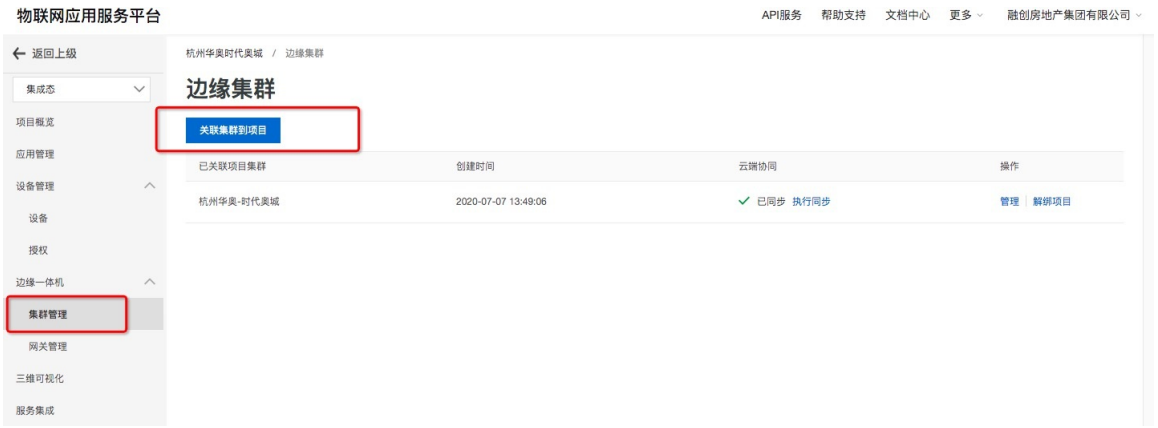
前面已完成了边缘一体机的整体组件部署。基于已完成的边缘一体机，需要进行项目和边缘集群的关联，实现项目对边缘集群内部资源的实时管控和数据获取

1. 关联项目，一体机的实施过程和相关数据授权需要和项目进行管理

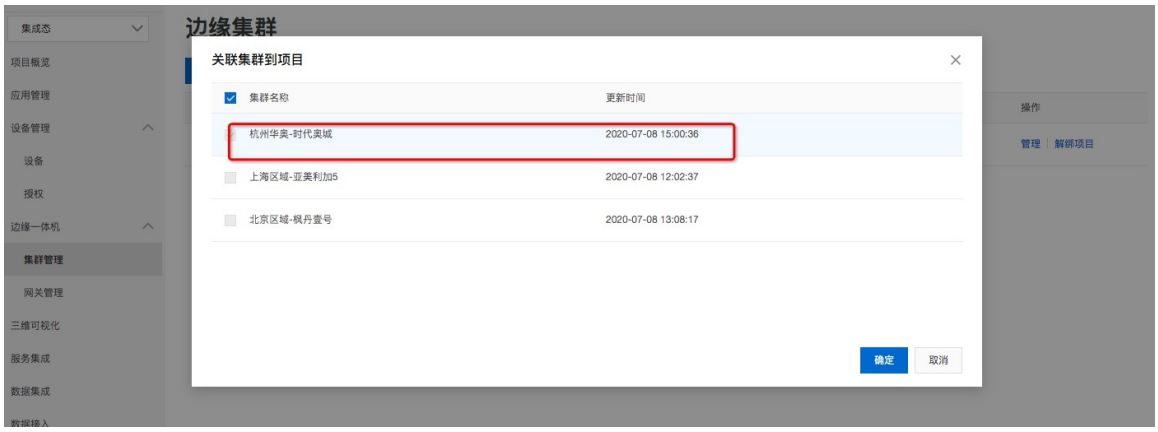
   i. 登录阿里云应用服务平台，找到需要配置的项目，该项目已由项目方创建，如果未创建，请联系项目方。如下图所示



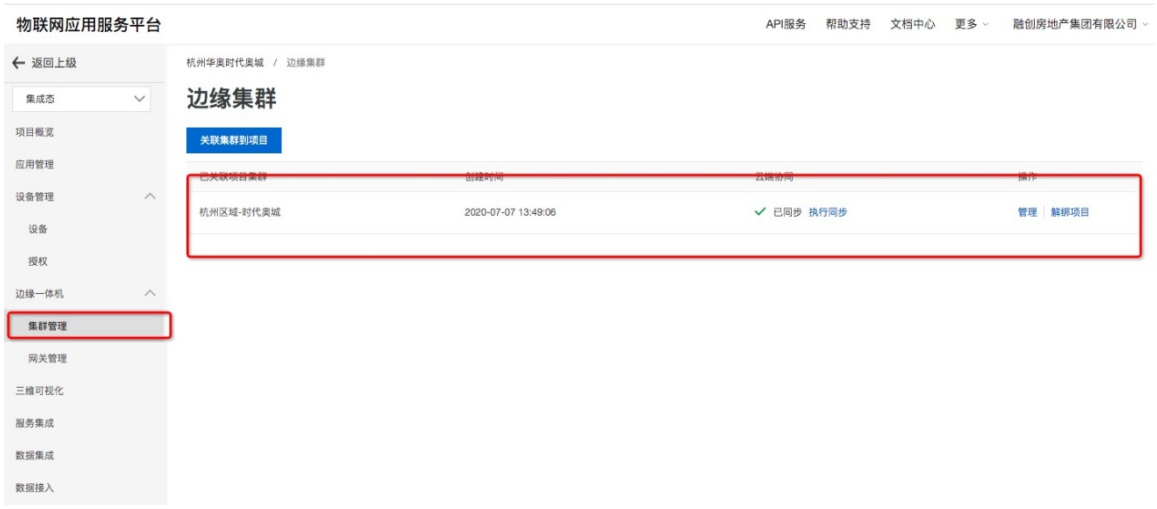
   ii. 进入项目后，找到**边缘一体机-集群管理**，把对应的集群关联到项目中。如下图所示



iii. 点击确认，即可完成边缘集群和项目的关联。

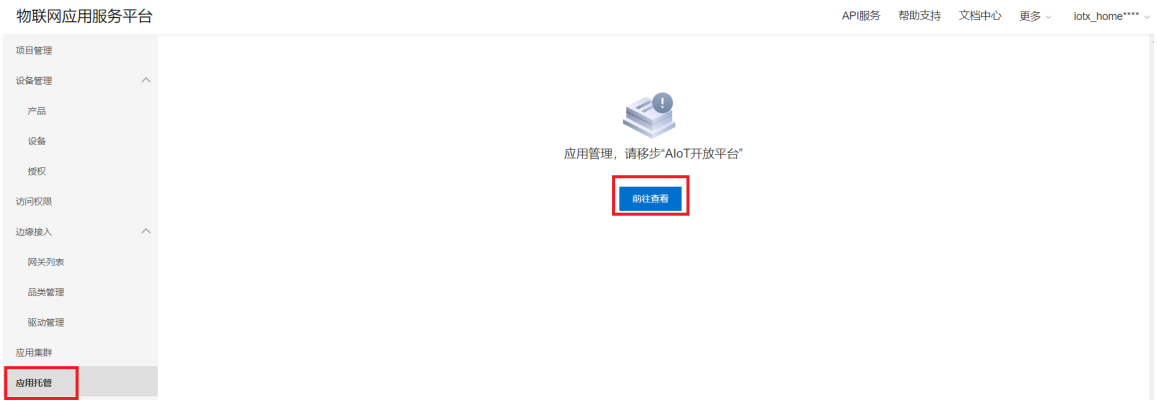


iv. 如下图所示，为关联成功展示页面：

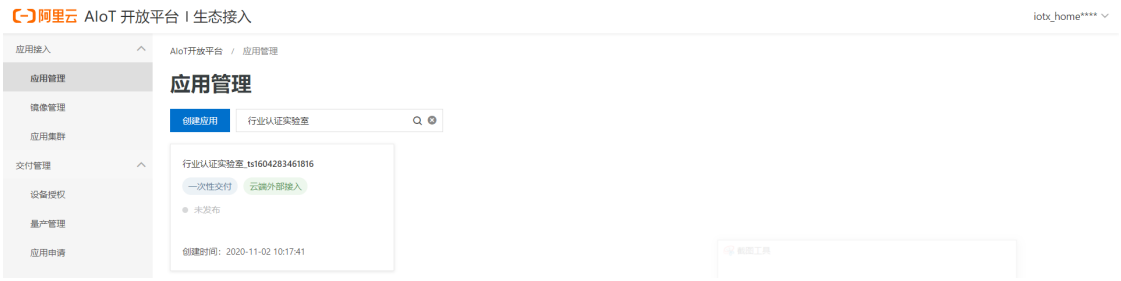


2. 授权访问人行所需数据模型，需要对项目默认应用进行授权，能够访问可视对讲通行记录和门禁通行事件记录的数据模型。

i. 打开项目默认应用，打开物联网应用服务平台，不要选择项目。点击**应用托管**，**前往查看**，如下图所示



a. 应用管理的搜索框里，输入项目名称，可以找到该项目的默认应用。默认应用的名称为项目名称+ts+13位数字。如下图所示，项目名称为行业认证试验室，对应的默认应用名称为行业认证实验室_ts1604283461816。



b. 点击打开该默认应用。请特别注意：按照上述说明，打开该项目的默认应用，而不是其他应用

ii. 添加数据模型，iot_entrance_videointercom_record_model

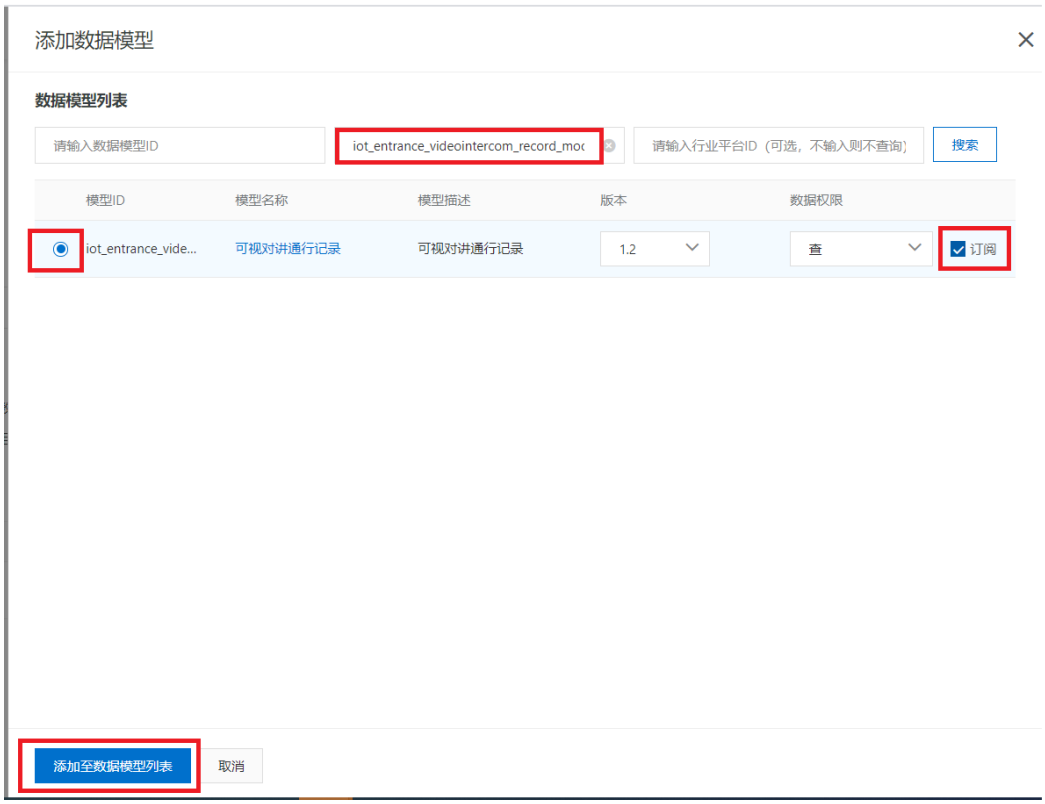
a. 在打开的默认应用，点击**版本管理-模型与权限**，如下图所示



b. 点击**数据模型-添加数据模型**，如下图所示：

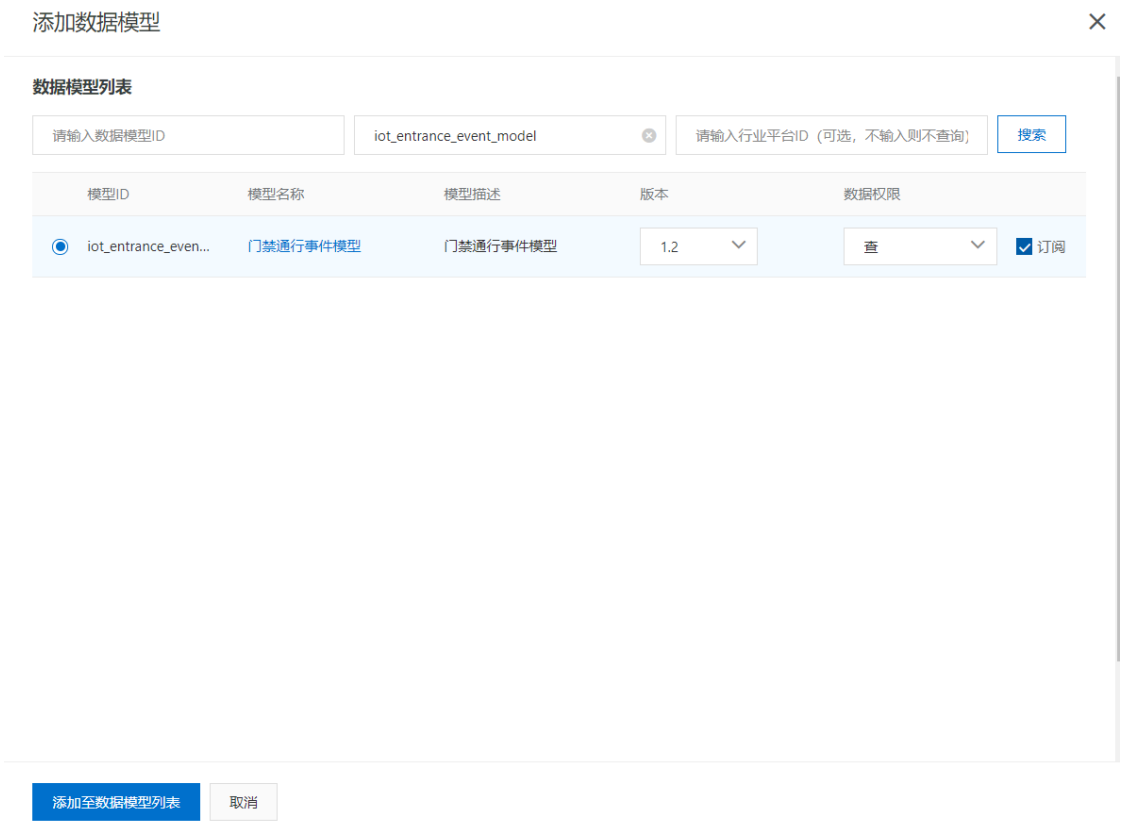


c. 在数据模型名称搜索框里，输入iot_entrance_videointercom_record_model，然后在搜索到的数据模型，点击左侧选中按钮，**并勾选右侧订阅。**最后点击下方的**添加至数据模型列表**，如下图所示。



iii. 添加数据模型iot_entrance_event_model

a. 在数据模型名称搜索框里，输入iot_entrance_event_model，选中后勾选订阅，最后**添加至数据模型列表**



3. 版本发布

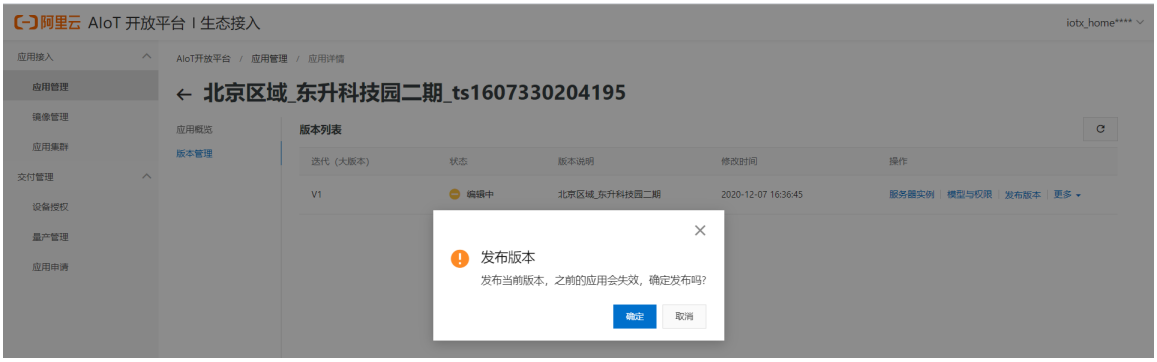
i. 数据模型添加完成后，点击模型和权限的**左侧箭头**，如下图所示：



ii. 点击该项目默认应用的**版本管理**，点击右侧**发布版本**，如下图所示：



iii. 弹出的对话框里，点击**确认**



4. 项目授权

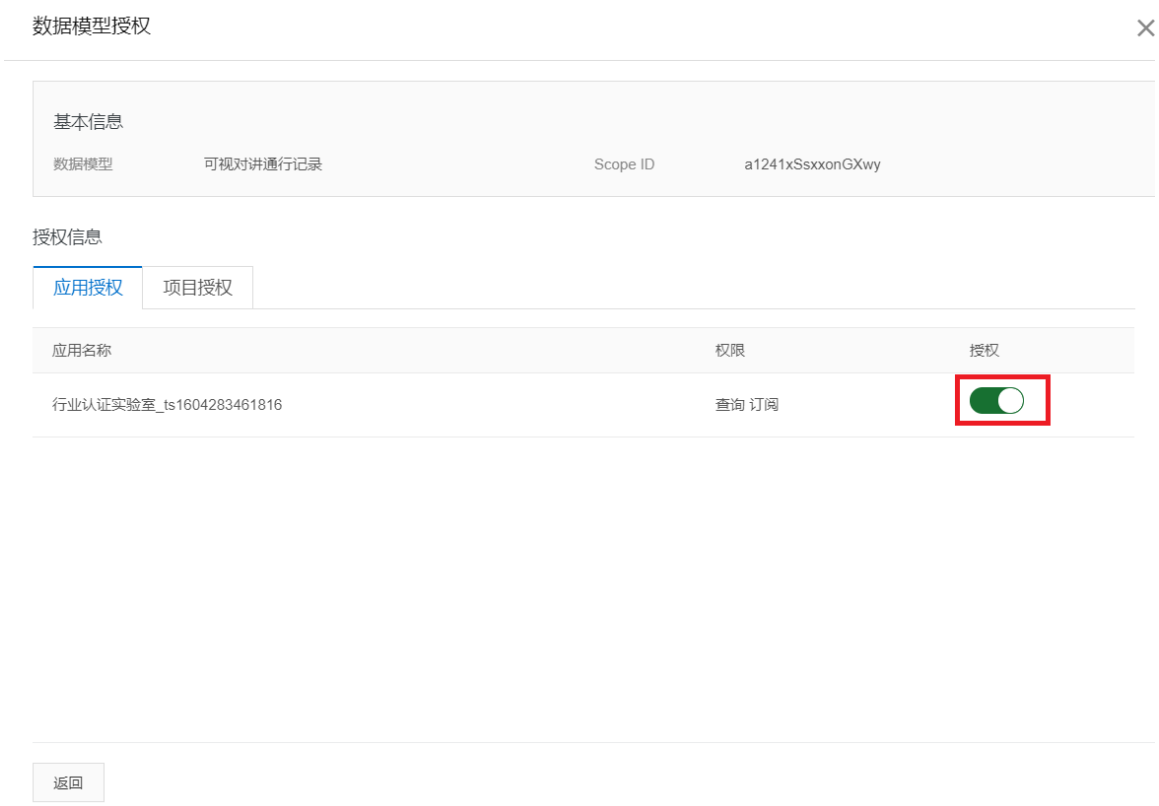
i. 打开对应的项目，如下图所示：



ii. 在**集成态**下，选择**数据集成**，找到**可视对讲通行记录**，点击右侧**授权**，如下图所示：



iii. 在弹出的页面里，将点击打开授权按钮，如下图所示



iv. 类似操作，再将**门禁通行事件模型**的授权按钮打开，如下图所示



3.3.5. 设备接入配置

3.3.5.1. 接入设备品牌列表

前言

本文提供阿里云IoT物业管理一体机产品可接入的本地智能设备的品牌等具体信息

物业管理一体机支持的人行设备

品牌	型号	固件版本	对接类型	对接功能

艾特智能	MK101 MK102 MK080	1.1.35	设备	人脸 二维码 刷卡 可视对讲
安居宝	AJB-ZJ15CCR(AM)IP AJB-ZJ18AAC(AM)IP AJB-ZJ15CAC(AM)IP	N/A	设备	人脸、二维码
大华	DH-ASI7213Y-V3(BIAN) DH-ASI7214Y-V3(BIAN) DH-ASI7213X(BIAN) DH-ASI7214X(BIAN) DH-ASI7223X-A(BIAN)	N/A	设备	人脸(边侧识别) 二维码 刷卡
	DH-VTO65 DH-VTO75 DH-VTO95	N/A	设备	人脸 二维码 刷卡 可视对讲
	DH-SH-MB9833Z-F	N/A	设备	人脸（边侧识别） 二维码
	DH-ASI7213Y-V3(DUAN) DH-ASI7214Y-V3(DUAN) DH-ASI7213X(DUAN) DH-ASI7214X(DUAN) DH-ASI7223X-A(DUAN)	N/A	设备	人脸（端侧识别） 二维码 刷卡
狄耐克	902D 905D 904D	N/A	设备	人脸 二维码 刷卡 可视对讲
	280D	N/A	设备	刷卡、蓝牙
	280AC	N/A	设备	二维码、刷卡
富士	FJC-MFT211-GXH FJC-MFT411-GXH FJC-MFT811-GXH FJC-MG616-GXH FJC-MG612-GXH FJC-DW3220-GXH FJC-ZRS832-GXH FJC-ZRS672-GXH FJC-ZRS631-GXH FJC-ZRS632-GXH FJC-ZR691S/R FJC-ZR692S/R FJC-ZRS694MCB FJC-ZRS693TB FJC-ZRS694TB	N/A	设备	人脸

冠林	GL8-K3CS GL8-T3CS GL8-K4CS	01.03.0001	设备	人脸 二维码 刷卡 可视对讲
海康威视	DS-KD9613-JRG DS-KD9613-JRW DS-K5604-Z(H) DS-K1T671M/QR DS-K1T671TXJR DS-KD9613-AFB/ZSJ	N/A	设备	人脸（端侧识别） 二维码 刷卡 蓝牙 可视对讲
华安视讯	HAS-D302H-PA07-B0	N/A	设备	人脸(支持测温）
慧锐通	BKM-801S7	4.14	设备	人脸、二维码
	B2M-801S5	9.00.07		人脸、二维码、可视对讲
霍尼韦尔	HLP-6050F	N/A	设备	人脸、刷卡
捷顺	JSMJY10A-Ti	1.0.0	设备	人脸 刷卡 二维码
立方	RF-FX300-H823TC	N/A	设备	刷卡、人脸
	RF-FX300-H8215S RF-FX300-H8215ST	N/A	设备	人脸 二维码(远程开门)
立林	EH-OS-M10 EH-OS-M15 EH-OS-M17 EH-OS-M19 EH-OS-M22 EH-OS-M26 EH-OS-M28 EH-OS-M29 EH-OS-M31 EH-OS-M10-T01 EH-OS-M17-T01T EH-OS-M17-T02 EH-OS-M17-T03 EH-OS-M17-T04 EH-OS-M19-T01T EH-OS-M19-T02T EH-OS-M19-T03T EH-OS-M19-T04T EH-OS-M19-T01 EH-OS-M19-T02 EH-OS-M19-T03 EH-OS-M19-T04	N/A	设备	人脸 二维码 刷卡 蓝牙 可视对讲
	EH-OS-M19-T31T EH-OS-M20-T01 EH-OS-M21-T11 EH-OS-M21-T31T EH-OS-M22-T02T EH-OS-M22-T02 EH-OS-M25-T01	N/A	设备	人脸 二维码 刷卡 可视对讲

罗格朗	Legrand 693065 Legrand 693045 Legrand 693064 Legrand 693044 Legrand 693063 Legrand 693043 Legrand 693165 Legrand 693145 Legrand 693164 Legrand 693144 Legrand 693163 Legrand 693143 Legrand 693265 Legrand 693245 Legrand 693264 Legrand 693244 Legrand 693263 Legrand 693243 Legrand 693213 Legrand 693362 Legrand 693342 Legrand 693015 Shidean KE6017T4 Shidean KE6017T3 Shidean KE6107T4 Shidean KE6107T3 Shidean KE62474 Shidean KE62473 Shidean KEP42473 Legrand 693015	N/A	设备	人脸 二维码 刷卡 蓝牙 可视对讲
	Legrand 693061 Legrand 693041 Legrand 693161 Legrand 693141 Legrand 693261 Legrand 693241 Legrand 693461 Legrand 693441 Shidean KE6017T1 Shidean KE6107T1 Shidean KE62471 Shidean KEP43671 Shidean KEP41671	N/A	设备	二维码 刷卡 蓝牙 可视对讲
	Legrand 693460 Legrand 693440 Legrand 693410 Shidean KEP41670	N/A	设备	二维码 蓝牙 可视对讲
太川	TC-5000D-H	N/A	设备	人脸 二维码 刷卡 可视对讲
	TC-5000D-B1/WD	N/A	设备	人脸（支持测温） 二维码
泰创	V/N WD01NM-CBU	N/A	设备	人脸、二维码
	PU-WD01NM-CBU		设备	人脸(端侧) 二维码

通力	DID-101	N/A	系统	人脸、二维码
悉尔	A18010	N/A	设备	人脸、二维码
星网锐捷	DH-T98 DH-T101	N/A	设备	人脸 二维码 刷卡 可视对讲
中证科技	H-OS06	H-OS06_DM_V3.01	设备	人脸 二维码 刷卡 可视对讲
宇视	ET-S51H@B ET-S32L@W	N/A	设备	人脸、二维码
	ACS-UAR321L	N/A	设备	人脸
途途数智	TT-ECQ5S	v1.0	设备	人脸、二维码、电梯规则网关
旺龙	ITL-YDT	ITL-YDT0110(V1.3)	设备	刷卡、二维码、 电梯规则网关
	ITL-FA06D-CQS	3288XT V1.2	设备	刷卡、二维码、 电梯规则网关

物业管理一体机支持的车行品牌

品牌	硬件型号	软件版本	对接类型
富士	摄像头：FJC-CPA8201AC	ice_ipcsdk_linux_x64_V1.0.0.62 对应一体机T33	硬件
	摄像头：FJC-CPA6201AC		
	摄像头：一体机FJC_TA026L 外挂的摄像头FJC_CPA9202BA		
	摄像头：FJC-CPA9202AA		
	摄像头：FJC-CPA8202AA		
	摄像头：FJC-CPA6201AA		
	摄像头：FJC-CPA1201A		
	摄像头：FJC-CPA8201AA		
大华	DH-DSS-H710S2		停车系统3.0
蓝卡		所有系统版本	停车系统3.0
立方		系统版本：acs4.2.0	停车系统3.0

物业管理一体机支持的周界报警主机品牌

品牌	硬件型号	对接类型
豪恩	LHB9000	设备
广拓	A7全系列报警主机MK200I	设备
精华隆	EP3164-IP	设备
兰星	LX-M18C	设备

物业管理一体机支持的EBA品牌

品牌	硬件型号	对接类型	协议
能效通	数据栈全部型号包括子设备：超声波流量计 智能电力仪表 压力传感器 液位传感器 温湿度传感器 水浸传感器 温度检测 遥信遥控智能仪表	网关	Modbus
江森自控	气体感测器 风机 照明回路	DDC	BACnet IP
	CN-BRTR-0/CH-PCA2513-0/16DI, BCM/EM/CO1B2000/GW包括子设备：BCPro BACnet路由器 BACnet高级应用可编程控制器 BACnet I/O扩展模块 浮球式液位开关 室内一氧化碳浓度探测器 压差开关	网关	BACnet IP

物业管理一体机支持的梯控品牌

品牌	硬件型号	对接类型	对接功能
狄耐克	905K-Y3	梯控网关	上报电梯运行状态 查询电梯运行状态 释放受控楼层 点亮目标楼层 呼叫电梯
	Y3010	梯控网关	上报电梯运行状态 查询电梯运行状态 释放受控楼层 点亮目标楼层 呼叫电梯
立林	EH-MD-K11-002	梯控网关	上报电梯运行状态 查询电梯运行状态 释放受控楼层 点亮目标楼层 呼叫电梯
西奥	XOA-XCEC01A	梯控网关	上报电梯运行状态 查询电梯运行状态 释放受控楼层 点亮目标楼层 呼叫电梯
蒂森	XMT-FACE9-Ali	梯控网关	
捷顺	JSMJK02-TK16-N	梯控网关	
太川	TC-5000D-XL	梯控网关	

物业管理一体机支持的室内安防品牌

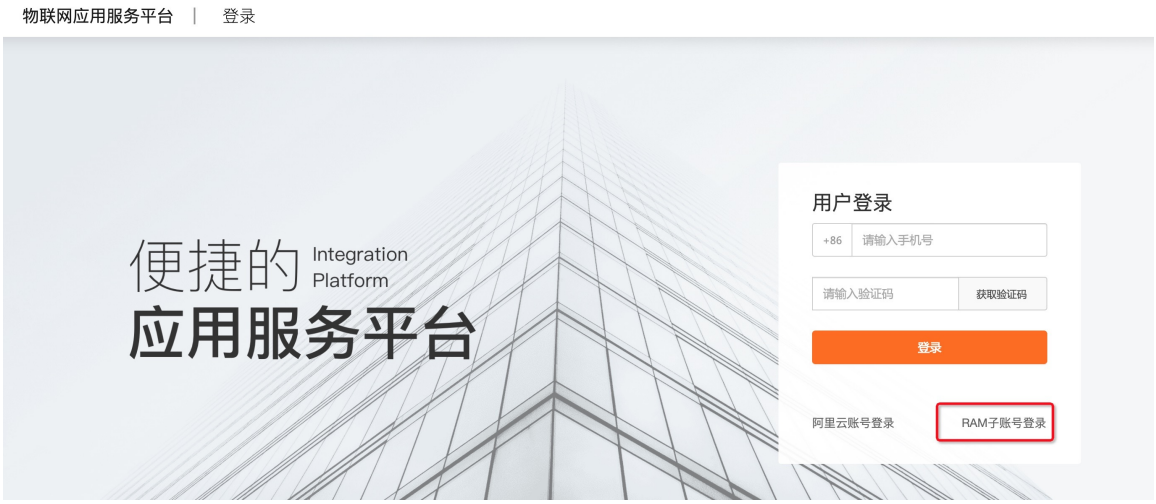
品牌	硬件型号	对接类型	设备类型
冠林	GL8-F101M	设备	SOS紧急按钮
中证科技	A-IS01	设备	SOS紧急按钮红外探测器

3.3.5.2. 通用实施说明

本章将介绍所有设备接入实施入口，请阅读具体品牌设备实施手册前完成本章的操作

> 说明
>
> 接入设备包括下面类型：
>
> 人行设备：各种门禁设备，包括人脸，二维码，刷卡，蓝牙，可视对讲设备
>
> 车行设备：包括车辆道闸，费显屏，车牌识别摄像头设备
>
> 安防设备：包括网络摄像机(IPC)和网络视频录像机(NVR)设备
>
> EBA设备：楼宇自动化用到的各种信息点位采集的设备
>
> 梯控设备：可以控制电梯运行的梯控模块设备

1. 登录应用服务平台，阿里云RAM子账号，还有智能施工人员OA账号都可以登录阿里云应用服务平台。通过手机号码登录的是OA账号。这两类账号都可以进行下面的部署实施操作，只是登录方式有不同。
   i. 点击阿里云应用服务平台地址链接，展示登录界面，选择RAM子账号登录。



   ii. 输入账号名和密码



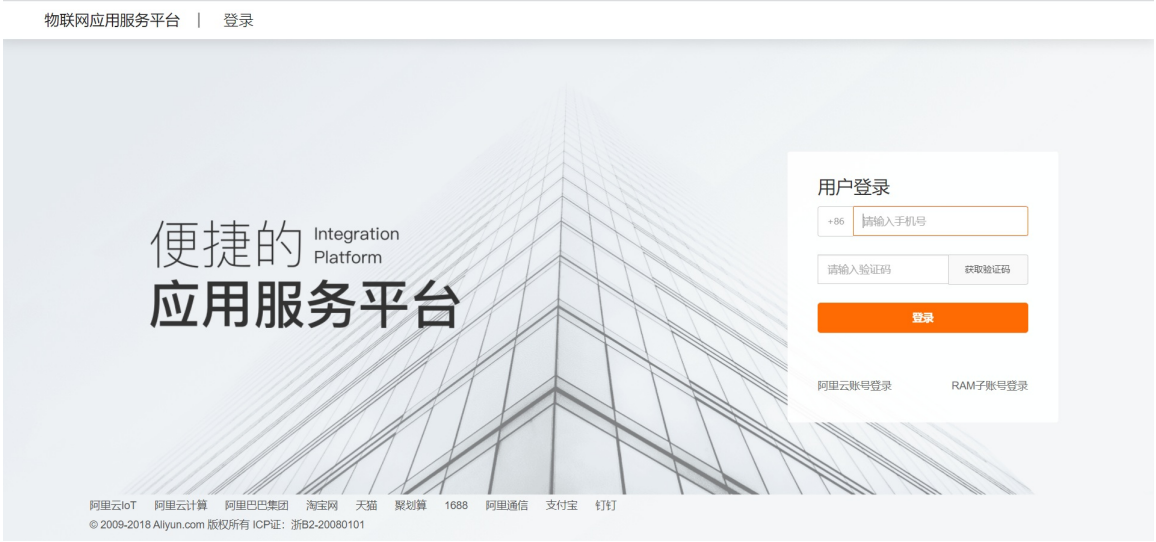
iii. 如果登录成功之后，可以看到如下界面，找到并点击需要部署的项目。该项目已由项目方创建，如果未创建，请联系项目方。



iv. 进入项目内，找到**边缘一体机-网关管理**，如下图所示。点击网关对应的**设备集成**



v. 智能施工人员通过手机短信验证码登录物联网应用服务平台，如下图所示：



vi. 如果登录成功之后，可以显示出所有被授权的项目列表如下图所示。如果没有找到相应项目，请联系项目方进行授权。



vii. 选择一个需要安装部署的项目，点击进入，显示页面如下



viii. 进入项目内，找到**边缘一体机-网关管理**，如下图所示。点击网关对应的**设备集成**

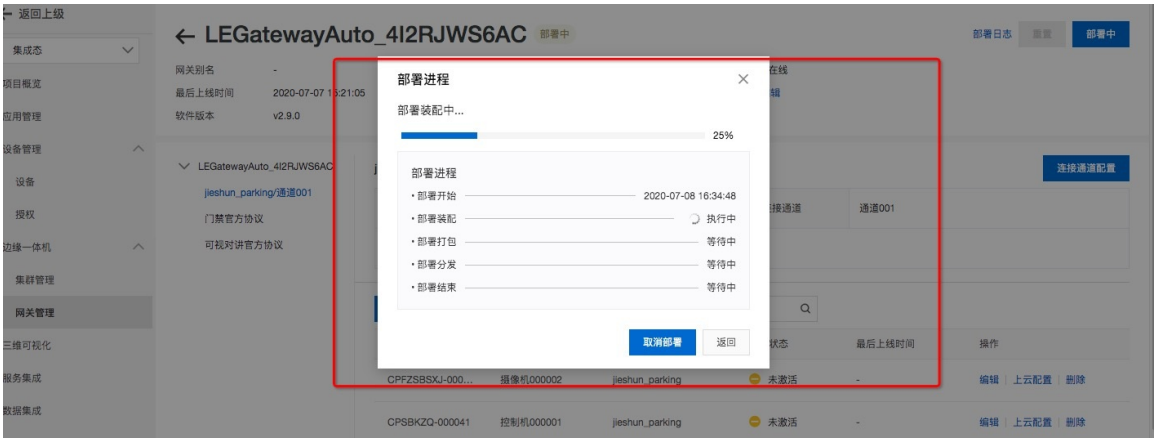


2. 设备配置实施，请参照各品牌文档进行实施，注意各品牌已接入的型号存在差异，对接前请仔细核对
3. 部署设备配置

i. 在完成设备的配置后，点击部署，进行设备配置的协议下发到边缘服务器。如下图所示。特别提醒：驱动部署时，网关侧对应驱动将重启，该驱动下的所有设备将出现离线后在线。这期间设备不可用，请部署人员提前与甲方沟通部署时间



ii. 点击部署后可以查看设备配置下发的状态和部署进度。



说明：如果某个设备有新更新的配置信息，重新点击部署后，只会下发有改动的设备配置协议，其余未修改的配置信息不会再次下发到边缘服务器，部署的时间根据修改的设备数量有关。

4. 设备联通性验证，**在进行设备联通性验证前，需要首先查验厂商的设备状态**

i. 厂商设备的软件版本已经升级到完成与阿里物业一体机对接过的版本。

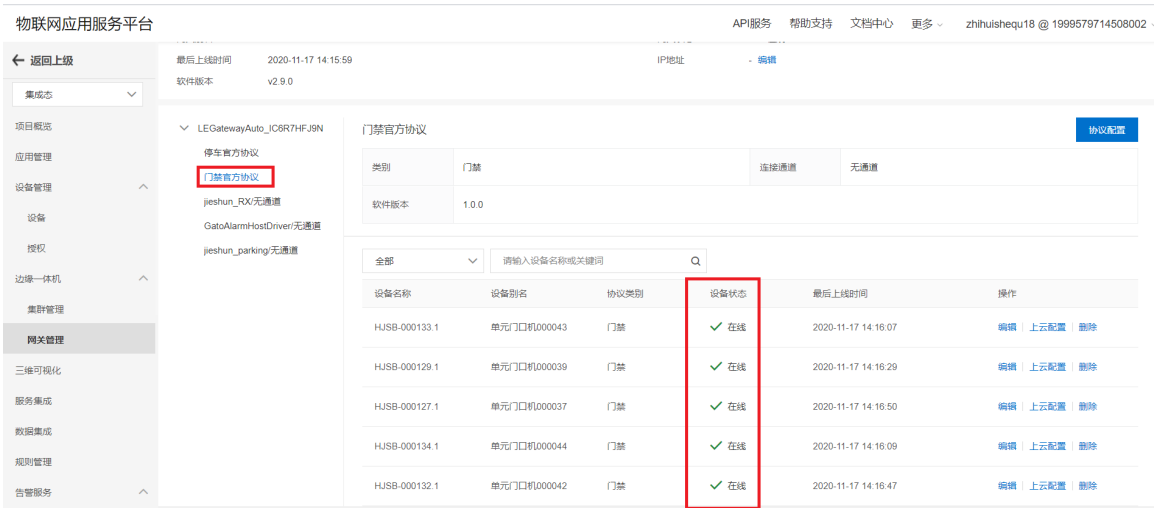
ii. 厂商的设备配置正确，例如配置了物业一体机的IP地址和端口号。通过设备对接方式接入物业一体机，端口号是否配置错误；

iii. 厂商的设备IP地址设置正确，物业一体机可以正常ping通厂商设备

5. 设备添加并部署完成后，可以点击**边缘一体机-网关管理-门禁官方协议**，逐个查看上面添加的所有设备状态，验证设备的联通性，如下图所示。如果状态是**在线**，则说明设备成功与边缘一体机连接成功；如果是未激活，说明设备与边缘一体机没有连接成功，这时可以排查的点：

i. (1) 查看设备的配置是否填写正确。如果存在问题，可以修正后重新部署

ii. (2) 可以远程登录物业一体机，尝试ping设备IP地址。如果无法ping通，说明设备IP地址有问题，或者内网存在问题，需要找项目的弱电施工方解决；如果可以ping通，需要找技术同学进一步排查解决



3.3.5.3. 人行设备接入

3.3.5.3.1. 立林人行设备接入

本文用于指导在门禁设备官方协议下配置立林人行设备。

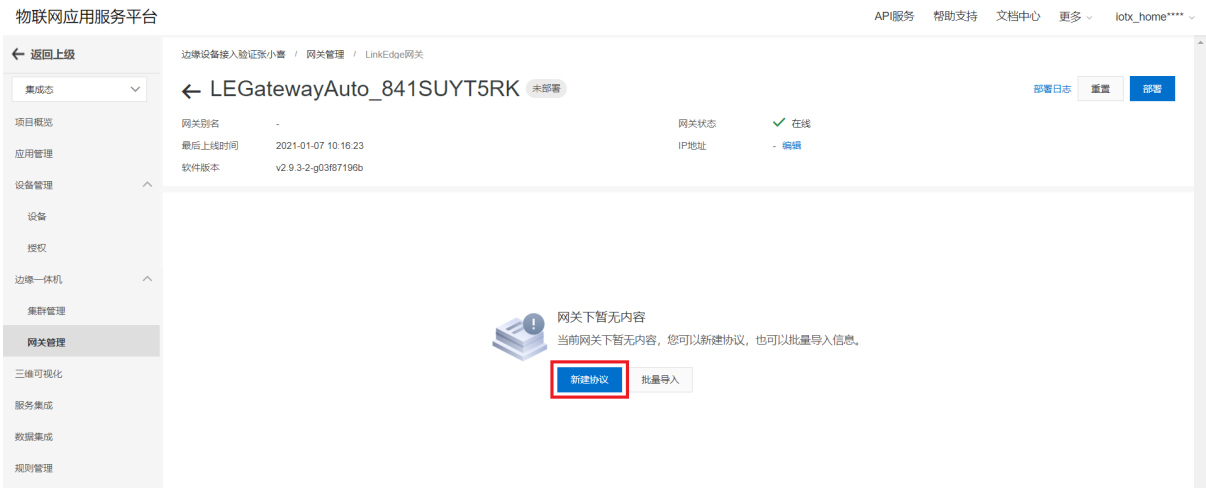
说明

支持对接的型号

对接型号支持	EH-OS-M19（边侧人脸比对） EH-OS-M19D（端侧人脸比对） EH-OS-M20
接入模式	门禁设备
支持功能	人脸/二维码/刷卡/蓝牙/可视对讲
系统版本	N/A
人脸比对方式	边侧比对/端侧比对（取决于设备型号）
门禁端口号	17725
可视对讲端口号	10060

一、门禁协议添加

- 第一次添加协议，点击 **新建协议** 按钮，如下图所示：



- 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击 **添加协议** 按钮，如下图所示：



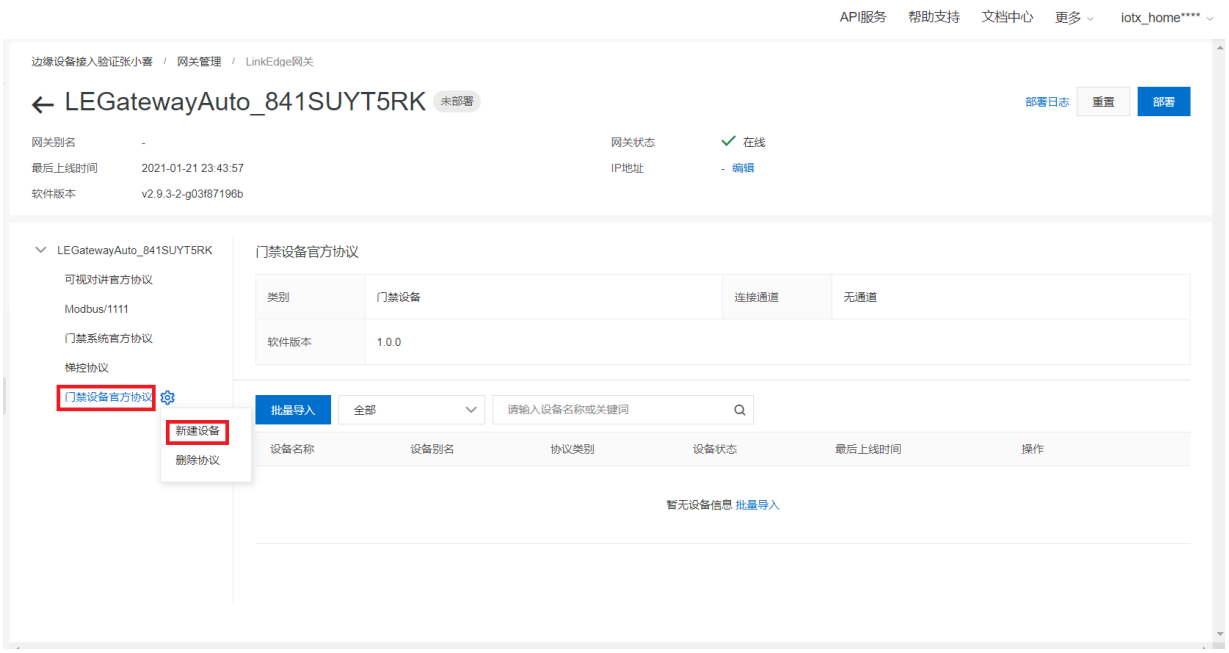
- 立林的接入模式是设备接入，在弹出的对话框里，选择 **门禁设备官方协议** 选项，如下图所示：



二、门禁设备集成

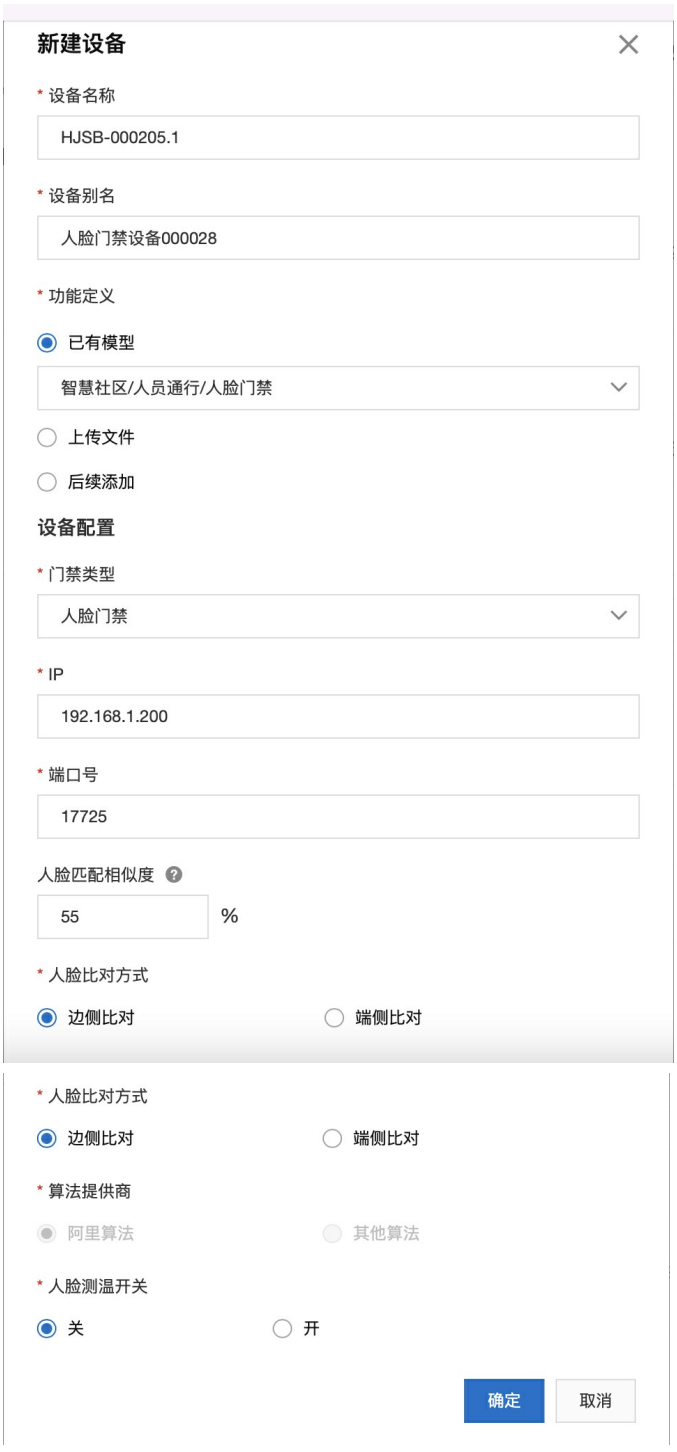
在门禁设备官方协议下添加新设备

鼠标移动到到刚添加的**门禁设备官方协议**，点击 **新建设备**，如下图所示：



新建“边侧比对人脸门禁设备”

1. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：**HJSB-000205.1** 其中"HJSB-000205"是该设备的IoT编码，".1"标识人脸门禁设备；
2. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028；
3. **功能定义**：选择已有模型，对应设备名称选择 “智慧社区/人员通行/人脸门禁”
4. **门禁类型**：选择“**人脸门禁**”
5. **IP**：参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为 192.168.1.200
6. **端口号**：**立林**门禁设备对外提供的网络访问端口号固定是 **17725**
7. **人脸匹配相似度**：输入**55%**。
8. **人脸比对方式**：**边侧比对**
9. **人脸测温开关**：**立林**门禁设备不支持人脸测温，设置为关



新建“端侧比对人脸门禁设备”

1. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：**HJSB-000205.1** 其中"**HJSB-000205**"是该设备的IoT编码，"**.1**"标识人脸门禁设备；
2. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028；
3. **功能定义**：选择已有模型，对应设备名称选择 “智慧社区/人员通行/人脸门禁”
4. **门禁类型**：选择“**人脸门禁**”
5. **IP**：参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为 192.168.1.200
6. **端口号**：**立林**门禁设备对外提供的网络访问端口号固定是 **17725**
7. **人脸匹配相似度**：输入**55%**。
8. **人脸比对方式**：**端侧比对**

9. **算法提供商**：后台自动选定为其它算法。
10. **人脸比对算法服务地址IP**： 实施人员需要根据实际情况填充实际可用的人脸比对算法服务所在的设备 IP，原则为局域网内可连通的任一可用人脸比对算法所在的立林门禁设备IP，图中为示例IP： 192.168.1.200
11. **人脸比对算法服务地址端口号**： 实施人员需要根据实际情况填充实际可用的人脸比对算法服务所在的端口号，这里为示例端口： **17726**
12. **人脸测温开关**：**立林**门禁设备不支持人脸测温，设置为关





新建“二维码门禁设备”

1. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备
2. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001
3. **功能定义**：选择已有模型，对应设备名称选择 “智慧社区/人员通行/二维码门禁”

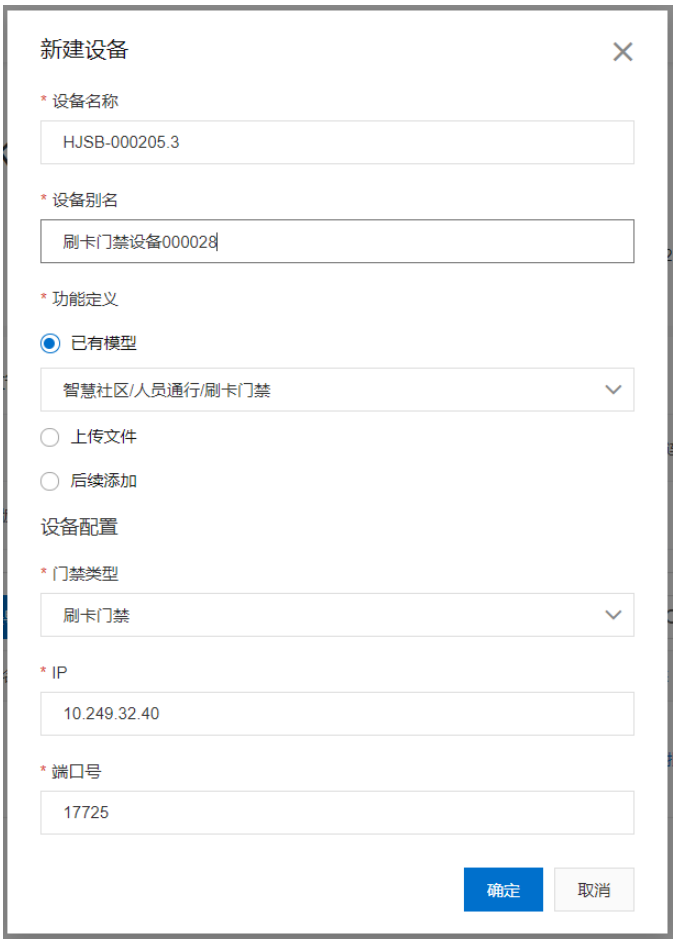


4. **门禁类型**：选择“二维码门禁”
5. **IP**：参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
6. **端口号**：**立林**门禁设备对外提供的网络访问端口号固定是 17725

新建“刷卡门禁设备”

1. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中"**HJSB-000205**"是该设备的IoT编码，"**.3**"标识刷卡门禁设备
2. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028

3. **功能定义**：选择已有模型，对应设备名称选择 “智慧社区/人员通行/刷卡门禁”

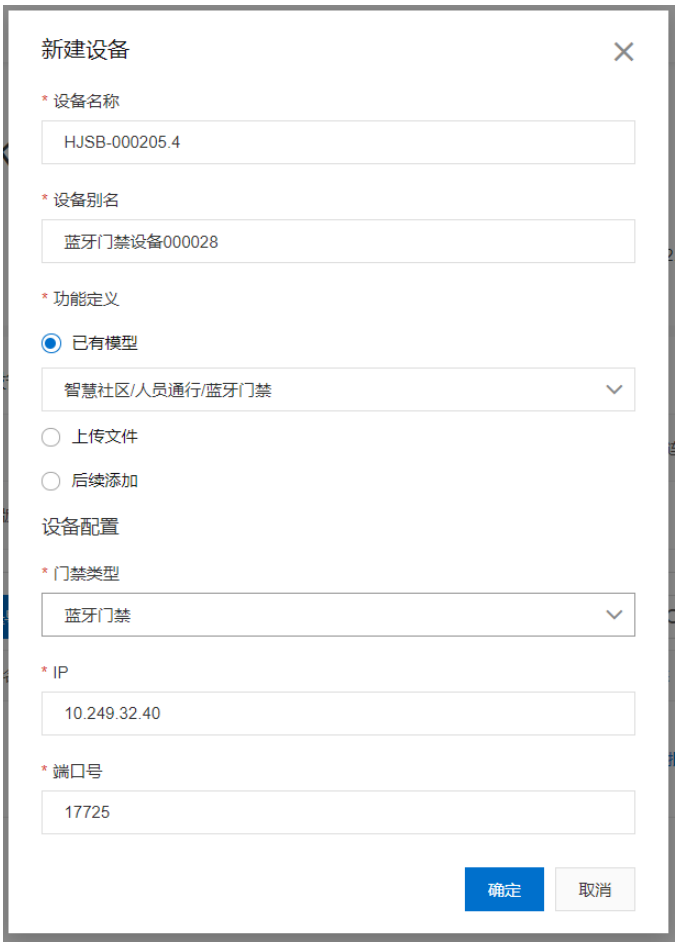


4. **门禁类型**：选择 “刷卡门禁”
5. **IP**：参照甲方提供的<设备信息表>中的该门禁设备IP，输入示例为10.249.32.40
6. **端口号**：**立林**门禁设备对外提供的网络访问端口号固定是 17725

新建 “蓝牙门禁设备”

1. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。蓝牙门禁配置示例为：HJSB-000205.4。其中"**HJSB-000205**"是该设备的IoT编码，"**.4**"标识蓝牙门禁设备
2. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：蓝牙门禁设备000028

3. **功能定义**：选择已有模型，对应设备名称选择 “智慧社区/人员通行/蓝牙门禁”

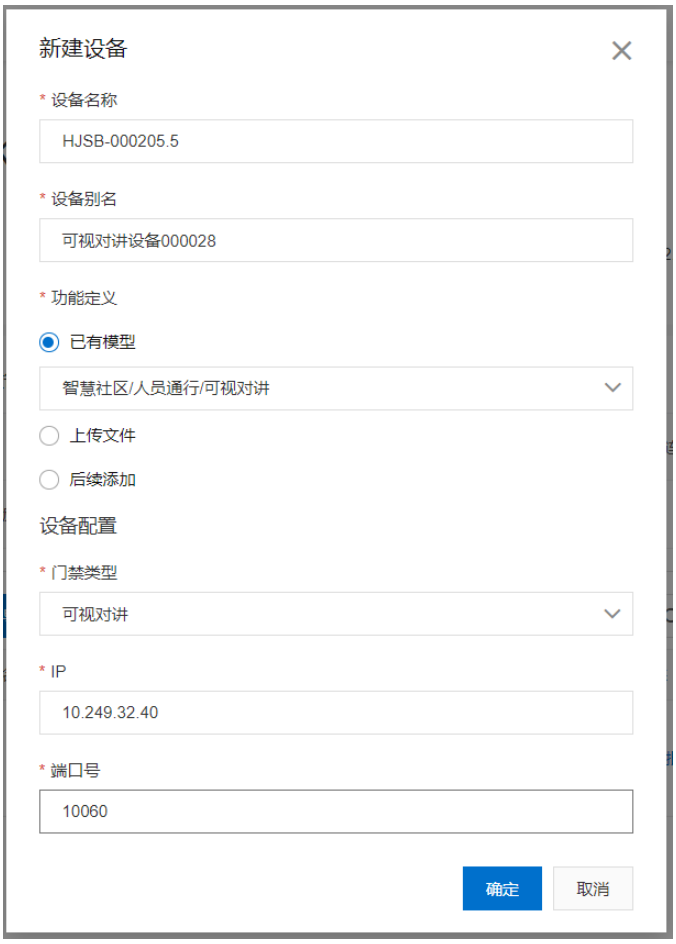


4. **门禁类型**：选择“蓝牙门禁”
5. **IP**：参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
6. **端口号**：**立林**门禁设备对外提供的网络访问端口号固定是**17725**

新建“可视对讲设备”

1. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。可视对讲设备配置示例为：HJSB-000205.5。其中"**HJSB-000205**"是该设备的IoT编码，".**5**"标识可视对讲设备
2. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：可视对讲设备000028

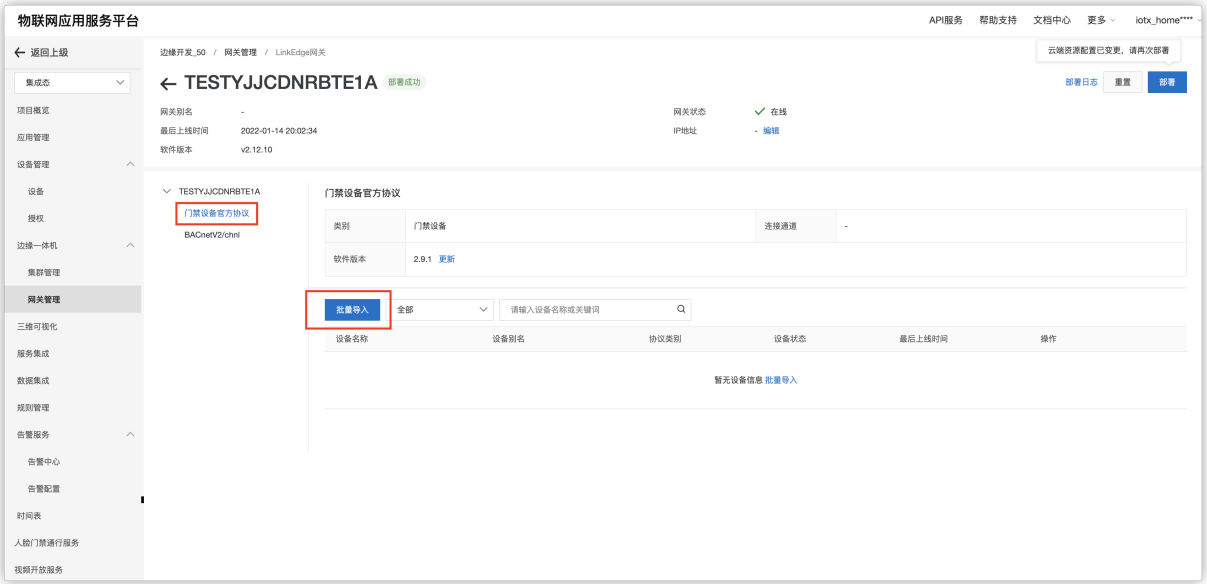
3. **功能定义**：选择已有模型，对应设备名称选择 “智慧社区/人员通行/可视对讲”



4. **门禁类型**：选择“可视对讲”
5. **IP**：参照甲方提供的<设备信息表>中的该可视对讲设备IP。输入示例为10.249.32.40
6. **端口号**：**立林**门禁设备对外提供的可视对讲网络访问端口号固定是**10060**

批量导入门禁设备

1. 选择门禁设备官方协议，出现如下界面，点击 **批量导入**， 如下图所示：



2. 此时会弹出批量导入界面，在该界面点击 **1.请点击下载《协议下设备批量导入模版》**，如下图所示：



2. 下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示：

A	B	C	D	E	F	G	H	I	J	K	L	M
设备名称 (必填，支持英文字母、数字和特殊字符 "-"、"_"、@、"."、"/"，长度限制4~32)	设备别名 (必填，支持中文、英文字母、数字和下划线，4~64个字符，中文算两个)	设备品类 (必填，表示设备归属的品类模型，只能从下拉选项中选择)	门禁类型 (必填，只能从下拉选项中选择)	IP (必填，设备接入IP)	端口 (必填，设备接入端口，0-65535)	人脸匹配相识度 (当设备包含人脸识别功能时必填，范围是0-100的整数)	人脸比对方式 (当门禁类型是人脸门禁时必填)	人脸识别算法服务地址IP (当人脸比对方式为端侧识别时必填，填写算法服务器的IP)	人脸识别算法服务地址端口号 (当人脸识别算法为端侧时必填，填写算法服务器的端口)	人脸测温开关 (当门禁类型是人脸门禁时必填，只能从下拉框选择)	人脸测温下限(C°) (当人脸测温开关打开时必填，范围是30.0°C~45.0°C)	人脸测温上限(C°) (当人脸测温开关打开时必填，范围是30.0°C~45.0°C)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

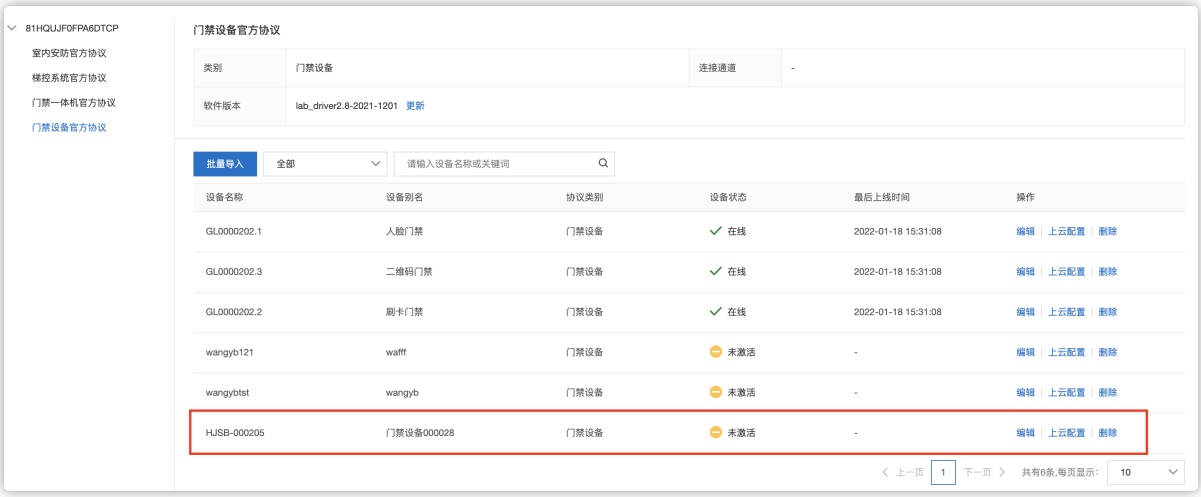
- 设备名称、设备别名、设备品类、门禁类型、IP、端口为**必填项**，
- 设备品类、门禁类型，人脸比对方式，人脸测温开关提供 **下拉框选项**

3. 填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





4. 导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.2. 宇视人行设备接入

说明

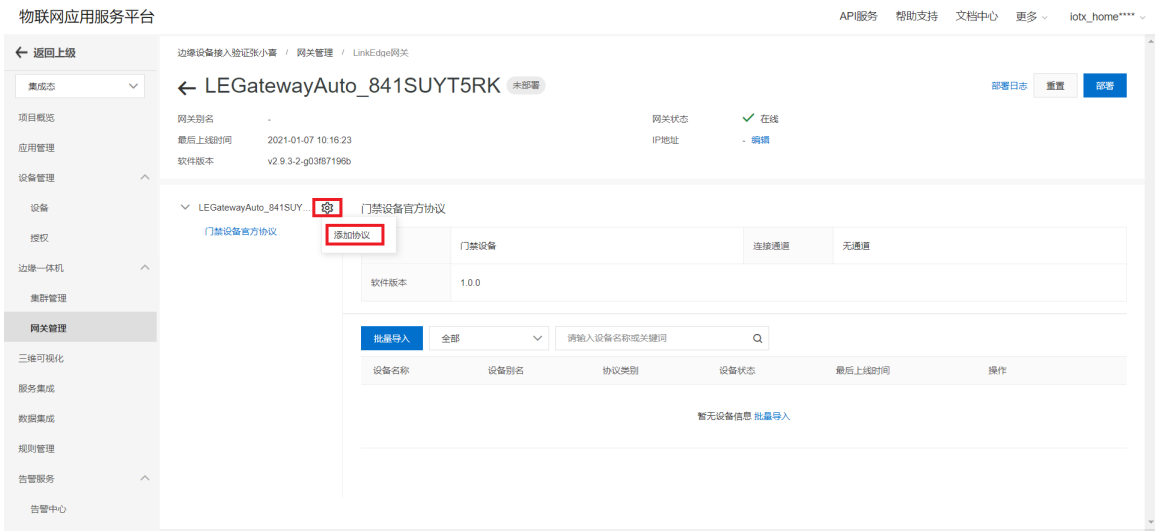
宇视支持对接的型号

对接型号支持	ACS-UAR321L UAR321L (人脸)ET-S35H@B；（人脸/二维码）ET-S51H@B；（人脸/二维码）ET-S32L@W；（人脸/二维码） ET-B32L: (人脸/刷卡)
接入模式	门禁设备
支持功能	人脸
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	80

1. 添加门禁协议，第一次添加协议，可以点击**新建协议**按钮，如下图所示



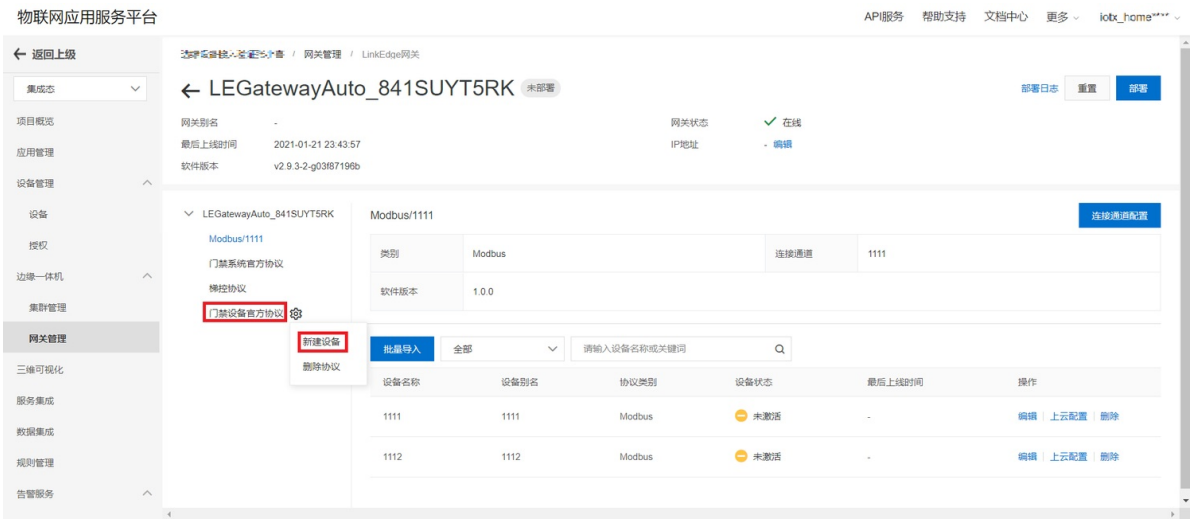
i. 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**2.2.2 添加门禁设备



i. 宇视的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示



2. 添加门禁设备，鼠标移动至**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。

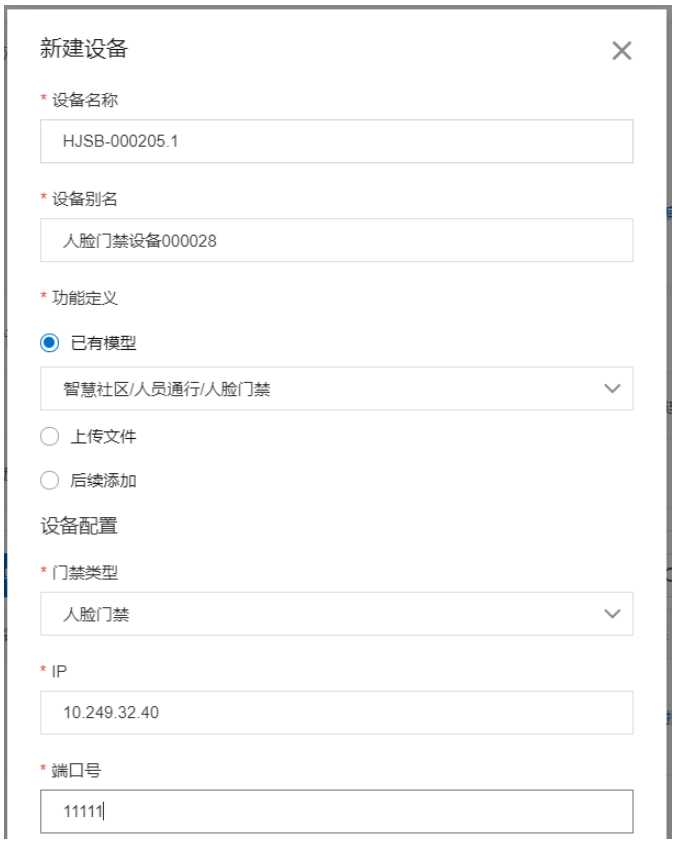


i. 在官方协议下新增人脸门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，"**.1**"标识人脸门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的设备名称。配置示例为：人脸门禁设备000028

c. **功能定义**：门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-人脸门禁。



d. **门禁类型**：选择“人脸门禁”

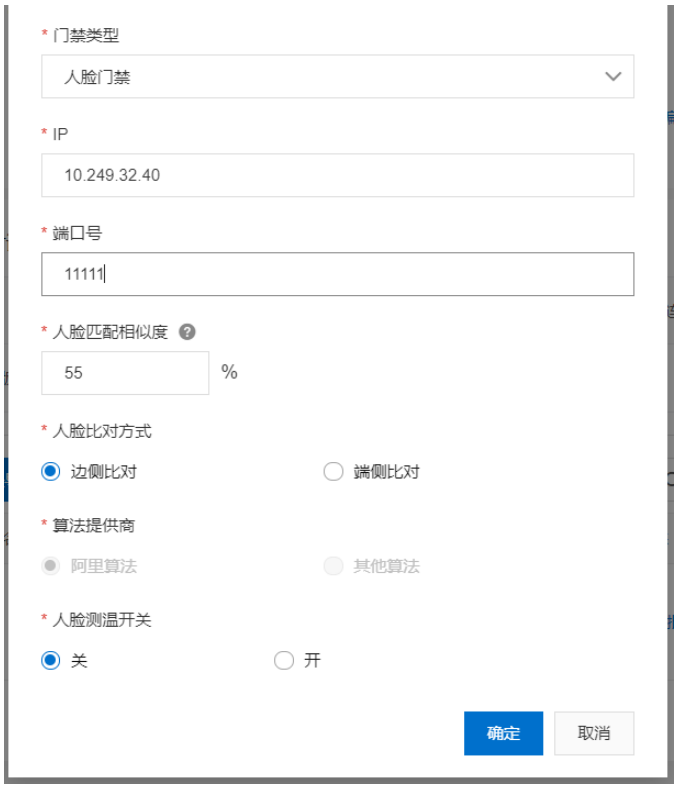
e. **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40

f. **端口号**：**宇视**人脸在边侧识别，不需要设备端口号。这里固定填写80

g. **人脸匹配相似度**：输入55%

h. **人脸比对方式**：**宇视**人脸比对方式为边侧比对

i. **人脸测温开关：宇视**门禁设备不支持人脸测温，设置为关



ii. 在官方协议下新增二维码门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001



c. **门禁类型**：选择“二维码门禁”

d. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

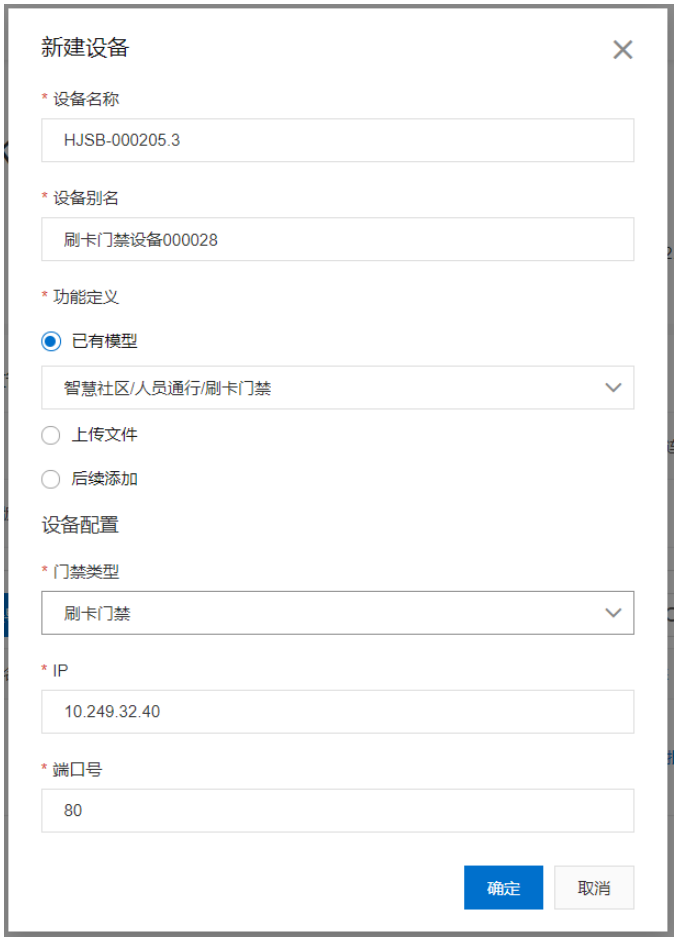
e. **端口号**：宇视门禁设备对外提供的网络访问端口号固定是80

iii. 在官方协议下新增刷卡门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中"**HJSB-000205**"是该设备的IoT编码，"**.3**"标识刷卡门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028

c. **功能定义**：宇视门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-刷卡门禁。



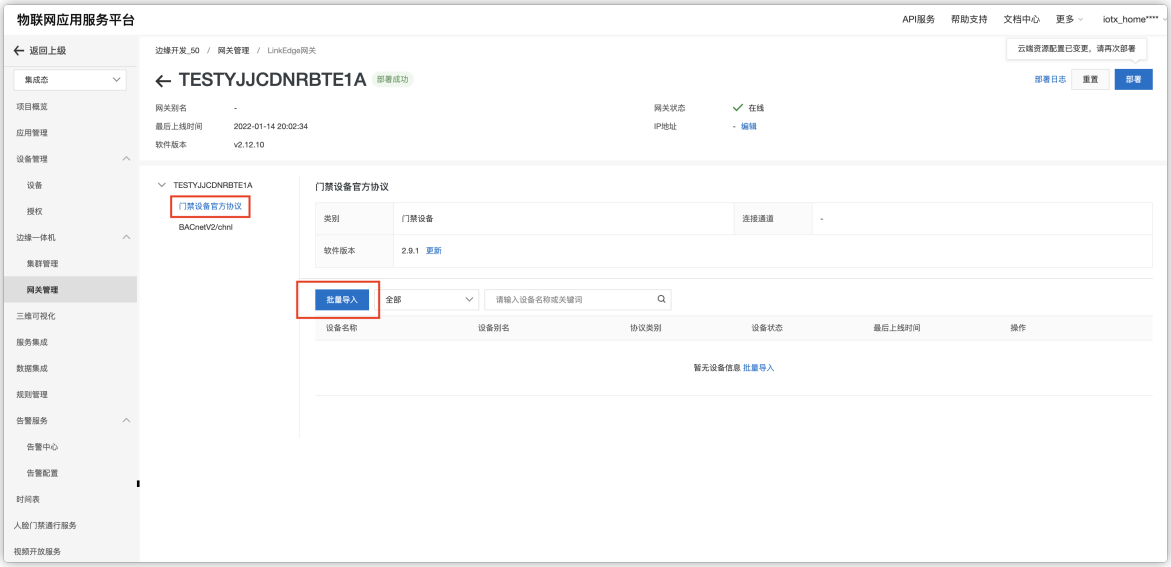
d. **门禁类型**：选择“刷卡门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40

f. **端口号**：宇视门禁设备对外提供的网络访问端口号固定是80

3. **批量导入门禁设备**

选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

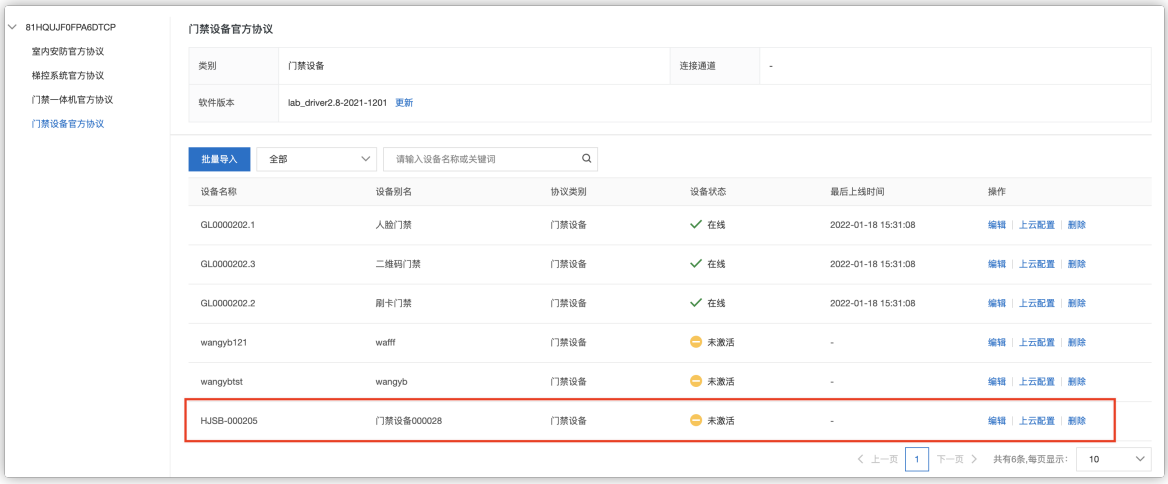
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.3. 捷顺人行设备接入

> 说明
>
> 捷顺支持对接的型号

门禁品类	门禁领御III	门禁Y08
驱动名称	jieshun_RX	jieshun_parking
产品型号	JSMJK02-20B、JSMJK02-40B	JSMJY08A-Q-N、JSMJY08A-02-N、JSMJY08A-N
门禁功能	刷卡门禁	刷卡门禁

捷顺人行门禁品类有"门禁领域III"和"门禁Y08"两个，下面将分别介绍这两类门禁设备的添加操作过程

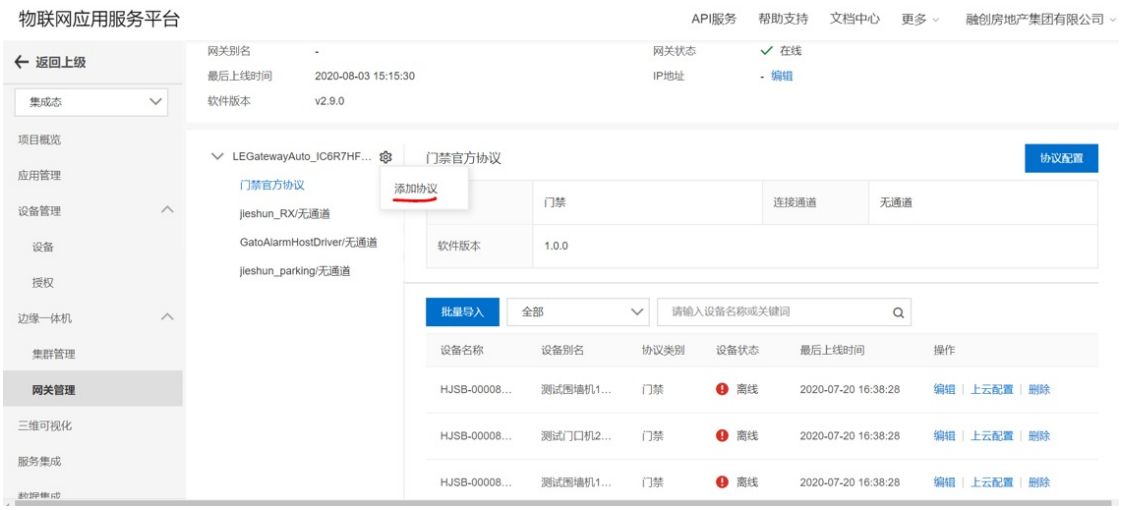
1. “门禁领御III”系列门禁设备集成

i. 添加门禁协议

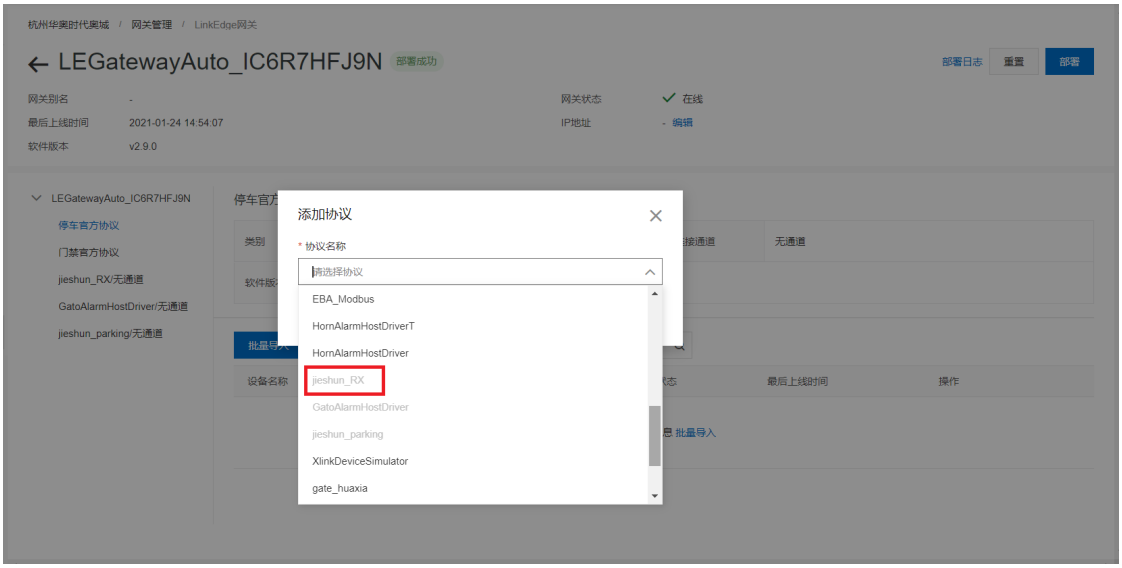
a. 首次添加，选择新建协议



b. 网关里已添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



c. 协议名称选择 jieshun_RX, 如下图所示：



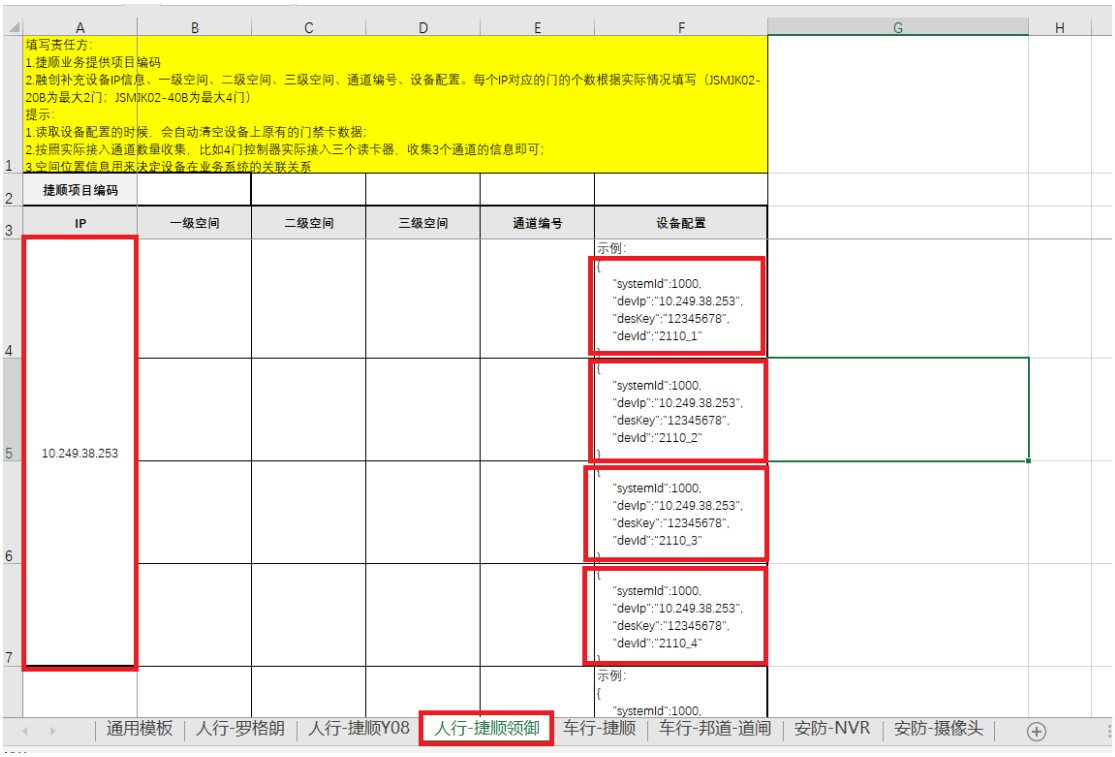
d. 选择完协议，点击确定。

ii. 添加门禁设备

a. 如下图所示，鼠标移动到到刚添加的**jieshun_RX**驱动，点击**新建设备**，进行人行设备的添加。



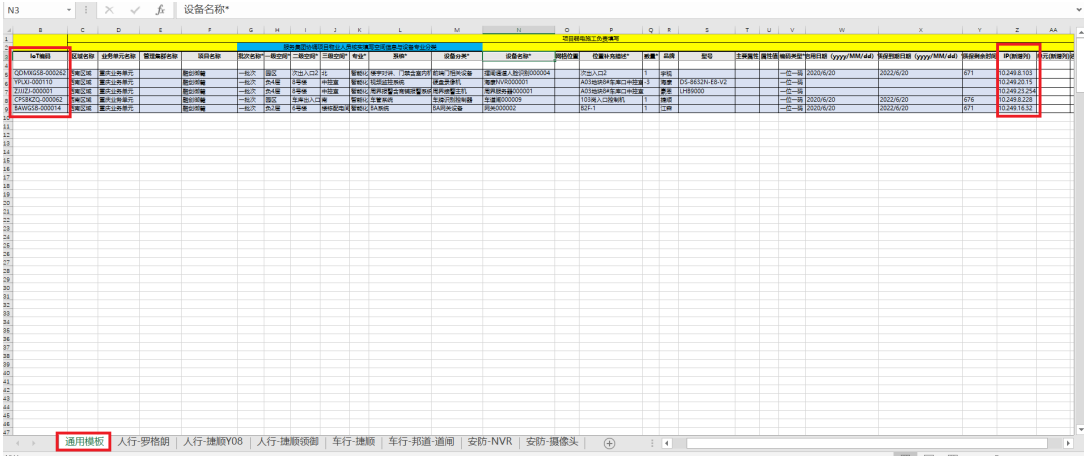
b. 捷顺门禁领域的对接方式是门禁系统，每个门禁系统可管控多个门禁设备。需要为每个门设备进行设备创建。例如下图所示：甲方提供的<IoT设备信息表>，在**人行-捷顺邻御**TAB页，该门禁系统的**IP**地址是10.249.38.253，能够控制的有四个门，对应4个**设备配置**



c. 下来先为第一个门设备进行设备创建。如下图所示，在**jieshun_RX**协议下新增刷卡门禁设备。



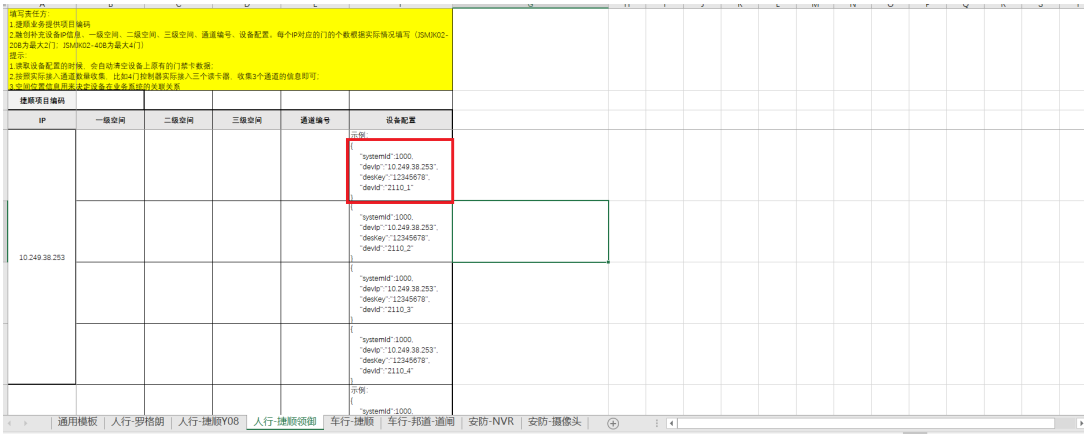
a. **设备名称**：如前面Excel表格截图，参照甲方提供的<IoT设备信息表>，在**人行-捷顺领御**TAB页查看设备**IP**，例如10.249.38.253。然后在<IoT设备信息表>的**通用模板**TAB页，根据IP地址查找门禁控制器**设备IP**对应的**IoT编码**，例如CPSBKZQ-000062，如下图所示。第一个被控制的门禁设备，IoT编码是**CPSBKZQ-000062.1**。其中".1"标识第1个门禁设备



b. **设备别名**：参照甲方提供的<IoT设备信息表>**通用模板**TAB页的**设备名称**。配置示例为：领域门禁_门1

c. **功能定义**：捷顺门禁设备为**已有模型**，对应设备名称选择**智慧社区-人员通行-刷卡门禁**。

d. **设备配置**：如前面Excel表格截图，参照甲方提供的<IoT设备信息表>，在**人行-捷顺领御**TAB页，选择IP地址为10.249.38.253的第一个门禁设备的**设备配置**内容，进行设备配置。如下图所示：



配置示例：{"systemId":1000,"devIp":"10.249.38.253","desKey":"12345678","devId":"2110_1"}，将设备配置信息添入设备配置编辑框里，如上图所示

d. 第二个门禁设备创建：

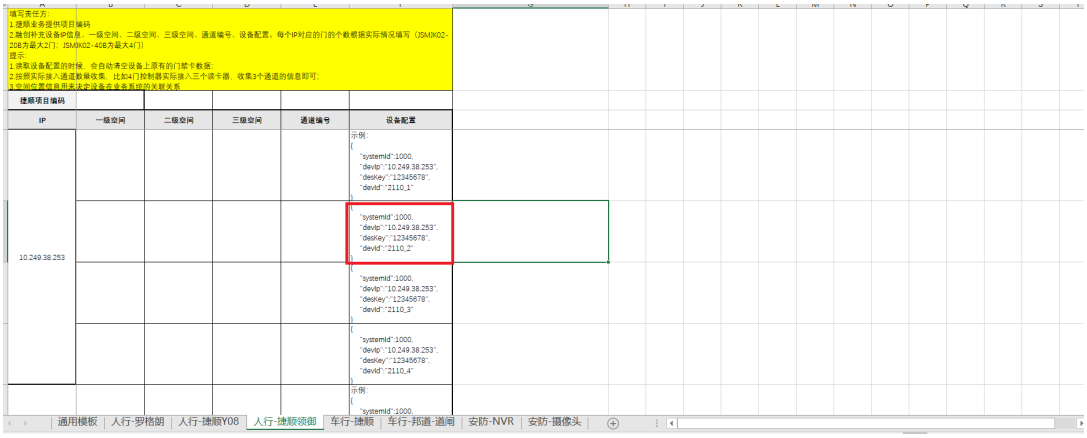


a. **设备名称**：在创建第一个门禁设备时，已经找到门禁控制器的IoT编码是CPSBKZQ-000062。第二个门禁设备的IoT编码是**CPSBKZQ-000062.2**。其中".2"标识第2个门禁设备

b. **设备别名**：与第一个门禁设备相同：参照甲方提供的<设备信息表>中的(捷顺)门禁设备的**设备别名**。配置示例为：领域门禁_门1

c. **功能定义**：与第一个门禁设备相同：捷顺门禁设备为**已有模型**，对应设备名称选择**智慧社区-人员通行-刷卡门禁**。

d. **设备配置**：如前面Excel表格截图，参照甲方提供的<IoT设备信息表>，在**人行-捷顺领御**TAB页，选择IP地址为10.249.38.253的第二个门禁设备的**设备配置**内容，进行设备配置。如下图所示：



e. 第三个门禁设备创建，如下图所示：

a. **设备名称**：第三个门禁设备的IoT编码是**CPSBKZQ-000062.3**。其中".3"标识第3个门禁设备

b. **设备别名**：与第一个相同：领域门禁_门1

c. **功能定义**：与第一个相同：选择为**已有模型**，对应设备名称选择**智慧社区-人员通行-刷卡门禁**。

d. **设备配置**：选择IP地址为10.249.38.253的第三个门禁设备的**设备配置**内容



f. 第四个门禁设备创建，如下图所示：

a. **设备名称**：第四个门禁设备的IoT编码是**CPSBKZQ-000062.4**。其中**".4**"标识第4个门禁设备

b. **设备别名**：与第一个相同：领域门禁_门1

c. **功能定义**：与第一个相同：选择为**已有模型**，对应设备名称选择**智慧社区-人员通行-刷卡门禁**。

d. **设备配置**：选择IP地址为10.249.38.253的第四个门禁设备的**设备配置**内容



e. 门禁领御设备需要与空间绑定。根据设备的IP地址，参照甲方提供的<IoT设备信息表>**通用模板**TAB页的**位置补充描述**，可以得到捷顺门禁系统的空间信息。在2.1.2创建的门禁系统的多个门禁设备，都使用该空间信息进行绑定。具体绑定方法，请参照<物业管理一体机实施手册>的第4章：设备绑定空间。

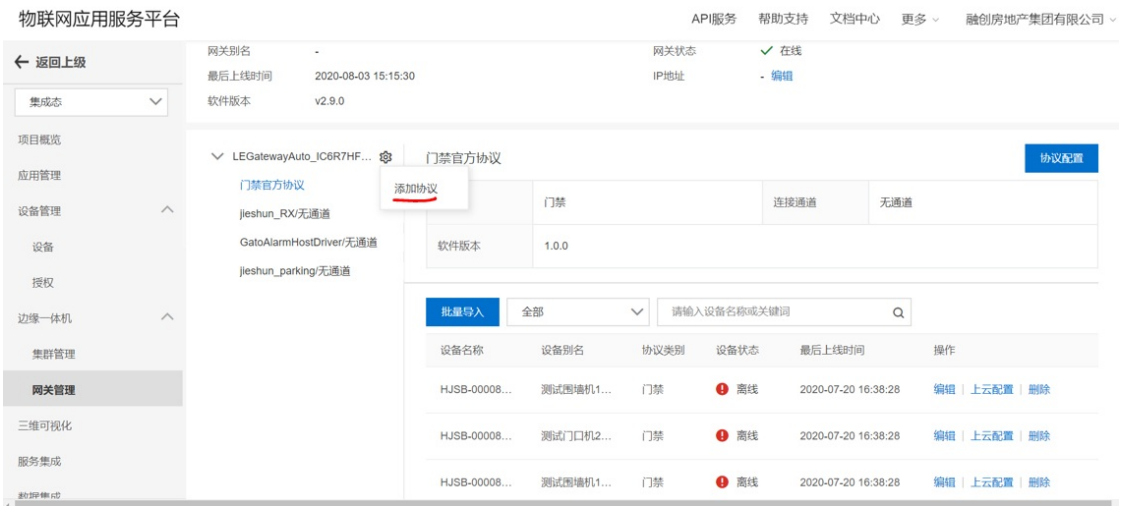
2. “门禁Y08”门禁设备集成

i. 添加门禁协议

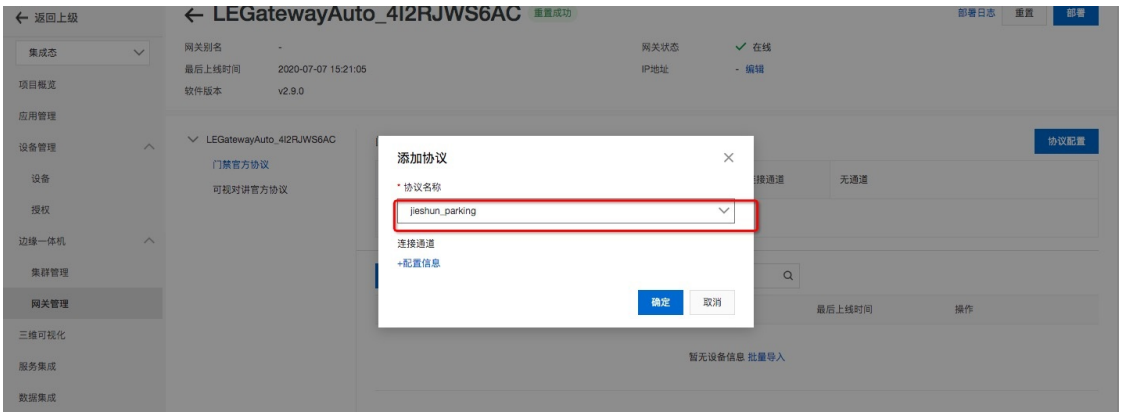
a. 首次添加，选择新建协议



b. 网关里已添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



c. 协议名称选择**jieshun_parking**，如下图所示：



d. 选择完协议，点击确定。

ii. 添加门禁设备，鼠标移动到到刚添加的**jieshun_parking**驱动，点击**新建设备**，进行人行设备的添加。



iii. 在**jieshun_parking**协议下新增刷卡门禁设备。其中：

a. **设备名称**：参照甲方提供的<IoT设备信息表>，在**人行-捷顺Y08**TAB页查看设备**IP**，例如10.249.38.253。然后在<IoT设备信息表>的**通用模板**TAB页，根据门禁**设备IP**查找对应的**IoT编码**。配置示例为QDMXGSB-000444。

b. **设备别名**：参照甲方提供的<设备信息表>，在**通用模板**TAB页，根据门禁设备**IP**查找对应的**设备名称**。配置示例为：捷顺Y08_IP_4

c. **功能定义**：捷顺门禁设备为**已有模型**，对应设备名称选择**智慧社区-人员通行-刷卡门禁**。



d. **设备配置**：参照甲方提供的<设备信息表>中的**人行-捷顺Y08**TAB页里每个设备的**deviceID**，进行设备配置。配置示例如下：{"deviceId":"199118592"}将设备配置信息添入设备配置编辑框里，如上图所示。

3.3.5.3.4. 罗格朗人行设备接入

说明

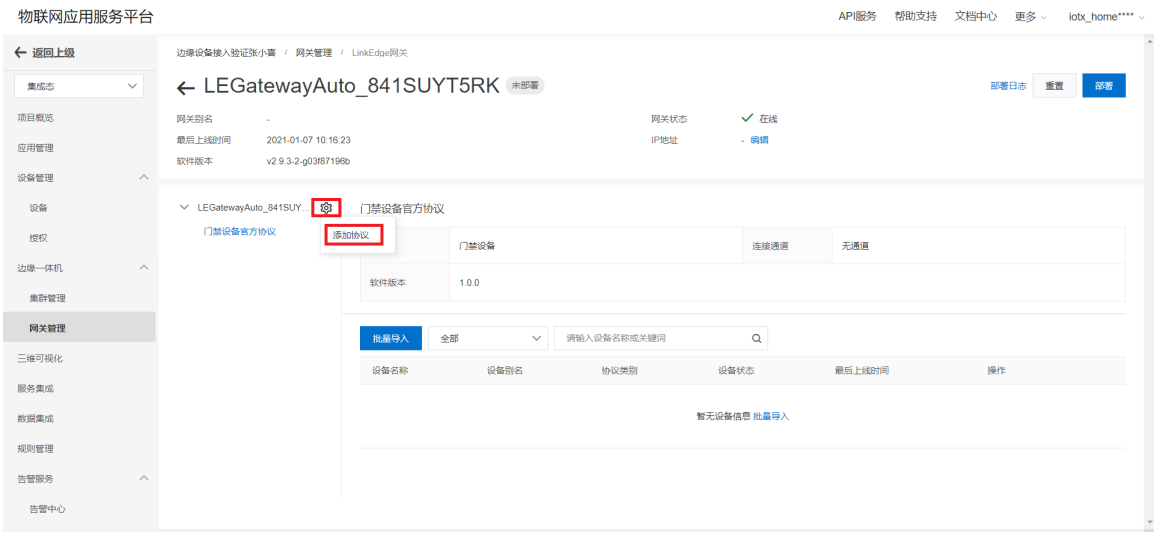
罗格朗支持的型号

对接型号支持	N/A
接入模式	门禁系统
支持功能	人脸/二维码
系统版本	V2.1.9.20200422_rong_chuang
人脸识别方式	边侧识别
门禁端口号	Windows版本的端口是8000，Linux版的是8080

1. 添加门禁协议
   i. 网关里首次添加协议，选择新建协议



   ii. 网关里已添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



iii. 罗格朗的接入模式是系统接入，在弹出的对话框里，选择**门禁系统官方协议**，如下图所示



2. 添加门禁设备

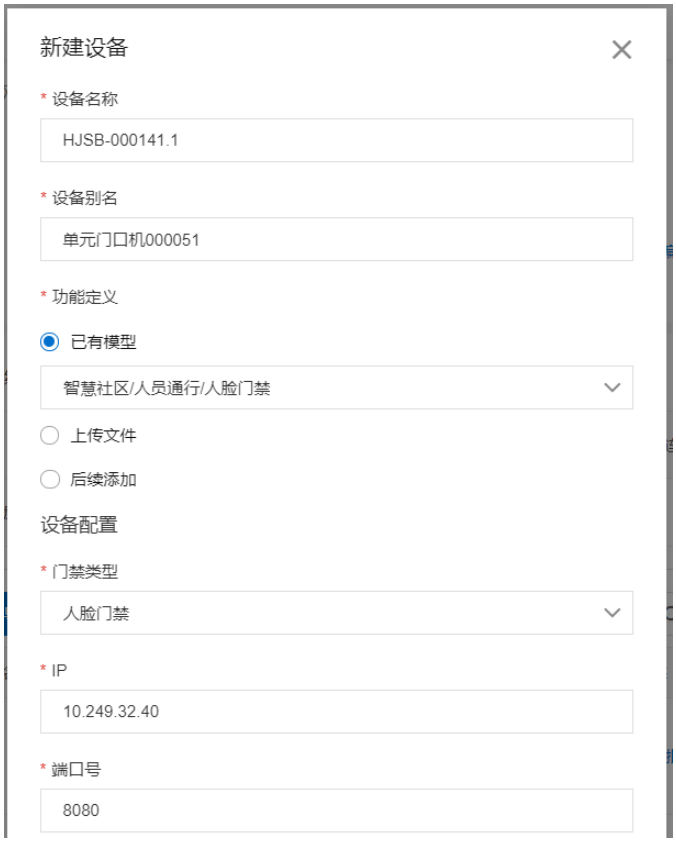
i. 鼠标移动到到刚添加的**门禁官方协议**，点击**新建设备**，进行人行设备的添加。

ii. 在官方协议下新增人脸门禁设备。其中：

a. **设备名称**：参照甲方提供的<IoT设备信息表>，在**人行-罗格朗**TAB页查看设备**IP**，例如10.249.38.253。然后在<IoT设备信息表>的**通用模板**TAB页，根据IP地址查找门禁控制器**设备IP**对应的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000141.1。其中"HJSB-000141"是该设备的IoT编码，**".1**"标识人脸门禁设备

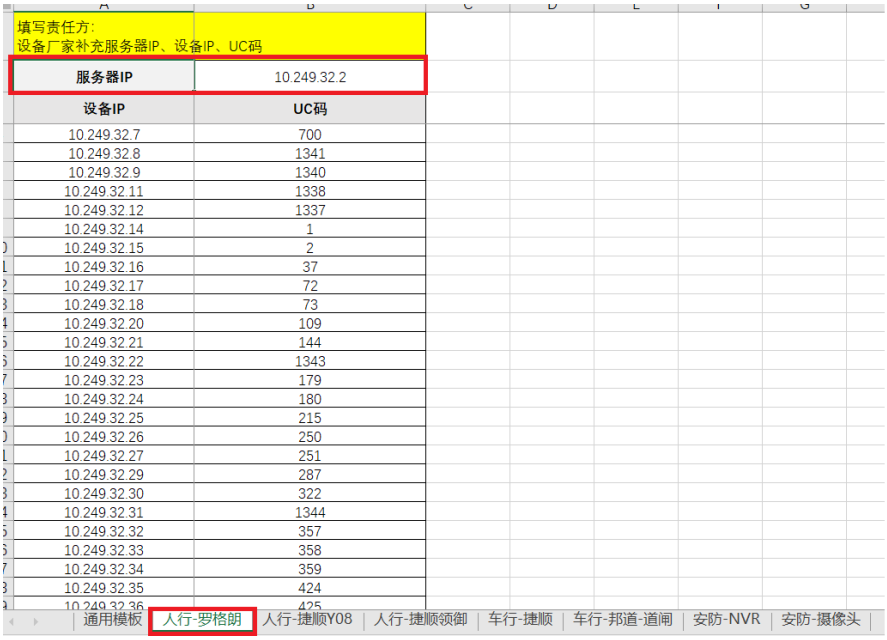
b. **设备别名**：参照甲方提供的<设备信息表>中的设备名称。配置示例为：单元门口机000051

c. **功能定义**：罗格朗门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-人脸门禁。



d. **门禁类型**：选择“人脸门禁”

e. **IP**:参照甲方提供的<设备信息表>中**人行-罗格朗**TAB页的该门禁系统**服务器IP**,而不是设备IP。输入示例为10.249.32.40。如下图所示



填写责任方:
设备厂家补充服务器IP、设备IP、UC码

服务器IP	10.249.32.2
设备IP	UC码
10.249.32.7	700
10.249.32.8	1341
10.249.32.9	1340
10.249.32.11	1338
10.249.32.12	1337
10.249.32.14	1
10.249.32.15	2
10.249.32.16	37
10.249.32.17	72
10.249.32.18	73
10.249.32.20	109
10.249.32.21	144
10.249.32.22	1343
10.249.32.23	179
10.249.32.24	180
10.249.32.25	215
10.249.32.26	250
10.249.32.27	251
10.249.32.29	287
10.249.32.30	322
10.249.32.31	1344
10.249.32.32	357
10.249.32.33	358
10.249.32.34	359
10.249.32.35	424
10.249.32.36	425

通用模板 | 人行-罗格朗 | 人行-捷顺Y08 | 人行-捷顺领御 | 车行-捷顺 | 车行-邦道-道闸 | 安防-NVR | 安防-摄像头

f. **端口号**：罗格朗门禁设备对外提供的网络访问端口号：Windows版本的端口是8000，Linux版的是8080。罗格朗落地设备是windows版本还是Linux版本，请联系罗格朗设备厂商确认。

g. **人脸匹配相似度**：输入55%

h. **设备唯一标识**：参照甲方提供的<设备信息表>中**人行-罗格朗**TAB页的该门禁设备的**UC码**。输入示例为1271；特别注意：每个设备的UC码必须是唯一的，如果有重复的，部署后将导致所有罗格朗设备离线

i. **人脸比对方式**：**罗格朗**人脸比对方式为边侧比对

j. **人脸测温开关**：**罗格朗**门禁设备不支持人脸测温，设置为关

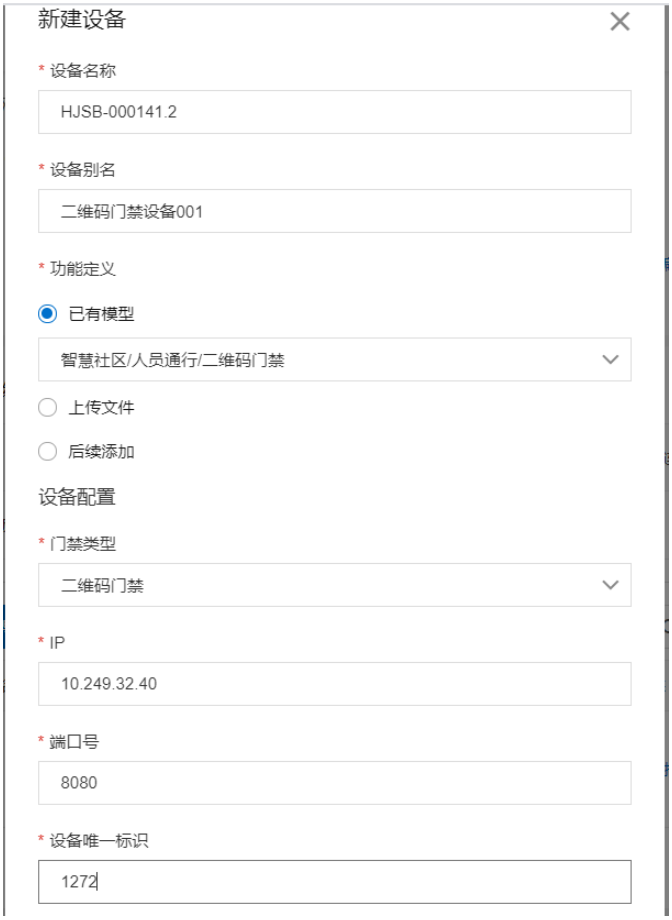


iii. 在官方协议下新增二维码门禁设备。其中：

a. **设备名称**：参照甲方提供的<IoT设备信息表>，在**人行-罗格朗**TAB页查看设备**IP**，例如10.249.38.253。然后在<IoT设备信息表>的**通用模板**TAB页，根据IP地址查找门禁控制器**设备IP**对应的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000141.2。其中"**HJSB-000141**"是该设备的IoT编码，"**.2**"标识二维码门禁设备

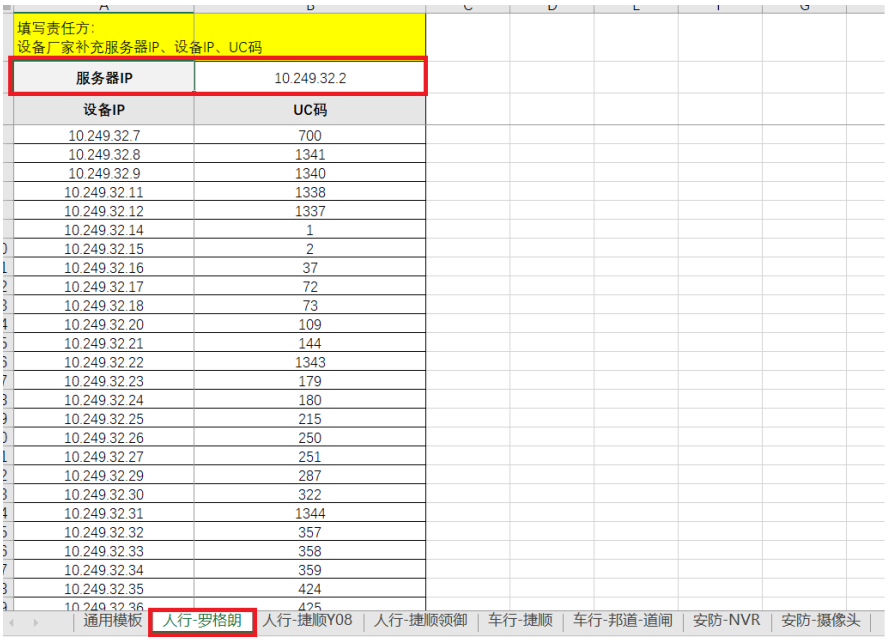
b. **设备别名**：参照甲方提供的<设备信息表>中的设备名称。配置示例为：二维码门禁设备001

c. **功能定义**：罗格朗门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



d. **门禁类型**：选择“二维码门禁”

e. **IP**:参照甲方提供的<设备信息表>中的门禁系统**服务器IP**,而不是设备IP。输入示例为10.249.32.40。如下图所示



填写责任方: 设备厂家补充服务器IP、设备IP、UC码	
服务器IP	10.249.32.2
设备IP	**UC码**
10.249.32.7	700
10.249.32.8	1341
10.249.32.9	1340
10.249.32.11	1338
10.249.32.12	1337
10.249.32.14	1
10.249.32.15	2
10.249.32.16	37
10.249.32.17	72
10.249.32.18	73
10.249.32.20	109
10.249.32.21	144
10.249.32.22	1343
10.249.32.23	179
10.249.32.24	180
10.249.32.25	215
10.249.32.26	250
10.249.32.27	251
10.249.32.29	287
10.249.32.30	322
10.249.32.31	1344
10.249.32.32	357
10.249.32.33	358
10.249.32.34	359
10.249.32.35	424
10.249.32.36	425

通用模板 | 人行-罗格朗 | 人行-捷顺Y08 | 人行-捷顺领御 | 车行-捷顺 | 车行-邦道-道闸 | 安防-NVR | 安防-摄像头

f. **端口号**：罗格朗门禁设备对外提供的网络访问端口号：Windows版本的端口是8000，Linux版的是8080。罗格朗落地设备是windows版本还是Linux版本，请联系罗格朗设备厂商确认

g. **设备唯一标识**：参照甲方提供的<设备信息表>中的该门禁设备的**设备ID**。输入示例为1272；特别注意：每个设备的UC码必须是唯一的，如果有重复的，部署后将导致所有罗格朗设备离线

3.3.5.3.5. 立方人行设备接入

说明

立方支持的型号

对接型号支持	门禁控制器RF-AX500G2
接入模式	门禁系统
支持功能	刷卡(一方)、二维码(三方)
系统版本	N/A
人脸识别方式	N/A
门禁端口号	8080

3.3.5.3.6. 海康人行设备接入

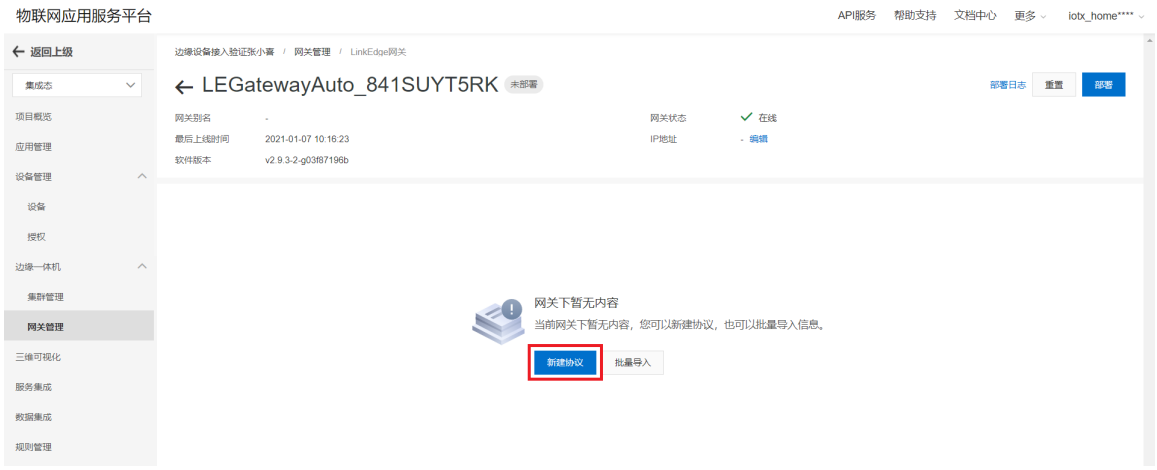
说明

海康支持的型号

对接型号支持	DS-K1T6XYZUVW-ABC
接入模式	门禁设备
支持功能	人脸/二维码/刷卡/蓝牙/可视对讲/人体测温(可选)
系统版本	V3.2.32
人脸识别方式	端侧识别
人脸建模算法端口	80 (说明：融创北京的端口号是9000)
门禁端口号	80
可视对讲端口号	80

1. 门禁设备集成

i. 第一次添加协议，可以点击**新建协议**按钮，如下图所示



ii. 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**

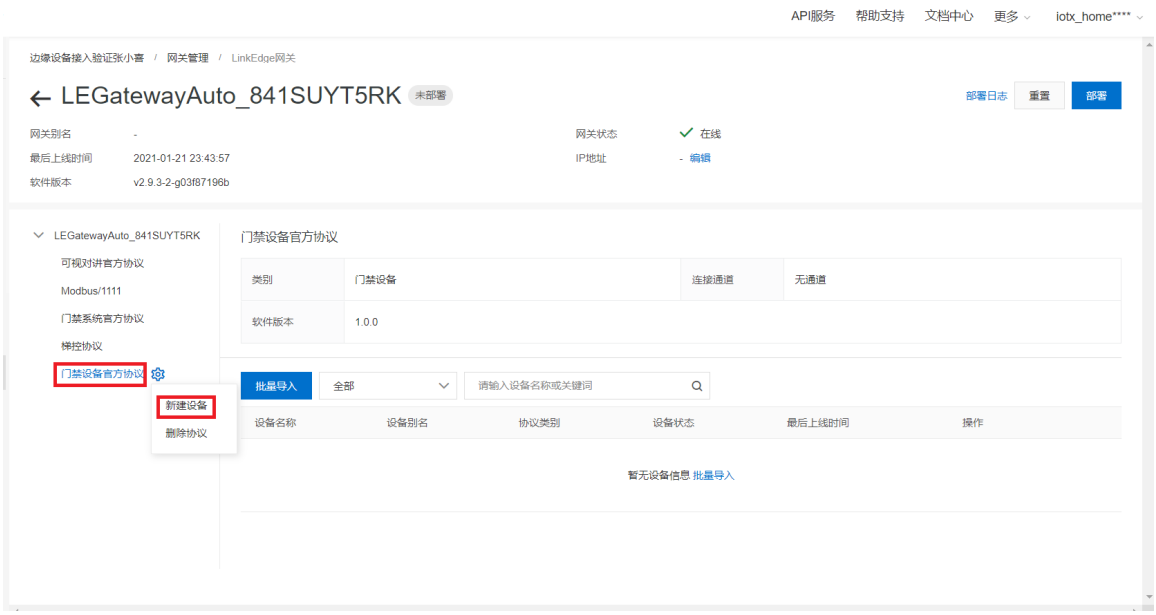


iii. 海康的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示



2. 添加门禁设备

i. 如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。



ii. 在官方协议下新增人脸门禁设备。其中：

a. 设备名称：参照甲方提供的<设备信息表>中的门禁设备的IoT编码，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"HJSB-000205"是该设备的IoT编码，".1"标识人脸门禁设备

b. 设备别名：参照甲方提供的<设备信息表>中的

c. 设备名称。配置示例为：人脸门禁设备000028功能定义：海康门禁设备为已有模型，对应设备名称选择"智慧社区-人员通行-人脸门禁"



d. **门禁类型**：选择“**人脸门禁**”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

f. **端口号**：**海康**门禁设备对外提供的网络访问端口号固定是80

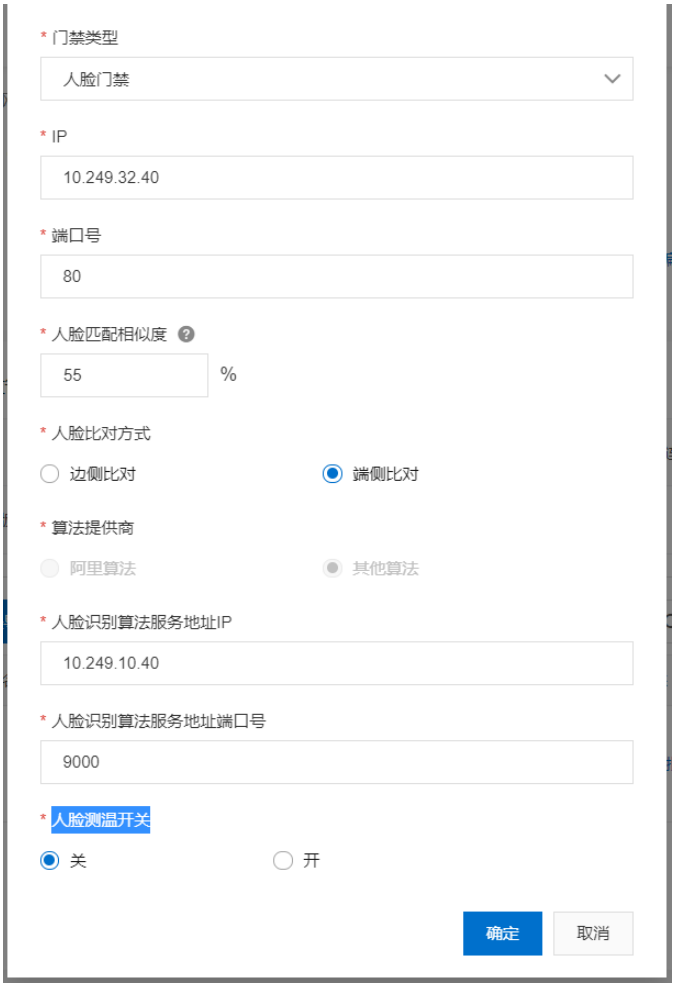
g. **人脸匹配相似度**：输入55%

h. **人脸比对方式**：海康人脸比对方式为**端侧比对**

i. **人脸识别算法服务地址IP**：**海康人脸识别算法服务器的地址IP。特别注意：这个IP地址请与项目联系确认。**输入示例为10.249.10.40

j. **人脸识别算法服务地址端口号**:**海康**人脸识别算法服务地址端口号固定为**80**

k. **人脸测温开关**：海康门禁设备不支持人脸测温，选择关。

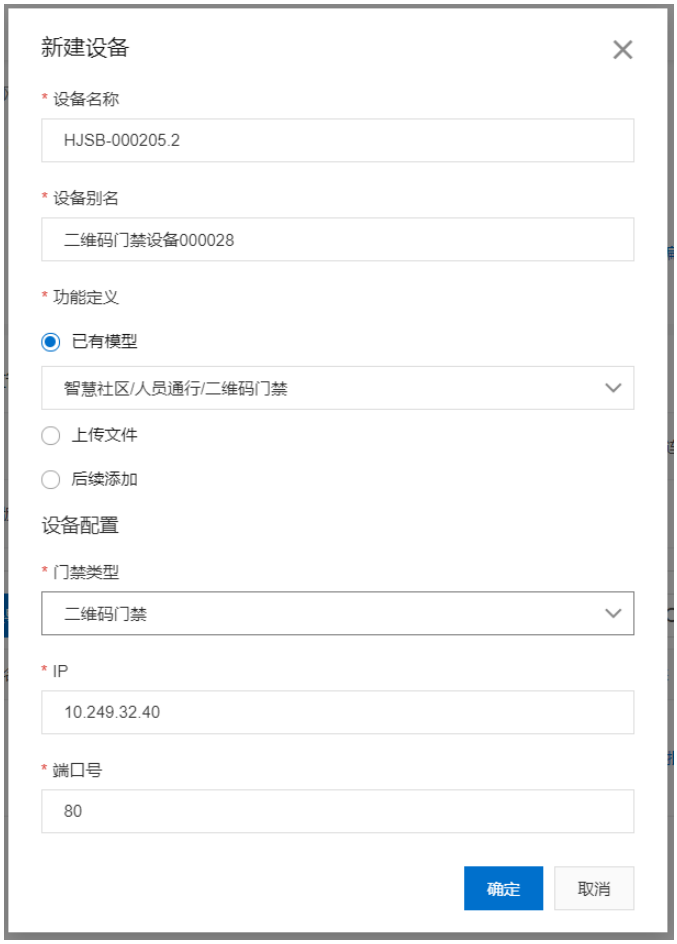


iii. 在官方协议下新增二维码门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备000028

c. **功能定义**：海康门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



d. **门禁类型**：选择“二维码门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

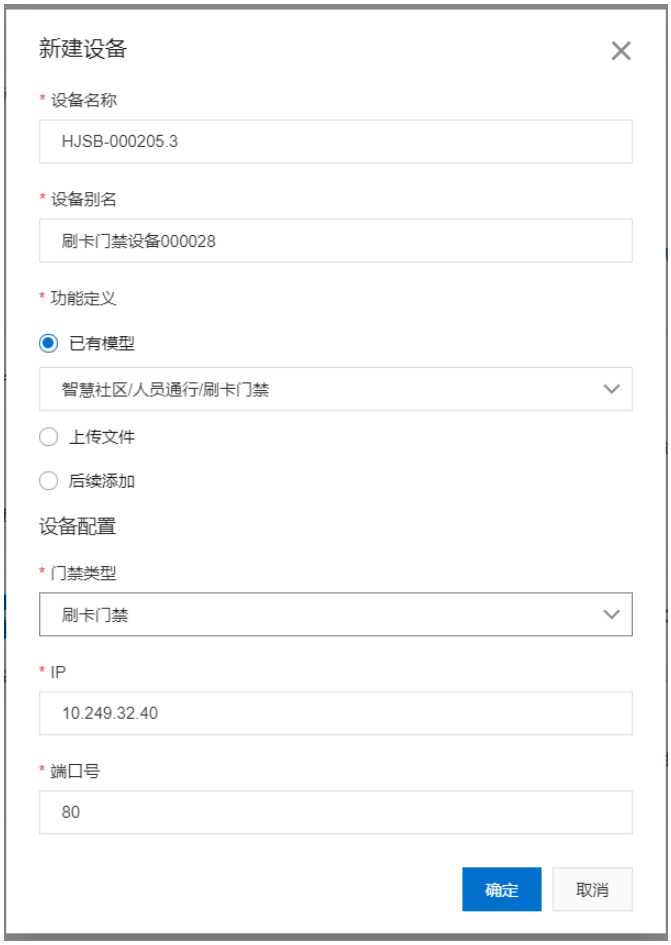
f. **端口号**：海康门禁设备对外提供的网络访问端口号固定是80

iv. 在官方协议下新增刷卡门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中"**HJSB-000205**"是该设备的IoT编码，".**3**"标识刷卡门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028

c. **功能定义**：海康门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-刷卡门禁。



d. **门禁类型**：选择“刷卡门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40

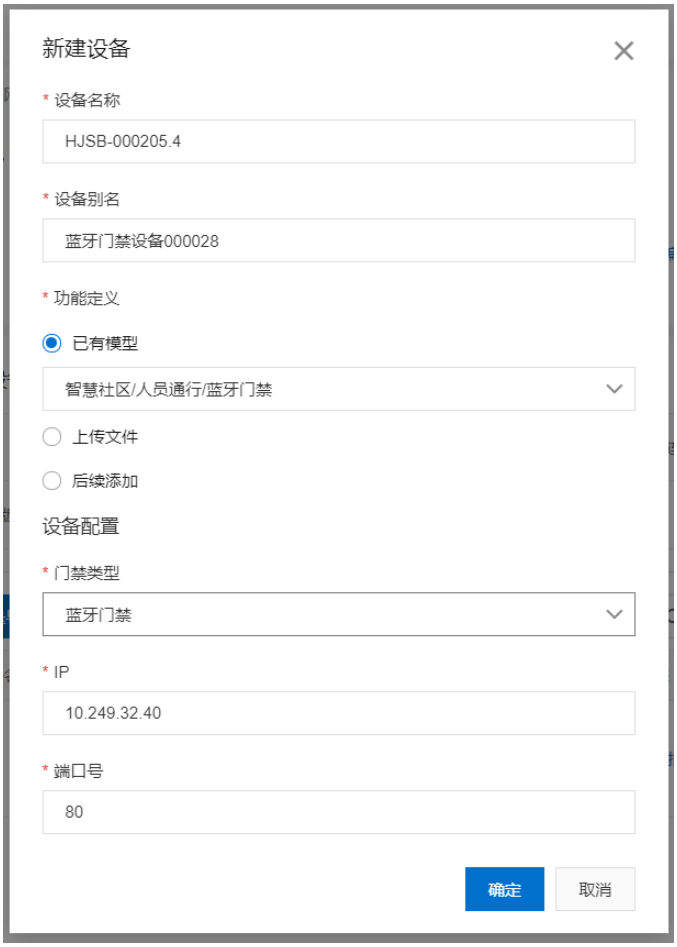
f. **端口号**：海康门禁设备对外提供的网络访问端口号固定是80

v. 在官方协议下新增蓝牙门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。蓝牙门禁配置示例为：HJSB-000205.4。其中"**HJSB-000205**"是该设备的IoT编码，"**.4**"标识蓝牙门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：蓝牙门禁设备000028

c. **功能定义**：海康门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-蓝牙门禁。



d. **门禁类型**：选择“蓝牙门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

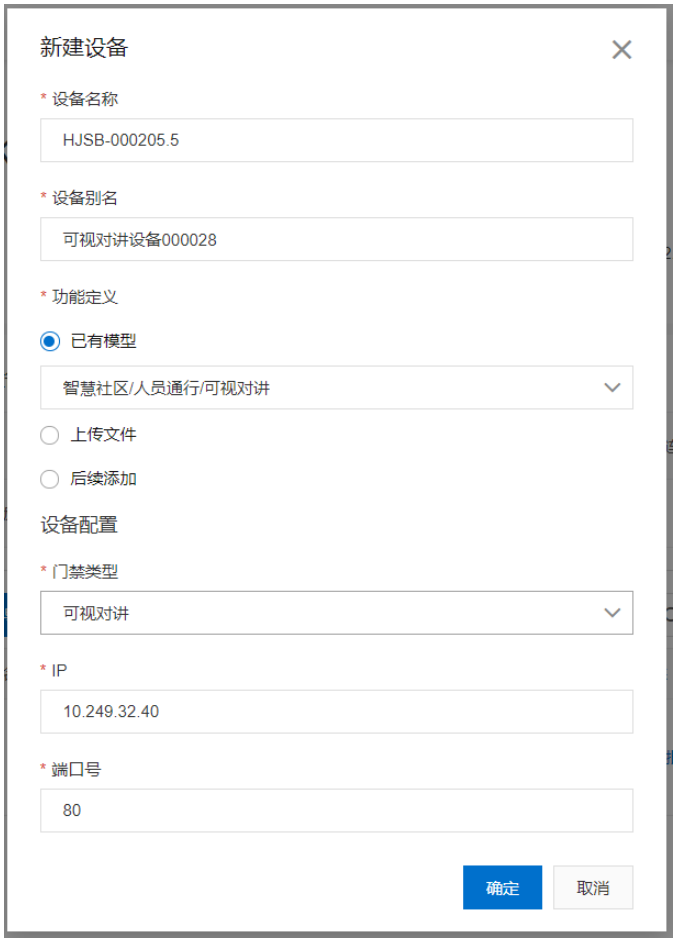
f. **端口号**：海康门禁设备对外提供的网络访问端口号固定是**80**

vi. 在官方协议下新增可视对讲设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。可视对讲设备配置示例为：HJSB-000205.5。其中"**HJSB-000205**"是该设备的IoT编码，".5"标识可视对讲设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：可视对讲设备000028

c. **功能定义**：选择已有模型，对应设备名称选择智慧社区-人员通行-可视对讲。



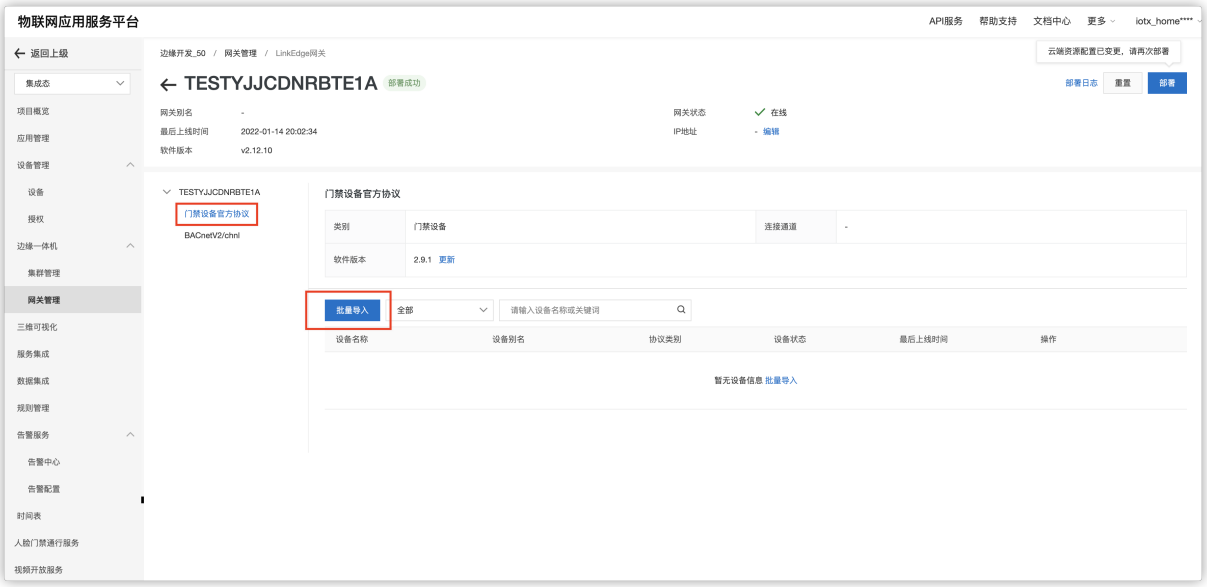
d. **门禁类型**：选择“可视对讲”

e. **IP**:参照甲方提供的<设备信息表>中的该可视对讲设备IP。输入示例为10.249.32.40

f. **端口号**：**海康**门禁设备对外提供的可视对讲网络访问端口号固定是**80**

3、**批量导入门禁设备**

选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件

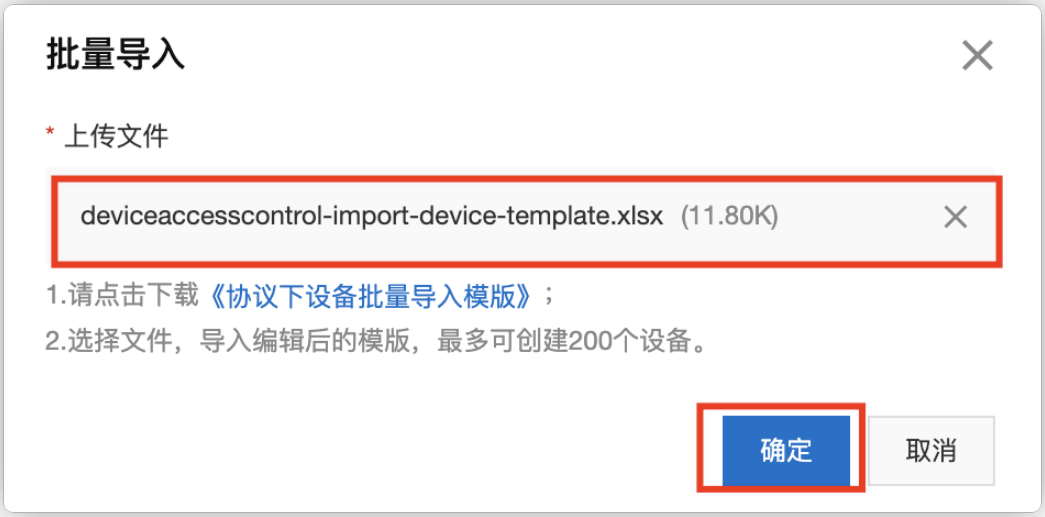


下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示

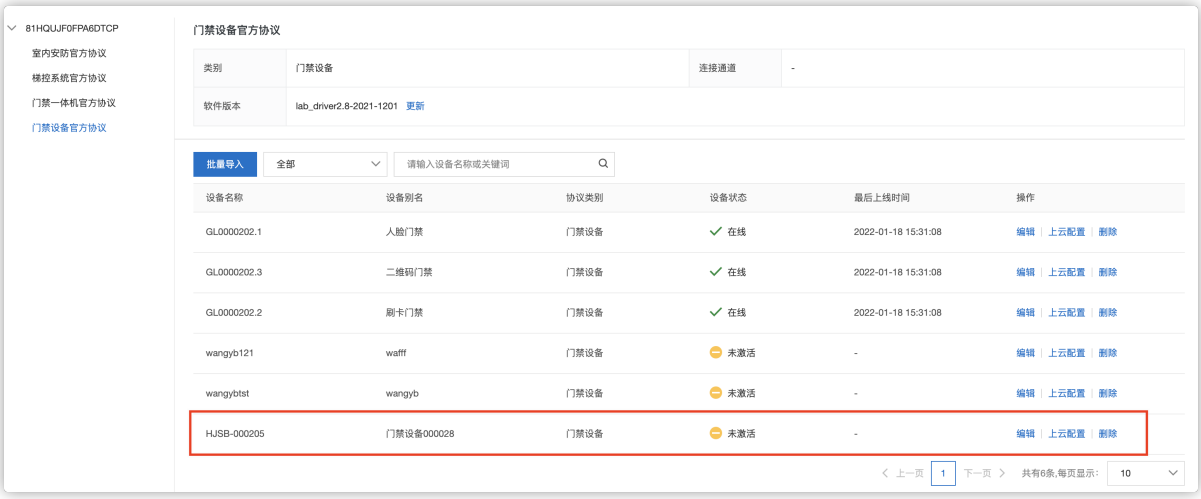
A	B	C	D	E	F	G	H	I	J	K	L	M
设备名称 (必填，支持英文字母、数字和特殊字符 ^ 、 ' 、 ; 、 @ 、 ~ 、 ` 、 ! 、 # 、 $ 、 % 、 ^ 、 & 、 * 、 (、) 、 - 、 _ 、 + 、 = 、 { 、 } 、 [、] 、 \ 、 | 、 : 、 " 、 < 、 > 、 , 、 . 、 ? 、 / 、 空格，长度限制4~32)	设备别名 (必填，支持中文、英文字母、数字和下划线，4~64个字符，中文算两个)	设备品类 (必填，表示设备归属的品类模型，只能从下拉选项中选择)	门禁类型 (必填，只能从下拉选项中选择)	IP (必填，设备接入IP)	端口 (必填，设备接入端口，0-65535)	人脸匹配相识度 (当设备包含人脸识别功能时必填，范围是0-100的整数)	人脸比对方式 (当门禁类型是人脸门禁时必填)	人脸识别算法服务地址IP (当人脸比对方式为端侧识别时必填，填写算法服务器的IP)	人脸识别算法服务地址端口号 (当人脸识别算法为端侧时必填，填写算法服务器的端口)	人脸测温开关 (当门禁类型是人脸门禁时必填，只能从下拉框选择)	人脸测温下限(C°) (当人脸测温开关打开时必填，范围是30.0°C~45.0°C)	人脸测温上限(C°) (当人脸测温开关打开时必填，范围是30.0°C~45.0°C)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

其中**设备名称**、**设备别名**、**设备品类**、**门禁类型**、**IP**、**端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入



导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.7. 狄耐克人行设备接入

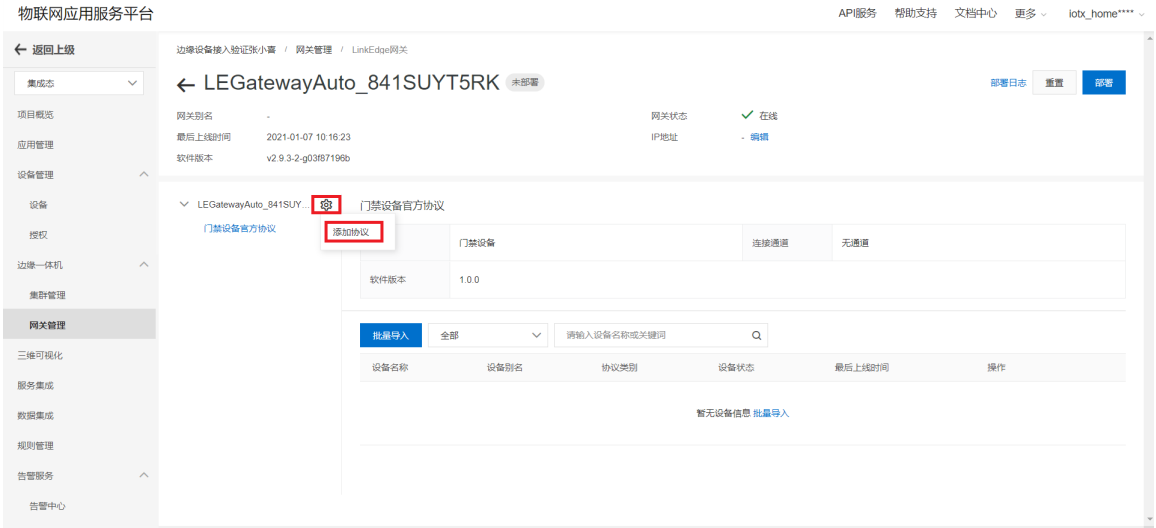
> 说明
>
> 狄耐克支持的型号

对接型号支持	902D, 905D, 904D, 280D, 280AC
接入模式	门禁设备
支持功能	人脸/二维码/刷卡/蓝牙/可视对讲
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	10088
可视对讲端口号	10088

1. 添加门禁协议
   i. 第一次添加协议，可以点击**新建协议**按钮，如下图所示



   ii. 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**

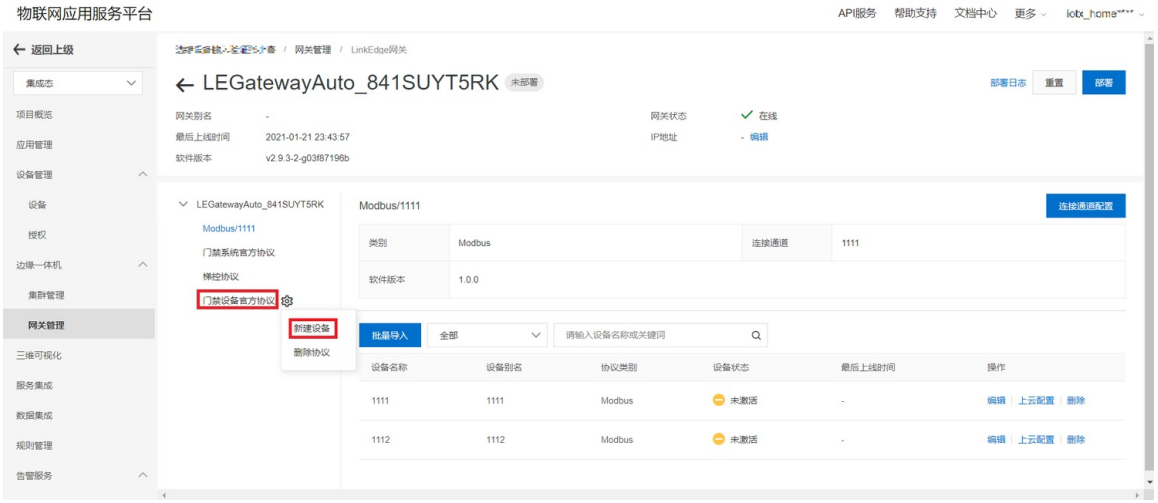


iii. 狄耐克的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示



2. 添加门禁设备

i. 如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。

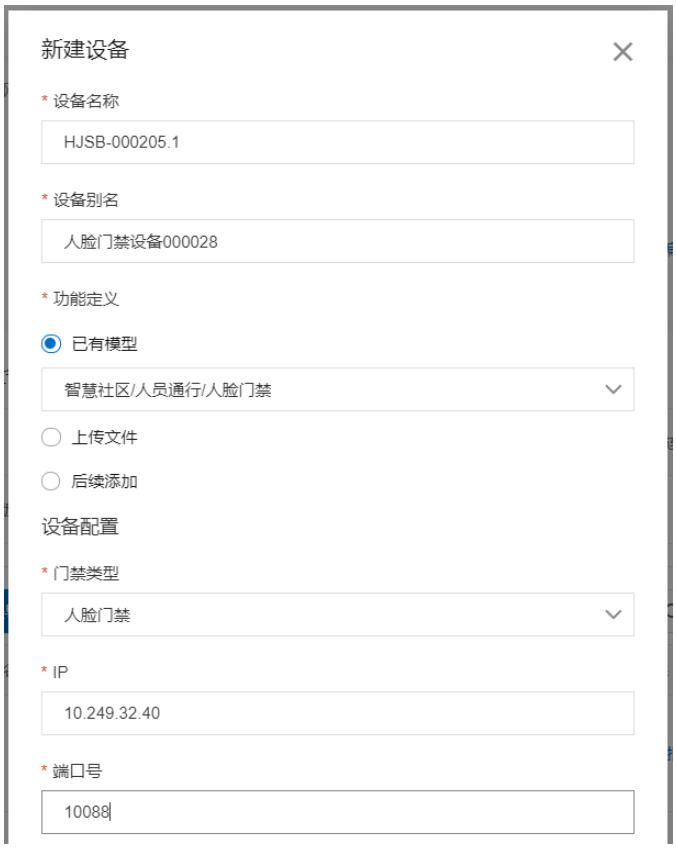


ii. 在官方协议下新增人脸门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，"**.1**"标识人脸门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028

c. **功能定义**：狄耐克门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"



d. **门禁类型**：选择“**人脸门禁**”

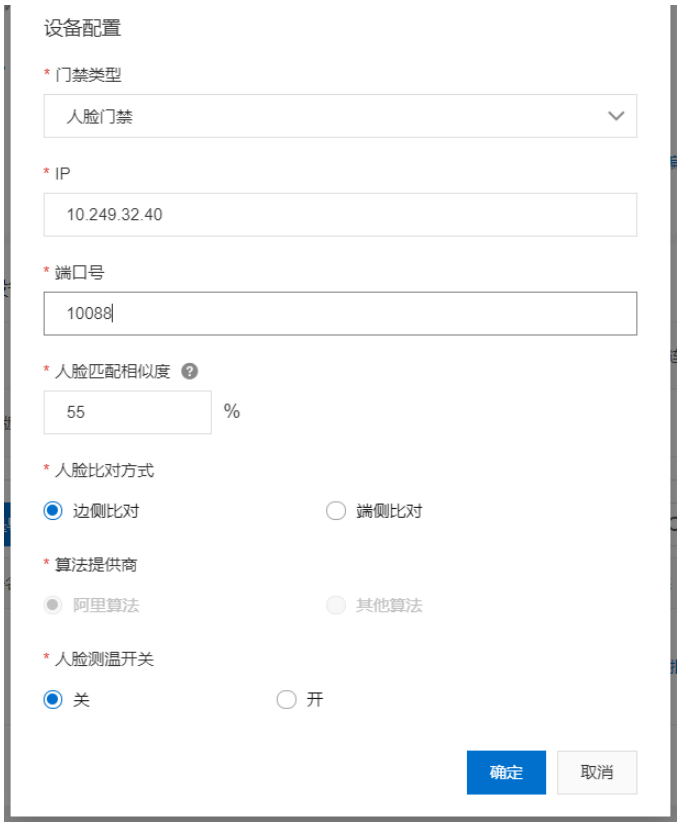
e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

f. **端口号**：狄耐克门禁设备对外提供的网络访问端口号固定是**10088**

g. **人脸匹配相似度**：输入55%

h. **人脸比对方式**：**狄耐克**人脸比对方式为**边侧比对**

i. **人脸测温开关：狄耐克**门禁设备不支持人脸测温，设置为关

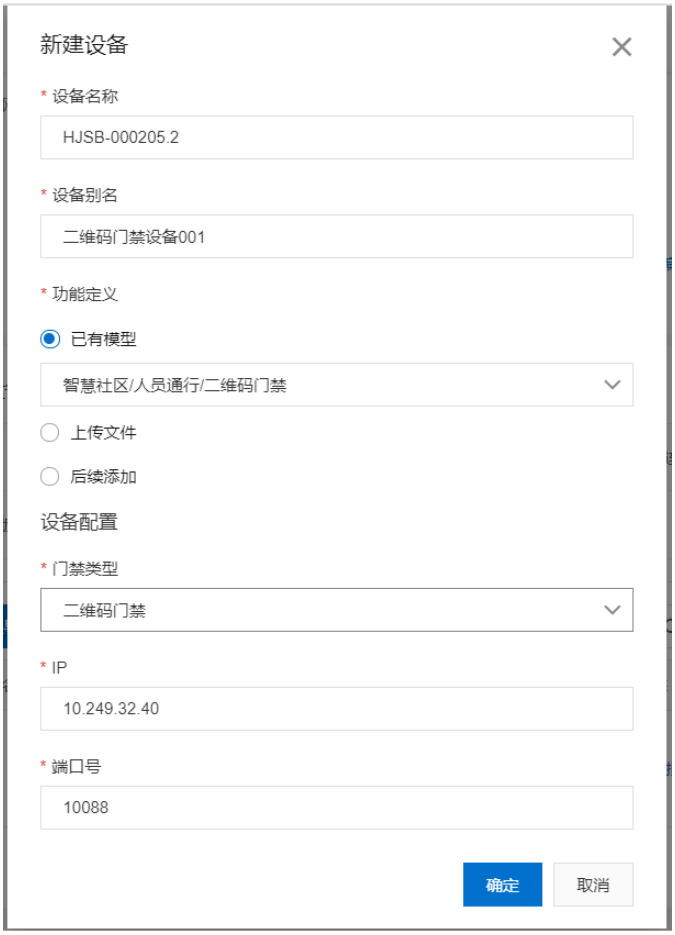


iii. 在官方协议下新增二维码门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001

c. **功能定义**：狄耐克门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



d. **门禁类型**：选择“二维码门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

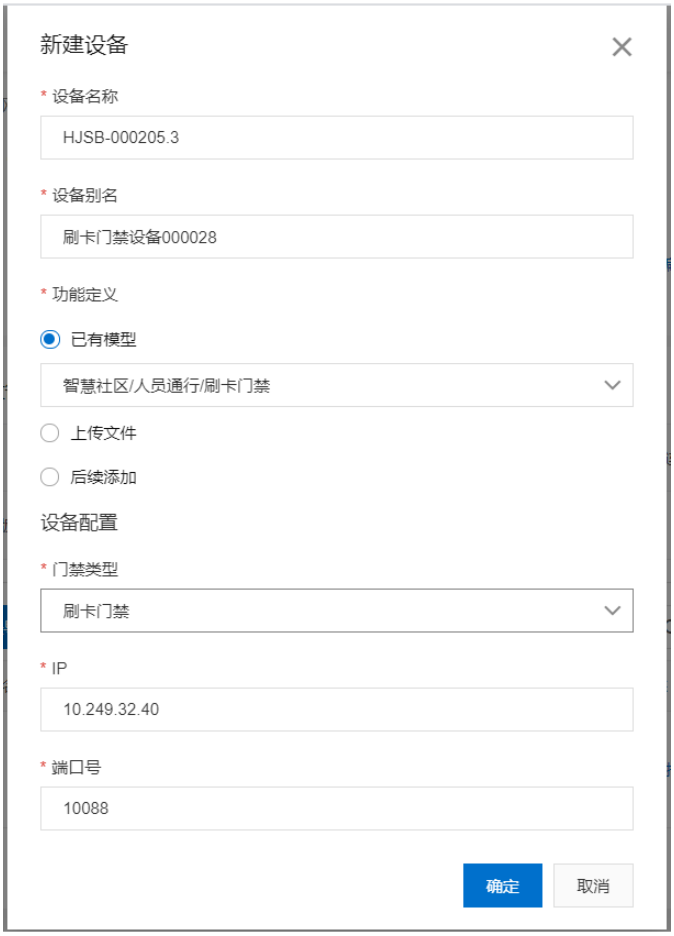
f. **端口号**：狄耐克门禁设备对外提供的网络访问端口号固定是**10088**

iv. 在官方协议下新增刷卡门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中"**HJSB-000205**"是该设备的IoT编码，".**3**"标识刷卡门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028

c. **功能定义**：狄耐克门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-刷卡门禁。



d. **门禁类型**：选择“刷卡门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40

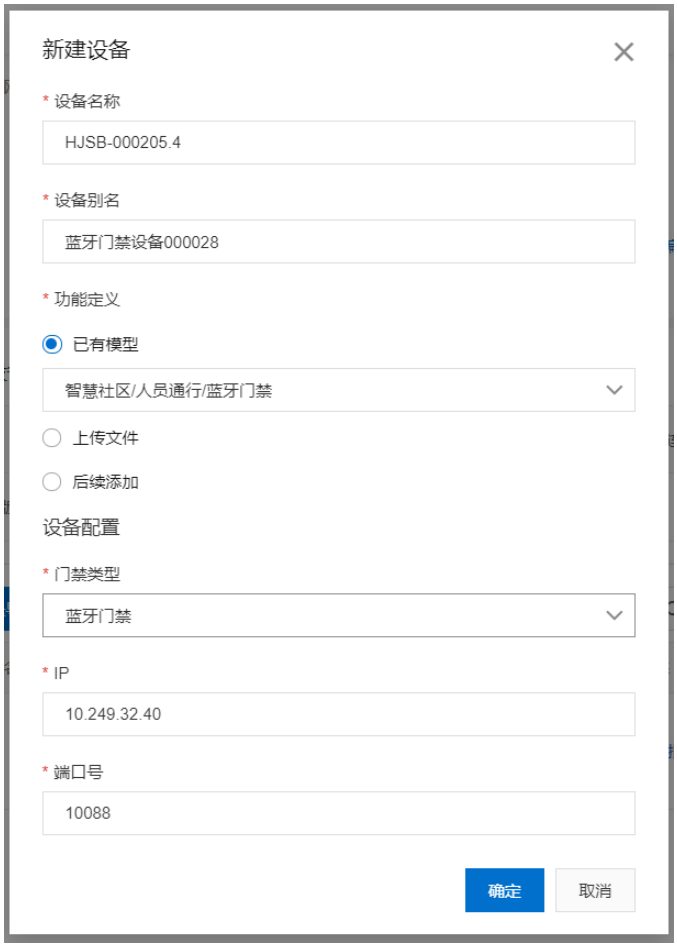
f. **端口号**：狄耐克门禁设备对外提供的网络访问端口号固定是**10088**

v. 在官方协议下新增蓝牙门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。蓝牙门禁配置示例为：HJSB-000205.4。其中"**HJSB-000205**"是该设备的IoT编码，"**.4**"标识蓝牙门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：蓝牙门禁设备000028

c. **功能定义**：狄耐克门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-蓝牙门禁。



d. **门禁类型**：选择“蓝牙门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

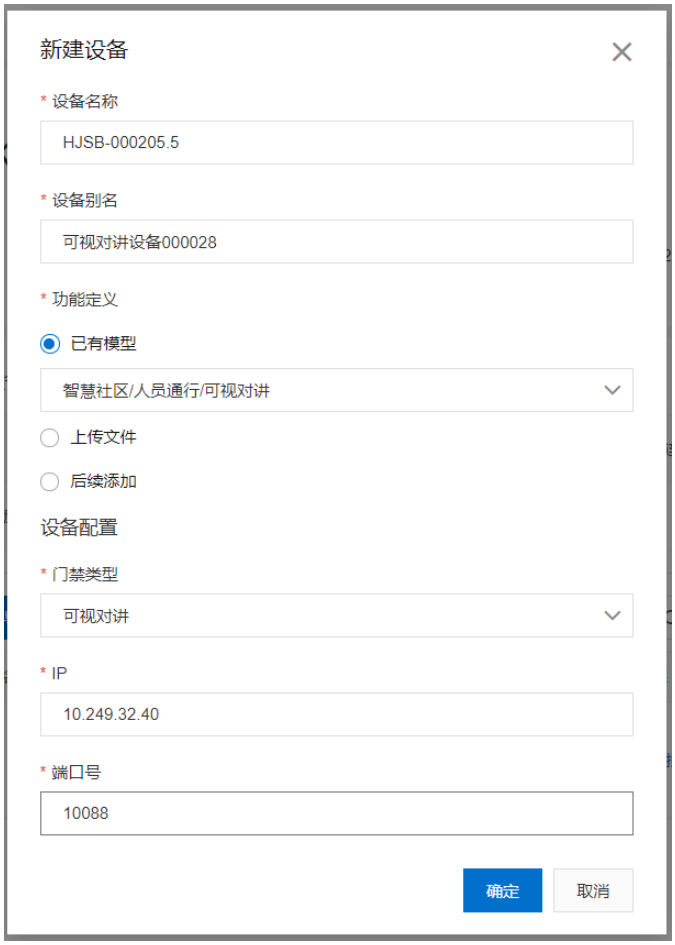
f. **端口号**：狄耐克门禁设备对外提供的网络访问端口号固定是**10088**

vi. 在官方协议下新增可视对讲设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。可视对讲设备配置示例为：HJSB-000205.5。其中"**HJSB-000205**"是该设备的IoT编码，".5"标识可视对讲设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：可视对讲设备000028

c. **功能定义**：选择已有模型，对应设备名称选择智慧社区-人员通行-可视对讲。



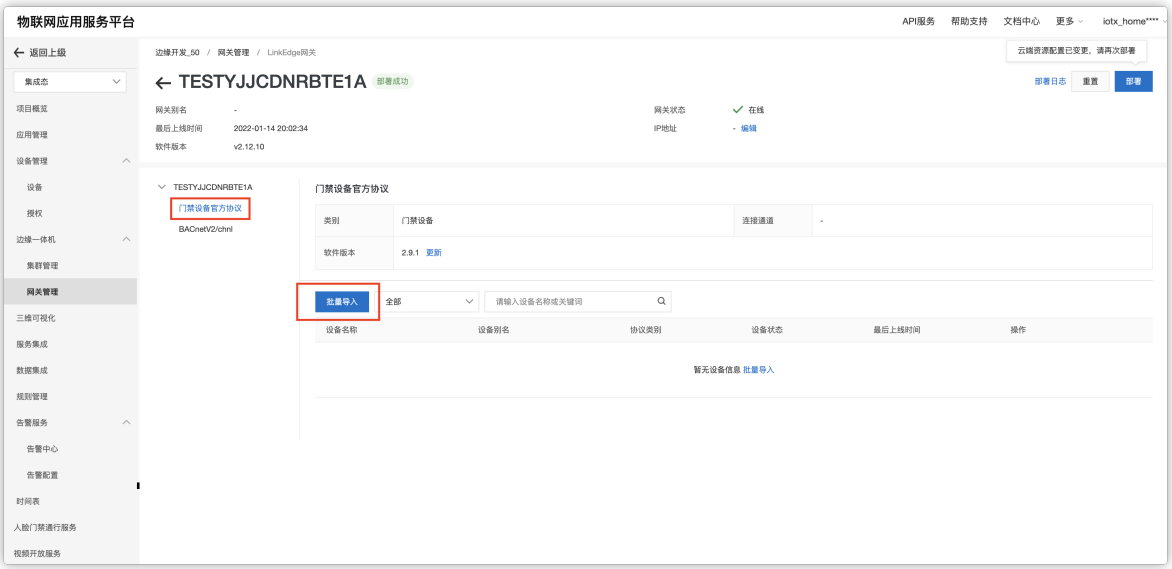
d. **门禁类型**：选择“可视对讲”

e. **IP**:参照甲方提供的<设备信息表>中的该可视对讲设备IP。输入示例为10.249.32.40

f. **端口号**：**狄耐克**门禁设备对外提供的可视对讲网络访问端口号固定是**10088**

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

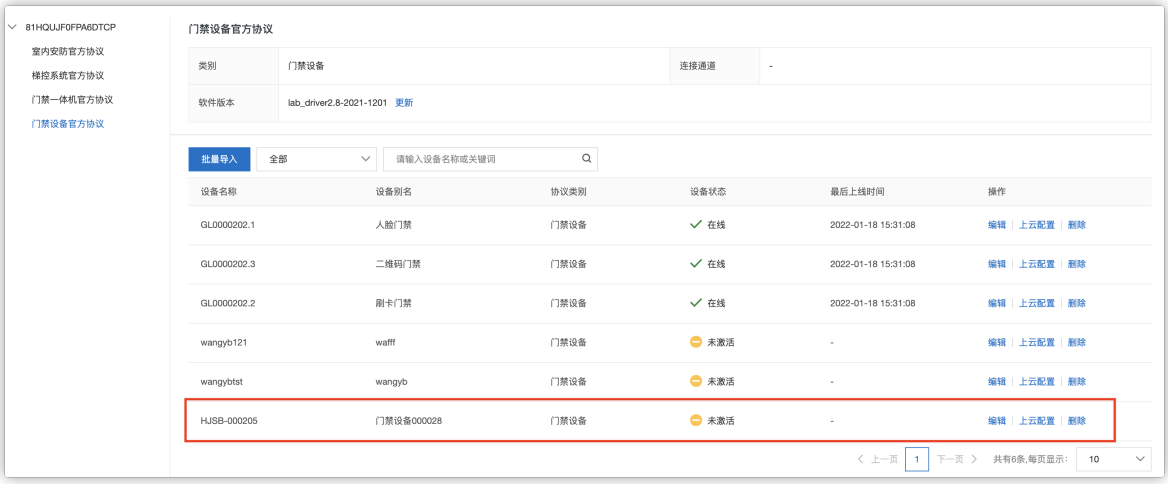
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.8. 大华人行设备接入

说明

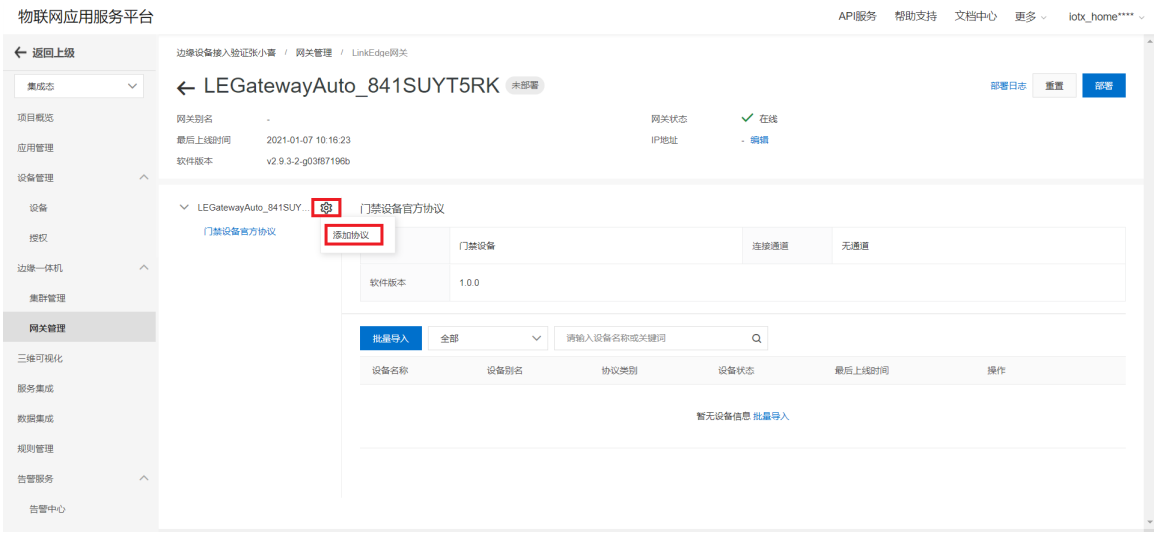
大华支持的型号

对接型号及功能	1、DH-ASI7213Y-V3：人脸(边侧)，二维码，刷卡2、DH-ASI7214Y-V3：人脸(边侧)，二维码，刷卡3、DH-ASI7213X：人脸(边侧)，二维码，刷卡4、DH-ASI7214X：人脸(边侧)，二维码，刷卡5、DH-ASI7223X-A：人脸(边侧)，二维码，刷卡6、DH-ASI7223Y-A-V3：人脸(端侧)，二维码7、DH-SH-MB9833Z-F：人脸(边侧)，二维码8、DH-VTO65：人脸(边侧)，二维码，刷卡，可视对讲9、DH-VTO75：人脸(边侧)，二维码，刷卡，可视对讲10、DH-VTO95：人脸(边侧)，二维码，刷卡，可视对讲
接入模式	门禁设备
支持功能	人脸/二维码/刷卡/可视对讲
系统版本	N/A
人脸边侧识别支持型号	其他型号
人脸端侧识别支持型号	DH-ASI7223Y-A-V3。端口号待补充
门禁端口号	80
可视对讲端口号	80

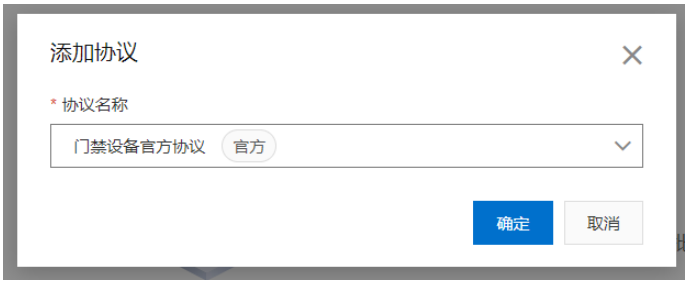
1. 添加门禁协议
   i. 第一次添加协议，可以点击**新建协议**按钮，如下图所示



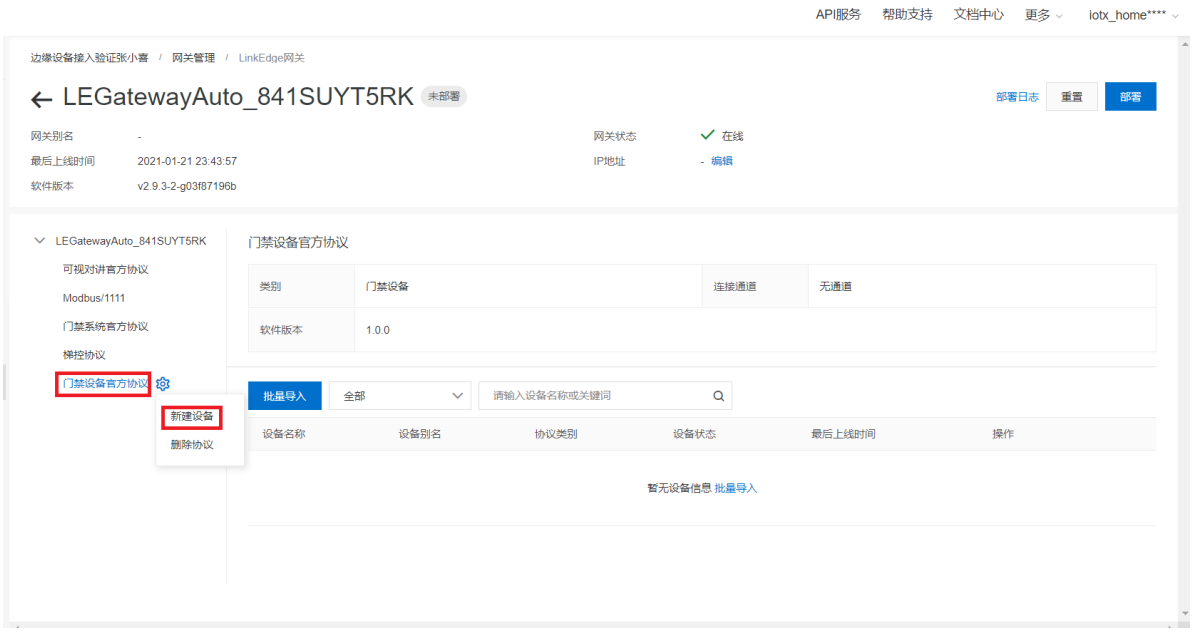
ii. 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



iii. **大华**的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示



2. 添加门禁设备，如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。



i. 在官方协议下新增人脸门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，"**.1**"标识人脸门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028

c. **功能定义**：**大华**门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"



d. **门禁类型**：选择“**人脸门禁**”

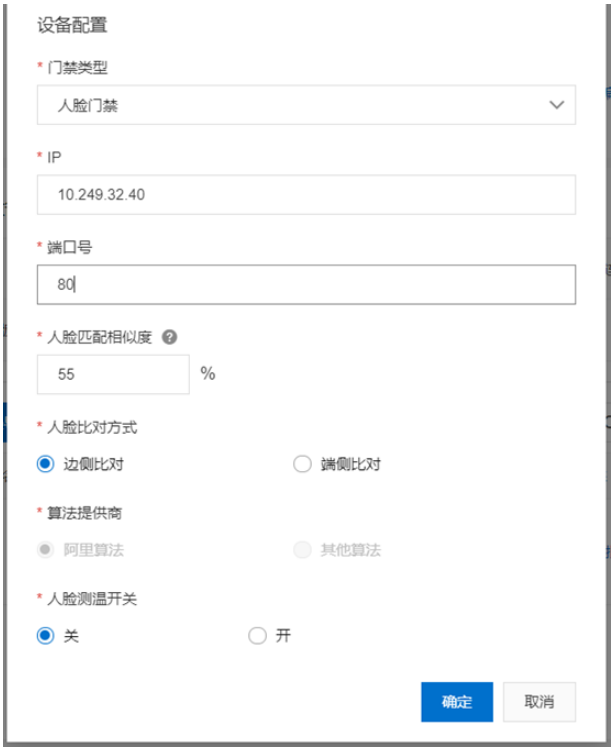
e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

f. **端口号**：**大华**门禁设备对外提供的网络访问端口号固定是80

g. **人脸匹配相似度**：输入55%

h. **人脸比对方式**：**大华**人脸比对方式，支持两种：**边侧比对**和**端侧比对。请注意与项目上确认具体方式进行选择**

如果是边侧比对：

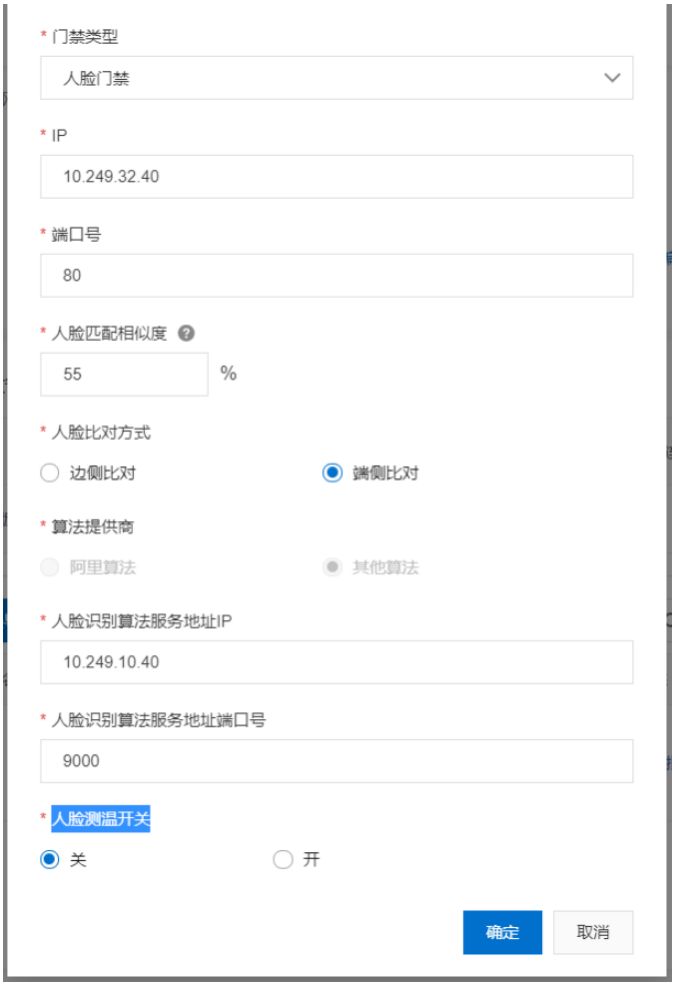


人脸测温开关：大华门禁设备不支持人脸测温，选择关。

如果是端侧比对：

人脸识别算法服务地址IP：大华人脸识别算法服务器的地址IP。特别注意：这个IP地址请与项目联系确认。输入示例为10.249.10.40

人脸识别算法服务地址端口号:大华人脸识别算法服务地址端口号固定为**80**



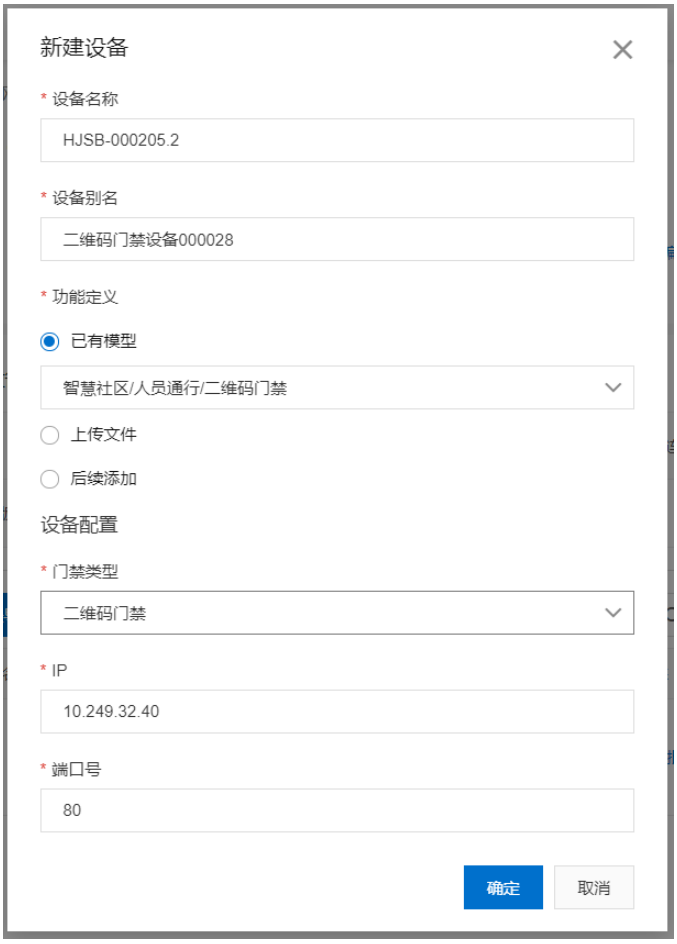
人脸测温开关：大华门禁设备不支持人脸测温，选择关。

如图所示，在官方协议下新增二维码门禁设备。其中：

设备名称：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备

设备别名：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备000028

功能定义：大华门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



门禁类型：选择“二维码门禁”

IP:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

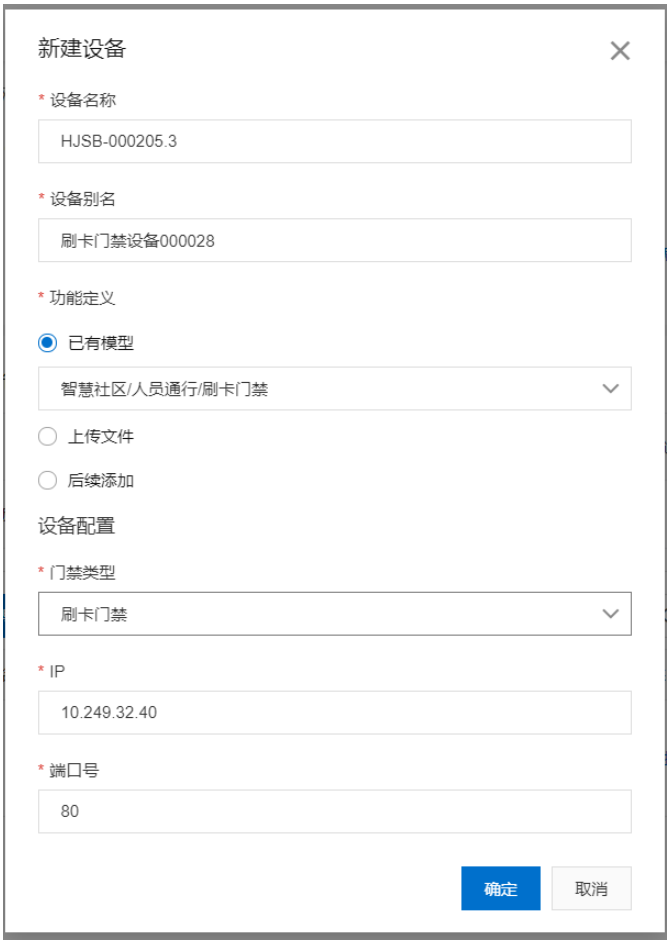
端口号：大华门禁设备对外提供的网络访问端口号固定是80

如图所示，在官方协议下新增刷卡门禁设备。其中：

设备名称：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中"**HJSB-000205**"是该设备的IoT编码，"**.3**"标识刷卡门禁设备

设备别名：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028

功能定义：大华门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-刷卡门禁。



门禁类型：选择“刷卡门禁”

IP:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40

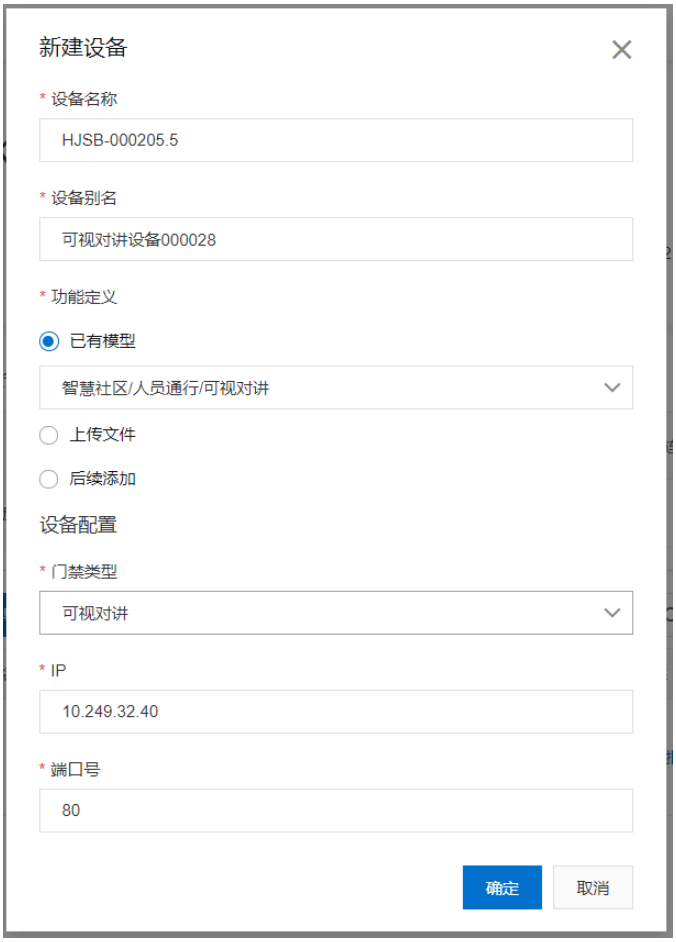
端口号：大华门禁设备对外提供的网络访问端口号固定是80

如图所示，在官方协议下新增可视对讲设备。其中：

设备名称：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。可视对讲设备配置示例为：HJSB-000205.5。其中"**HJSB-000205**"是该设备的IoT编码，"**.5**"标识可视对讲设备

设备别名：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：可视对讲设备000028

功能定义：选择已有模型，对应设备名称选择智慧社区-人员通行-可视对讲。



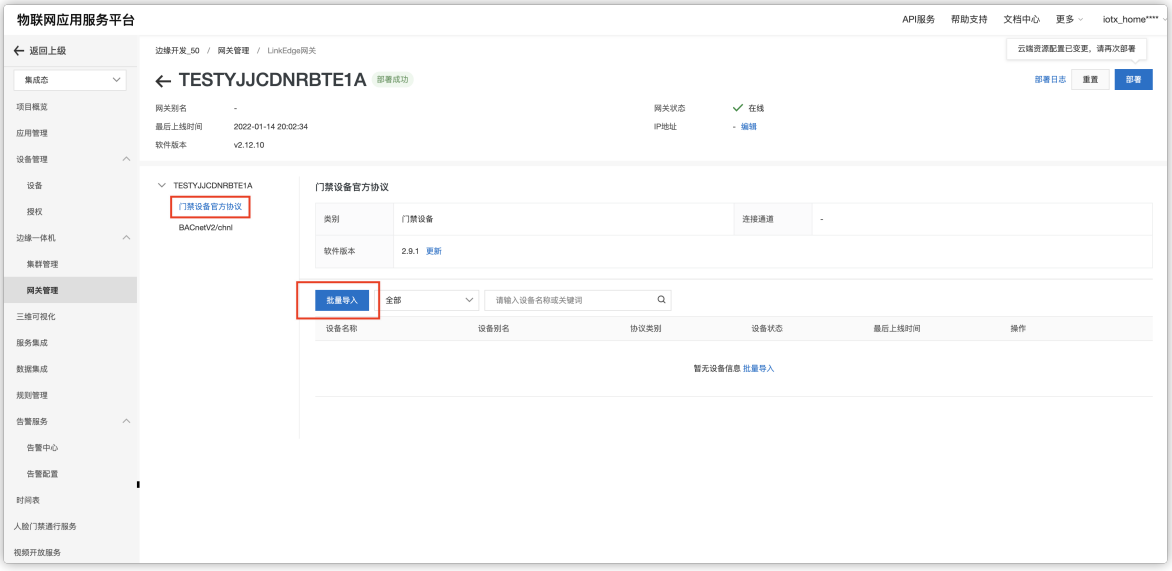
门禁类型：选择“可视对讲”

IP:参照甲方提供的<设备信息表>中的该可视对讲设备IP。输入示例为10.249.32.40

端口号：**大华**门禁设备对外提供的可视对讲网络访问端口号固定是**80**

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

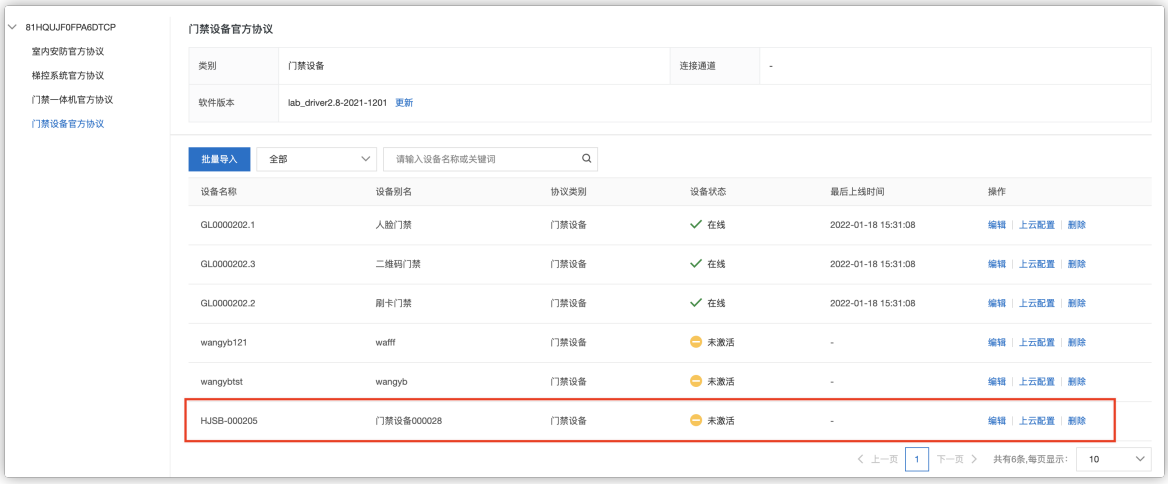
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.9. 艾特人行设备接入

> 说明
>
> 艾特支持的型号

对接型号支持	MK101，MK102，MK080
接入模式	门禁设备
支持功能	人脸/二维码/蓝牙/刷卡/可视对讲
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	10000
可视对讲端口号	？？？

1. 添加门禁协议
   i. 第一次添加协议，可以点击**新建协议**按钮，如下图所示



   ii. 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**

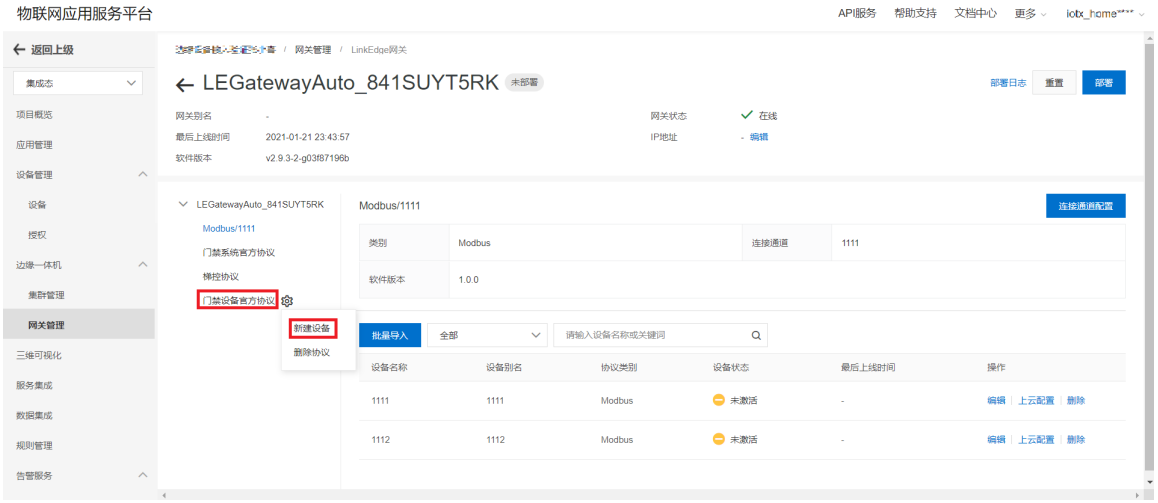


iii. 艾特的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示



2. 添加门禁设备

i. 如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。

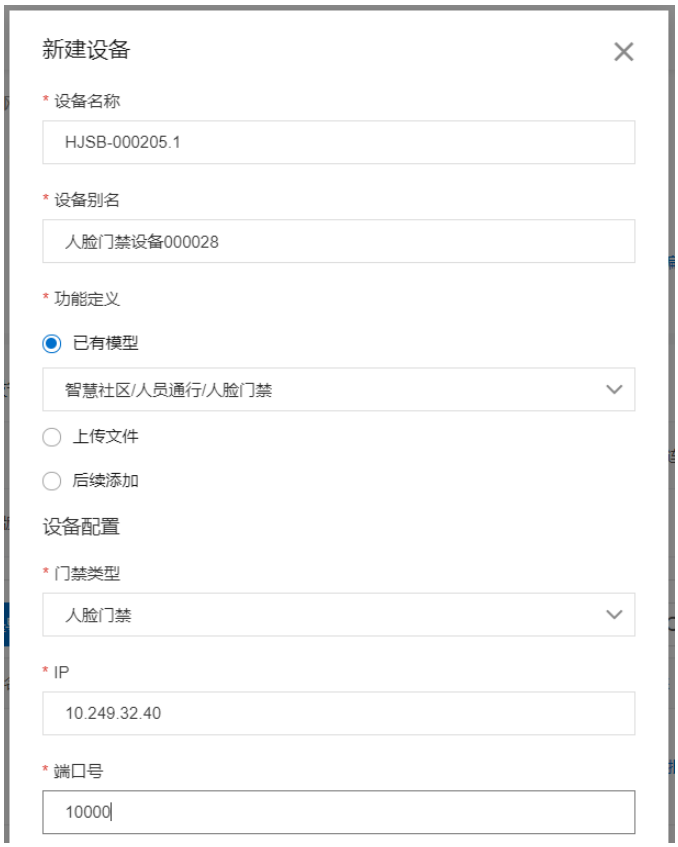


ii. 在官方协议下新增人脸门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，"**.1**"标识人脸门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028

c. **功能定义**：艾特门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"



d. **门禁类型**：选择“**人脸门禁**”

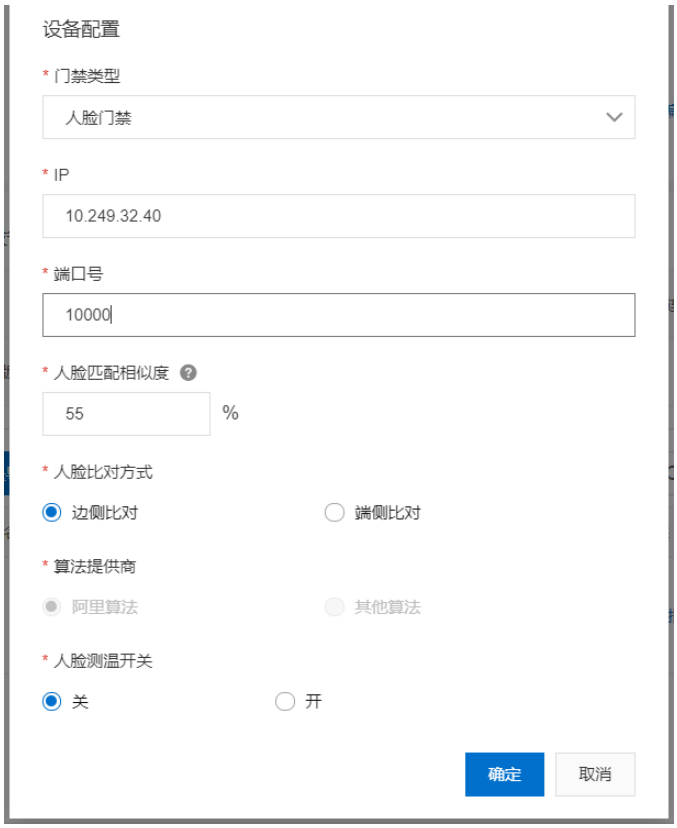
e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

f. **端口号**：**艾特**门禁设备对外提供的网络访问端口号固定是**10000**

g. **人脸匹配相似度**：输入**55%**。

h. **人脸比对方式**：**艾特**人脸比对方式为**边侧比对**

i. **人脸测温开关：艾特**门禁设备不支持人脸测温，设置为关

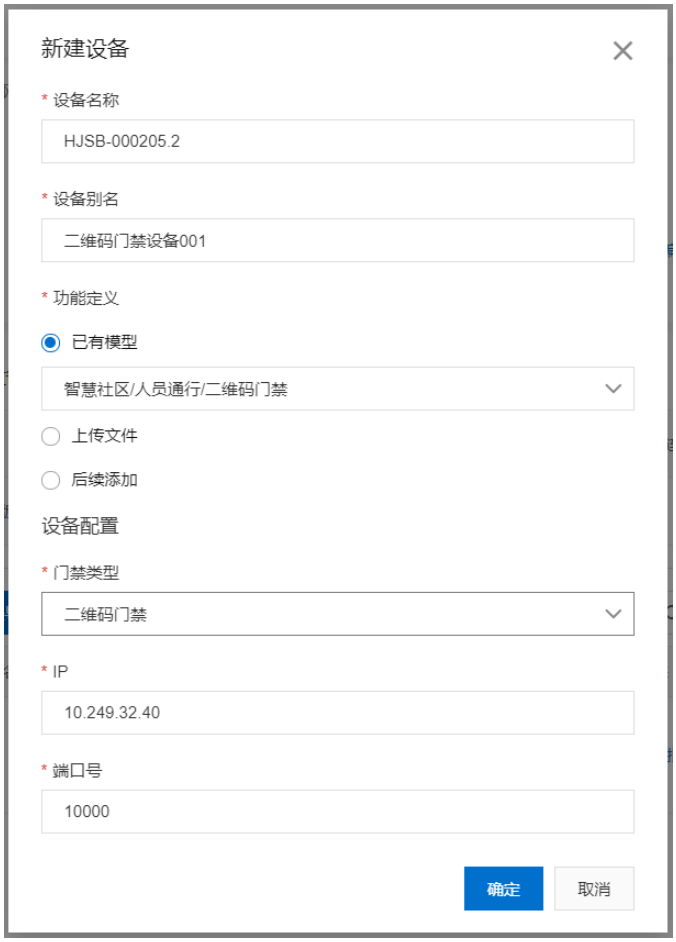


iii. 在官方协议下新增二维码门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001

c. **功能定义**：门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



d. **门禁类型**：选择“二维码门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

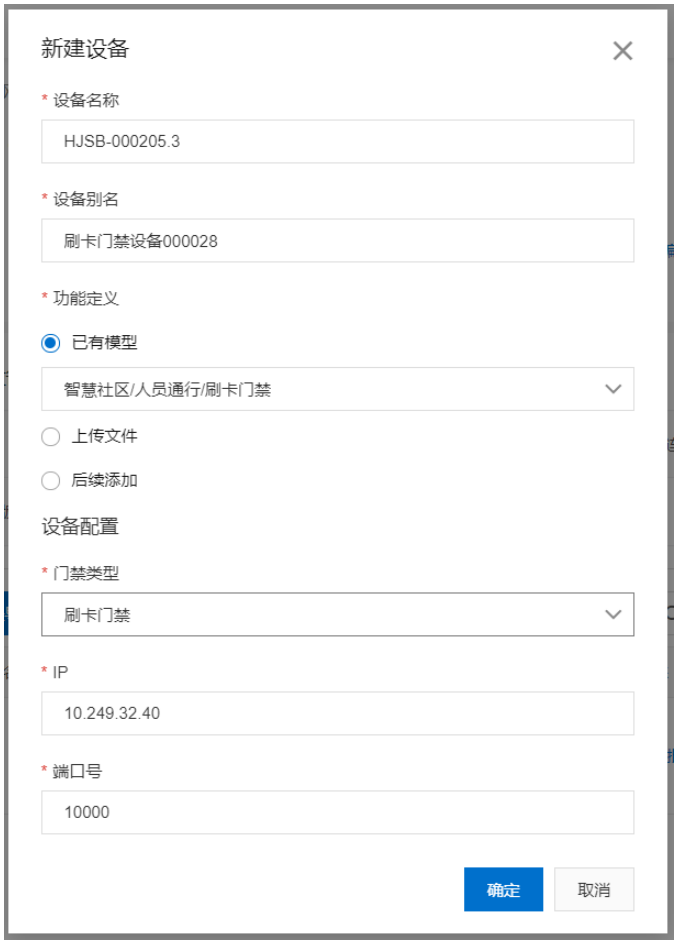
f. **端口号**：艾特门禁设备对外提供的网络访问端口号固定是10000

iv. 在官方协议下新增刷卡门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中"**HJSB-000205**"是该设备的IoT编码，".**3**"标识刷卡门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028

c. **功能定义**：艾特门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-刷卡门禁。



d. **门禁类型**：选择“刷卡门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40

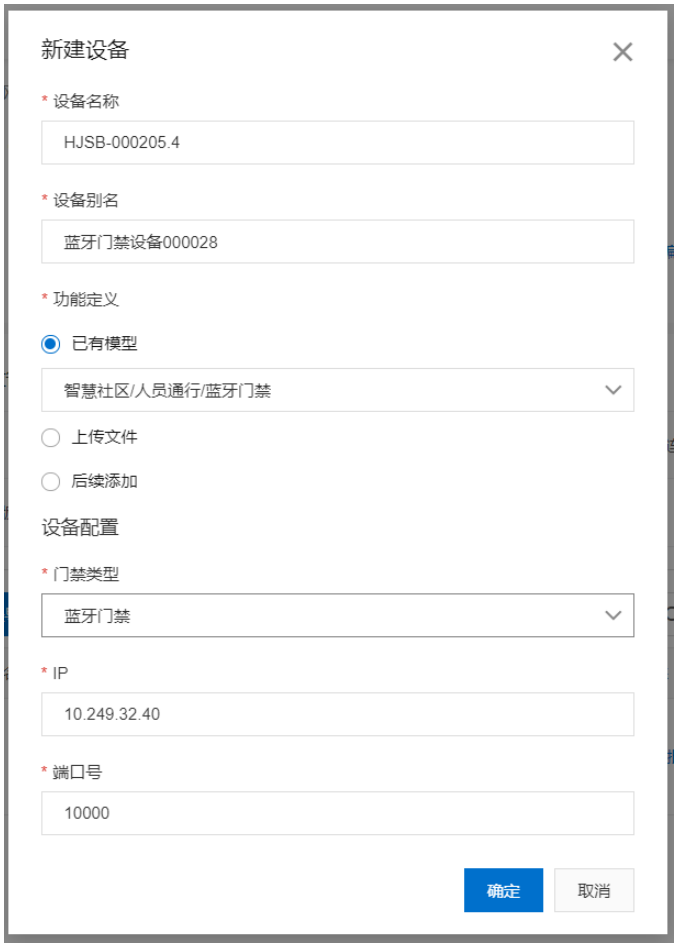
f. **端口号**：艾特门禁设备对外提供的网络访问端口号固定是10000

v. 在官方协议下新增蓝牙门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。蓝牙门禁配置示例为：HJSB-000205.4。其中"**HJSB-000205**"是该设备的IoT编码，"**.4**"标识蓝牙门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：蓝牙门禁设备000028

c. **功能定义**：艾特门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-蓝牙门禁。



d. **门禁类型**：选择“蓝牙门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

f. **端口号**：艾特门禁设备对外提供的网络访问端口号固定是**10000**

vi. 在官方协议下新增可视对讲设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。可视对讲设备配置示例为：HJSB-000205.5。其中"**HJSB-000205**"是该设备的IoT编码，".5"标识可视对讲设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：可视对讲设备000028

c. **功能定义**：选择已有模型，对应设备名称选择智慧社区-人员通行-可视对讲。



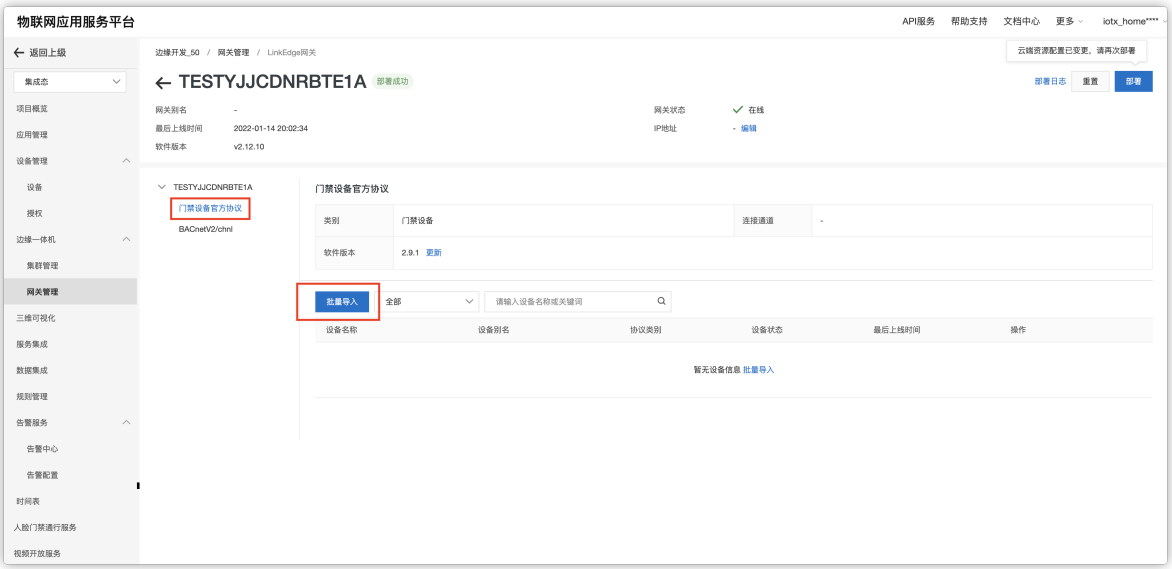
d. **门禁类型**：选择“可视对讲”

e. **IP**:参照甲方提供的<设备信息表>中的该可视对讲设备IP。输入示例为10.249.32.40

f. **端口号**：**艾特门禁设备可视对讲对外提供的访问端口号需要与艾特确认**。输入示例为10060

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

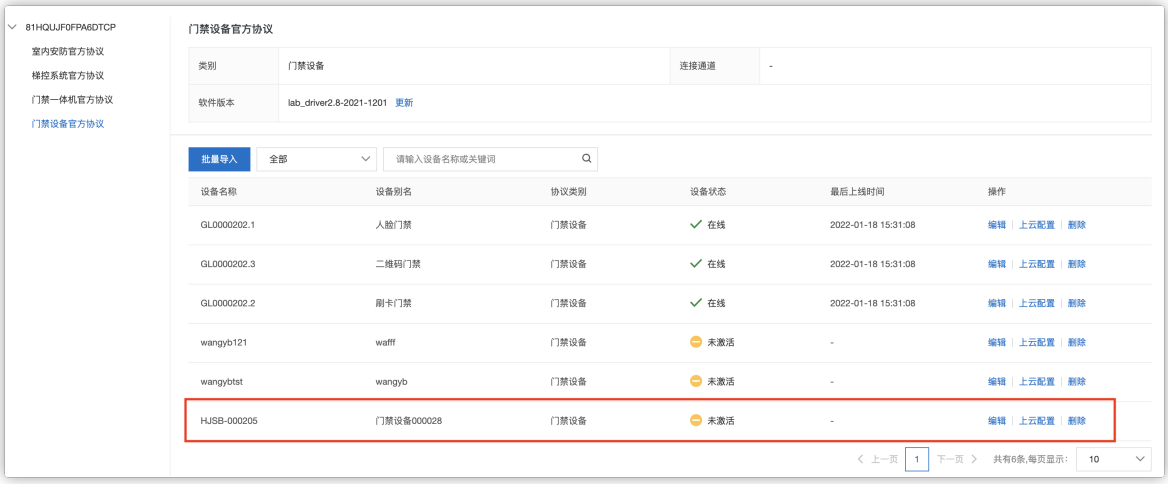
其中**设备名称**、**设备别名**、**设备品类**、**门禁类型**、**IP**、**端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.10. 冠林人行设备接入

说明

冠林支持的型号

对接型号支持	GL8-K3CSGL8-T3CSGL8-K4CS
接入模式	门禁设备
支持功能	人脸/二维码/刷卡/可视对讲
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	10020
可视对讲端口号	10060

1. 添加门禁协议

i. 第一次添加协议，可以点击**新建协议**按钮，如下图所示



ii. 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**

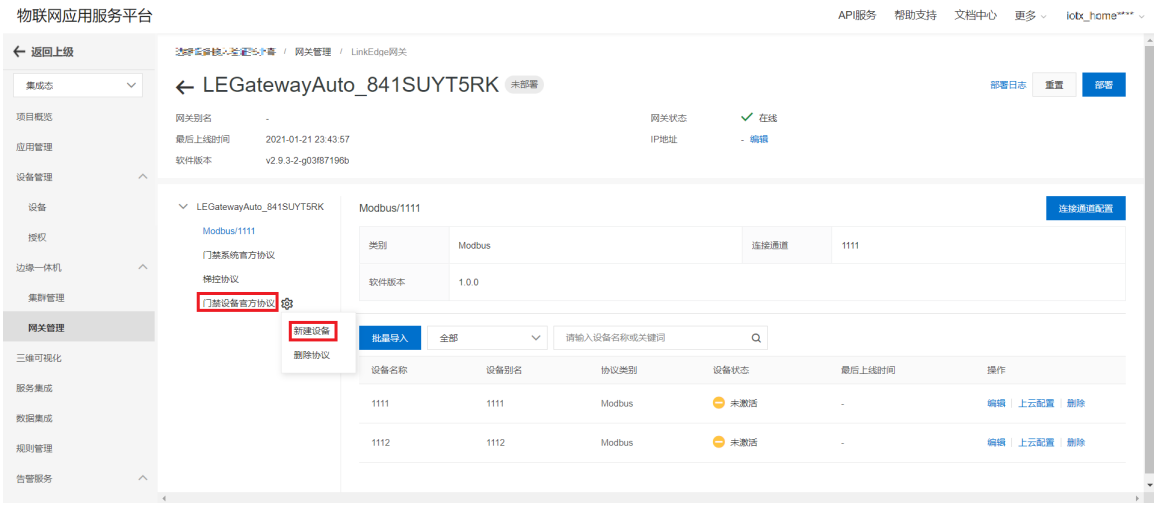


iii. 冠林的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示



2. 添加门禁设备

i. 鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。

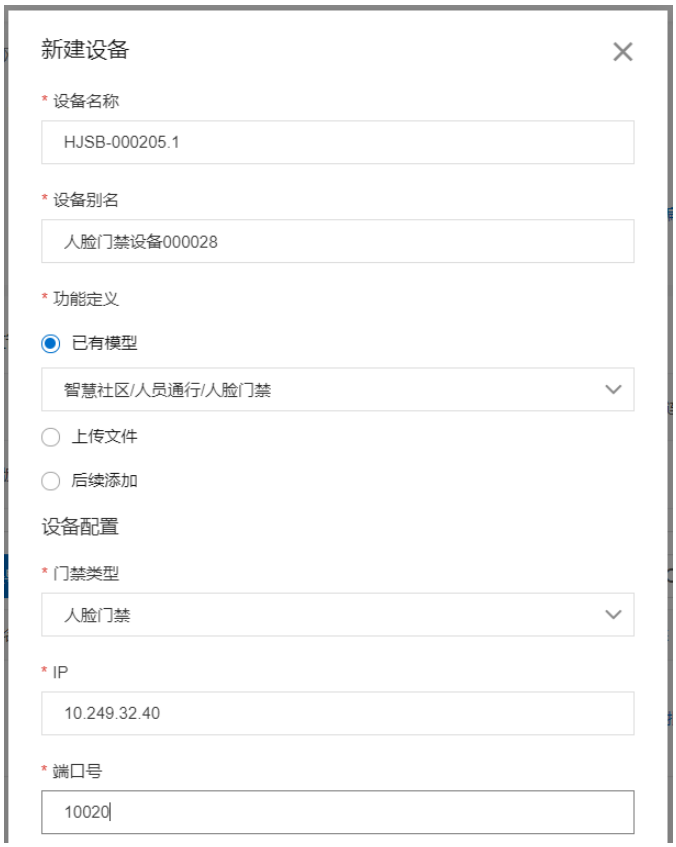


ii. 在官方协议下新增人脸门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，"**.1**"标识人脸门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028

c. **功能定义**：冠林门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"



d. **门禁类型**：选择“**人脸门禁**”

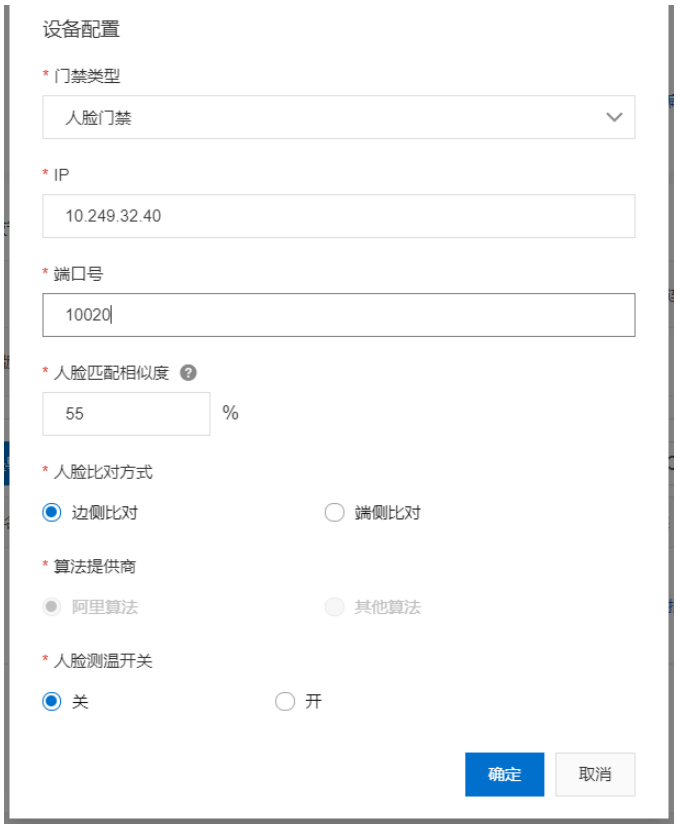
e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

f. **端口号**：冠林门禁设备对外提供的网络访问端口号固定是10020

g. **人脸匹配相似度**：输入**55%**。

h. **人脸比对方式**：**冠林**人脸比对方式为**边侧比对**

i. **人脸测温开关：冠林**门禁设备不支持人脸测温，设置为关

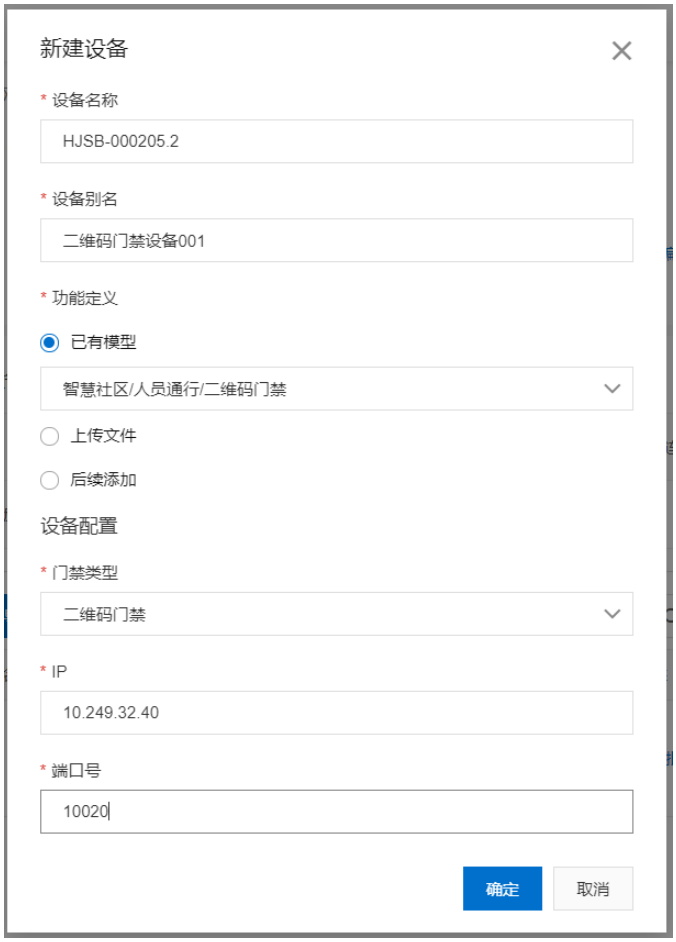


iii. 在官方协议下新增二维码门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001

c. **功能定义**：冠林门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



d. **门禁类型**：选择“二维码门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

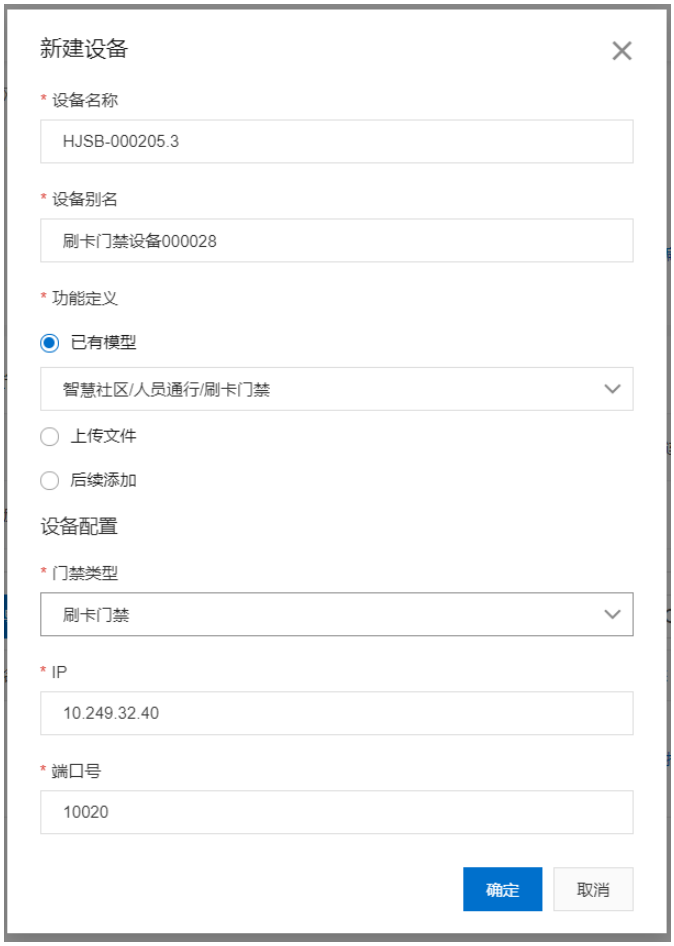
f. **端口号**：冠林门禁设备对外提供的网络访问端口号固定是10020

iv. 在官方协议下新增刷卡门禁设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中"**HJSB-000205**"是该设备的IoT编码，"**.3**"标识刷卡门禁设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028

c. **功能定义**：冠林门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-刷卡门禁。



d. **门禁类型**：选择“刷卡门禁”

e. **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40

f. **端口号**：冠林门禁设备对外提供的网络访问端口号固定是10020

v. 在官方协议下新增可视对讲设备。其中：

a. **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。可视对讲设备配置示例为：HJSB-000205.5。其中"**HJSB-000205**"是该设备的IoT编码，".5"标识可视对讲设备

b. **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：可视对讲设备000028

c. **功能定义**：选择已有模型，对应设备名称选择智慧社区-人员通行-可视对讲。



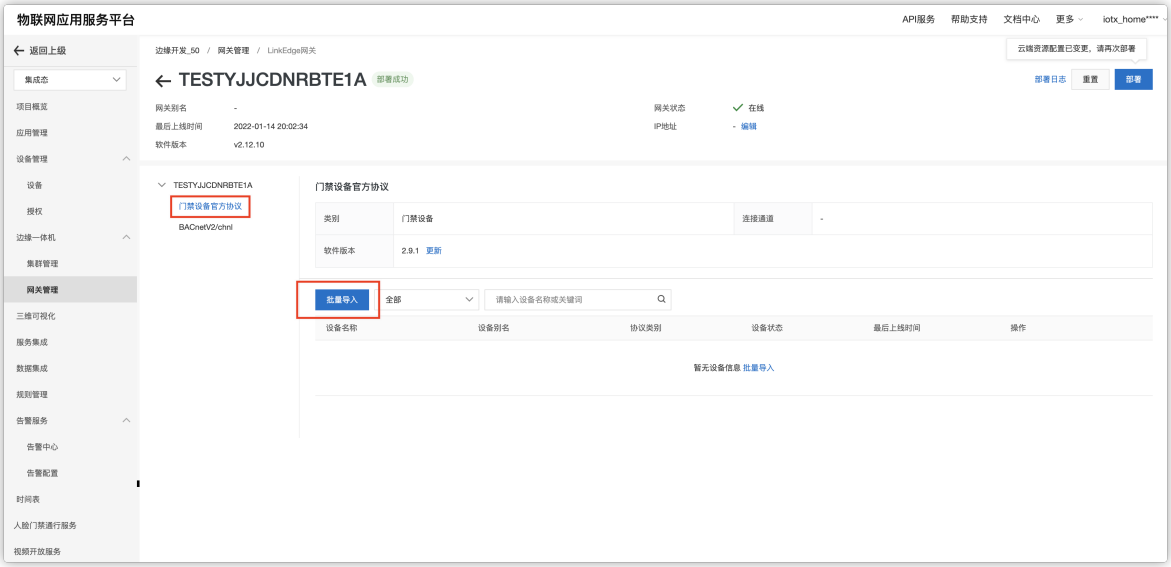
d. **门禁类型**：选择“可视对讲”

e. **IP**:参照甲方提供的<设备信息表>中的该可视对讲设备IP。输入示例为10.249.32.40

f. **端口号**：**冠林**门禁设备对外提供的可视对讲网络访问端口号固定是**10060**

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

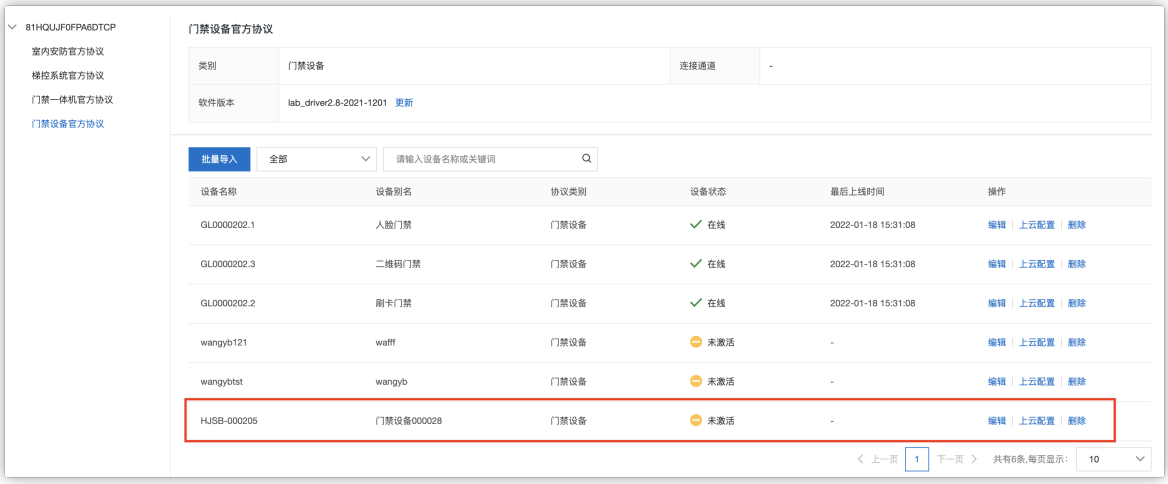
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.11. 中证科技人行设备

说明

中证科技支持的型号

对接型号支持	H-OS06
接入模式	门禁设备
支持功能	人脸/二维码/刷卡/可视对讲
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	10020
可视对讲端口号	10060

门禁设备集成

1. 添加门禁协议
   - 第一次添加协议，可以点击**新建协议**按钮，如下图所示



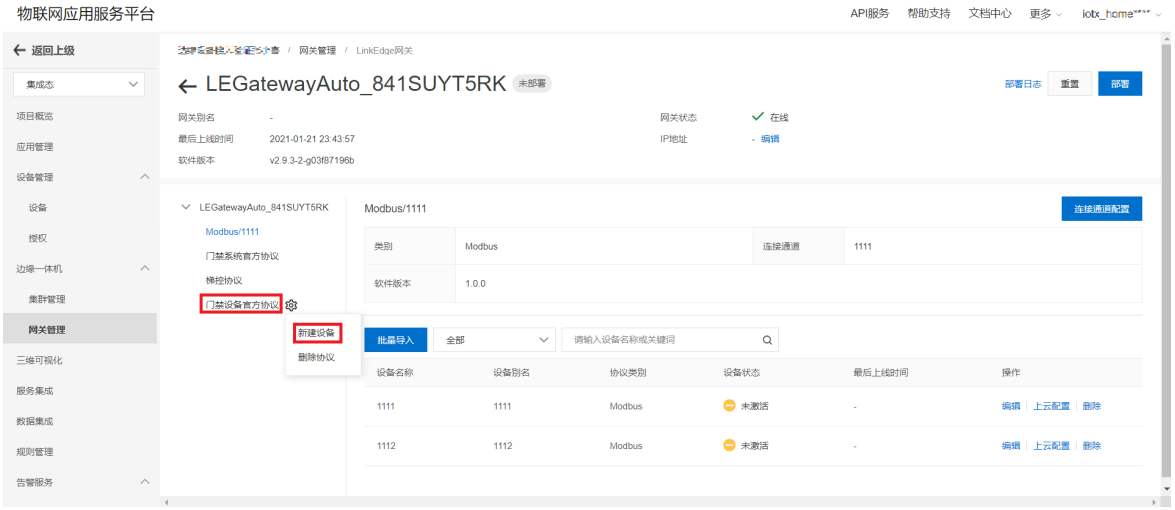
   - 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



- 冠林的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示

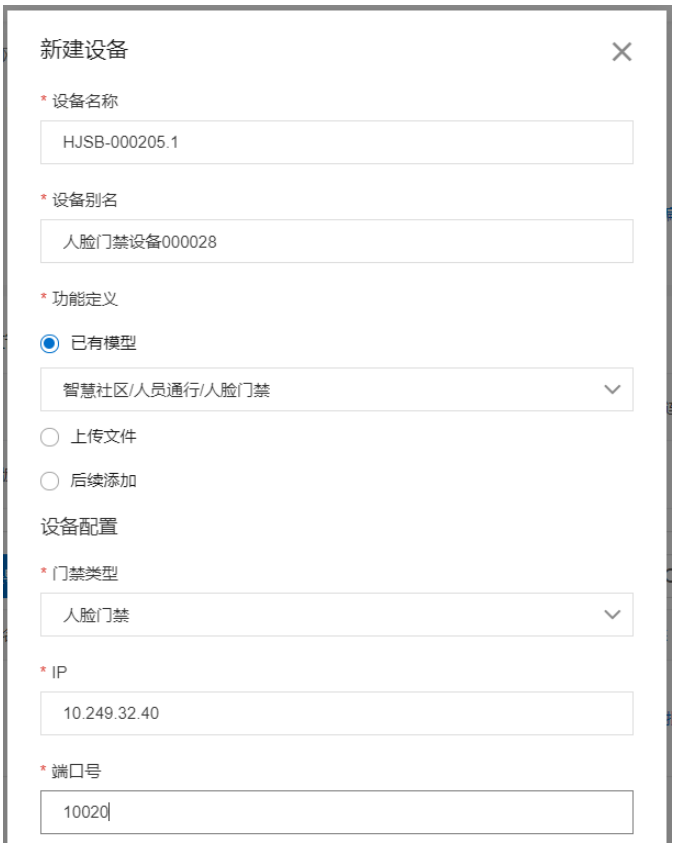


2. 添加门禁设备
   - 如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。



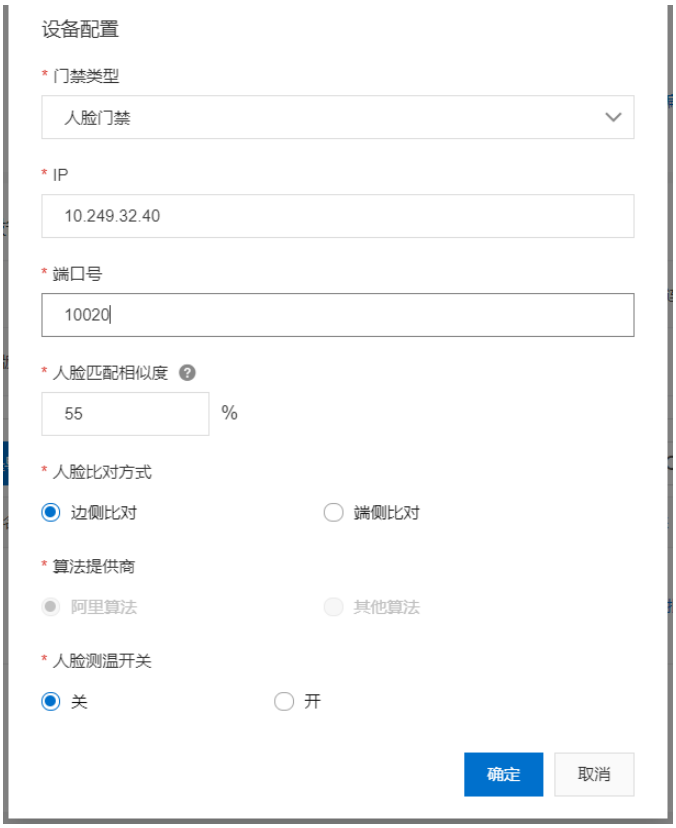
   - 如图所示，在官方协议下新增人脸门禁设备。其中：
     - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，"**.1**"标识人脸门禁设备
     - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028

- **功能定义**：中证科技门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"



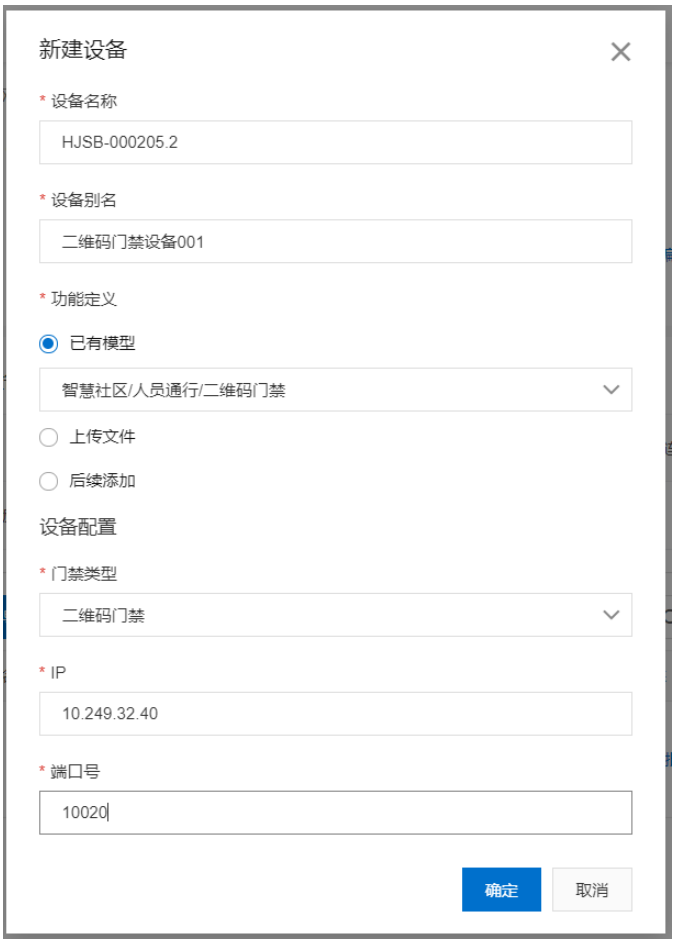
- **门禁类型**：选择“**人脸门禁**”
- **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
- **端口号**：中证科技门禁设备对外提供的网络访问端口号固定是10020
- **人脸匹配相似度**：输入**55%**。
- **人脸比对方式**：中证科技人脸比对方式为**边侧比对**

- **人脸测温开关**：中证科技门禁设备不支持人脸测温，设置为关



- 如图所示，在官方协议下新增二维码门禁设备。其中：
  - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备
  - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001

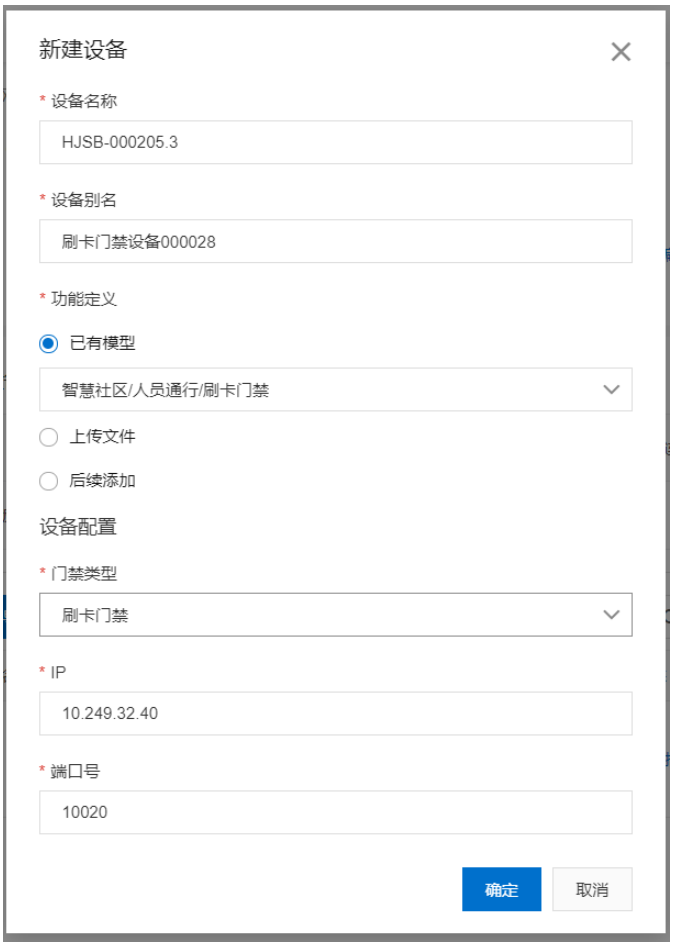
- **功能定义**：中证科技门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



- **门禁类型**：选择“二维码门禁”
- **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
- **端口号**：中证科技门禁设备对外提供的网络访问端口号固定是10020

- 如图所示，在官方协议下新增刷卡门禁设备。其中：
  - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中"**HJSB-000205**"是该设备的IoT编码，".3"标识刷卡门禁设备
  - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028

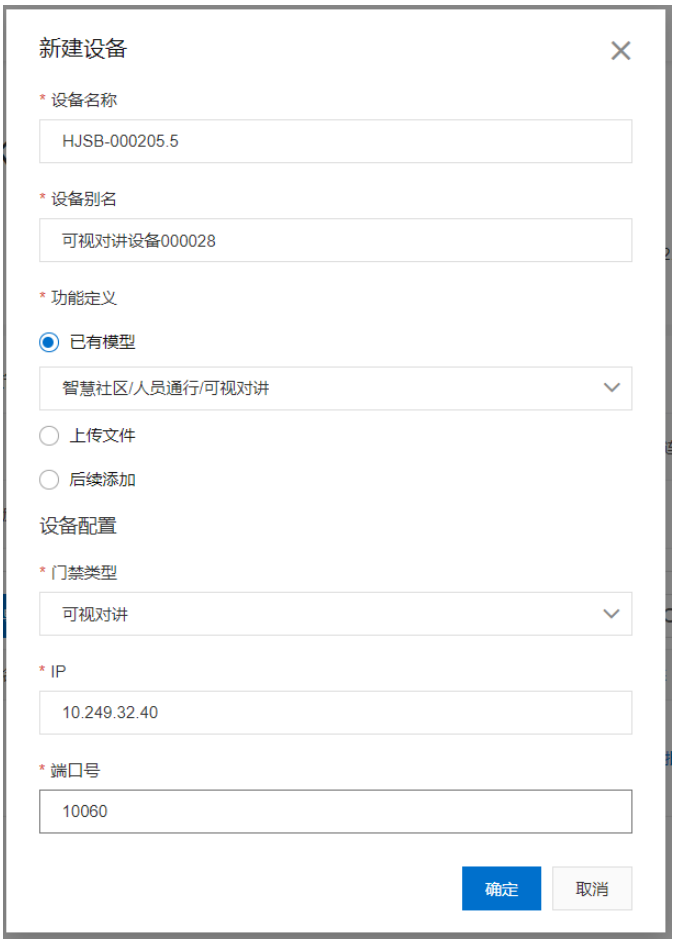
- **功能定义**：中证科技门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-刷卡门禁。



- **门禁类型**：选择“刷卡门禁”
- **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40
- **端口号**：中证科技门禁设备对外提供的网络访问端口号固定是10020

- 如图所示，在官方协议下新增可视对讲设备。其中：
  - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。可视对讲设备配置示例为：HJSB-000205.5。其中"**HJSB-000205**"是该设备的IoT编码，"**.5**"标识可视对讲设备
  - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：可视对讲设备000028

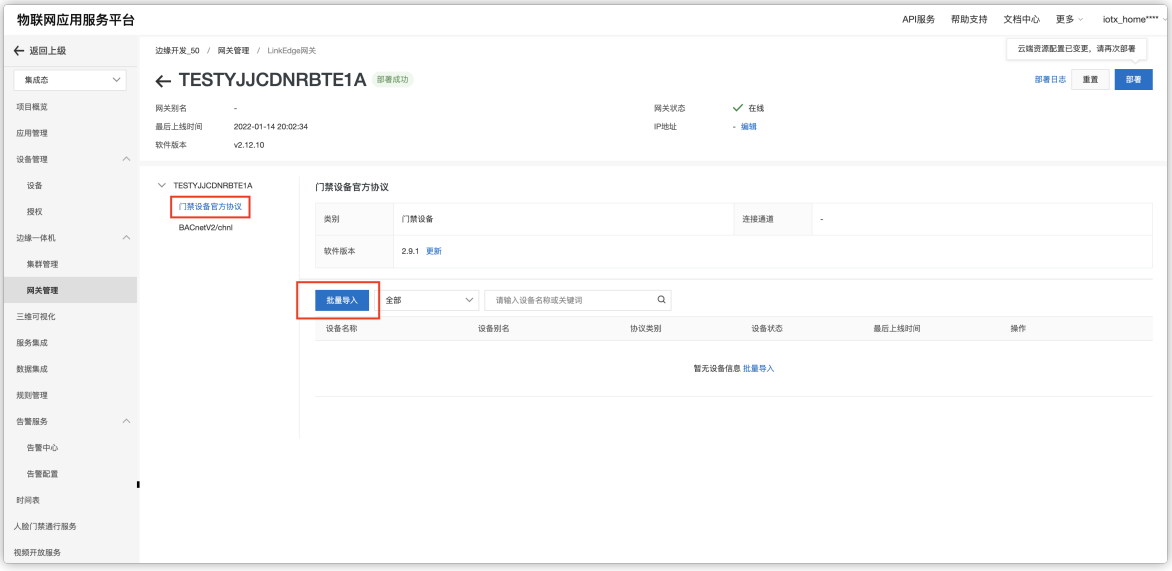
- **功能定义**：选择已有模型，对应设备名称选择智慧社区-人员通行-可视对讲。



- **门禁类型**：选择“可视对讲”
- **IP**:参照甲方提供的<设备信息表>中的该可视对讲设备IP。输入示例为10.249.32.40
- **端口号**：**中证科技**门禁设备对外提供的可视对讲网络访问端口号固定是**10060**

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

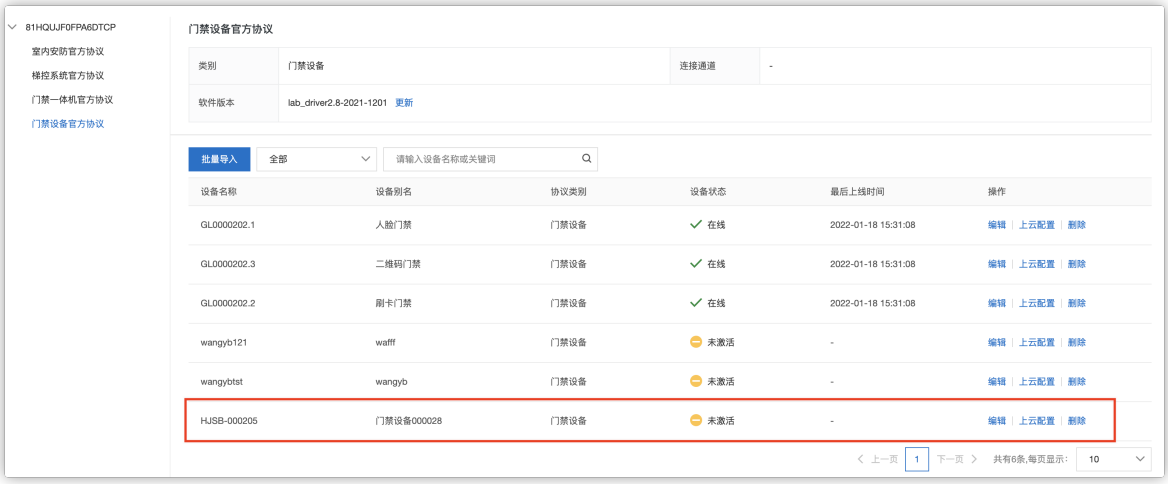
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.12. 太川人行设备

说明

太川支持的型号

对接型号支持	TC-5000D-H（支持刷卡）TC-5000D-B1/WD（不支持刷卡）
接入模式	门禁设备
支持功能	人脸/二维码/刷卡/可视对讲
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	10020
可视对讲端口号	10020

门禁设备集成

1. 添加门禁协议
   - 第一次添加协议，可以点击**新建协议**按钮，如下图所示



   - 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



- 太川的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示

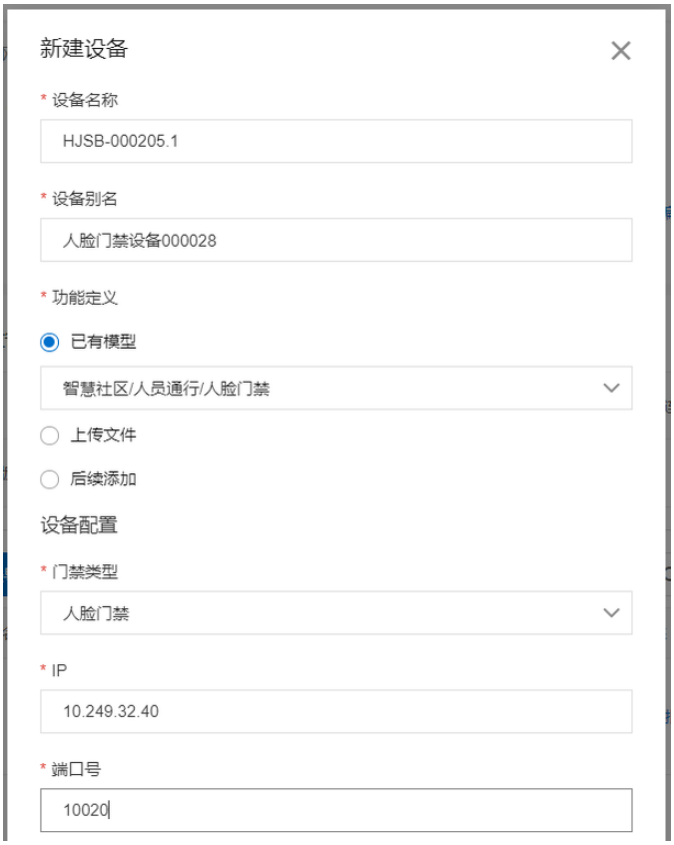


2. 添加门禁设备
   - 第一次添加协议，可以点击**新建协议**按钮，如下图所示



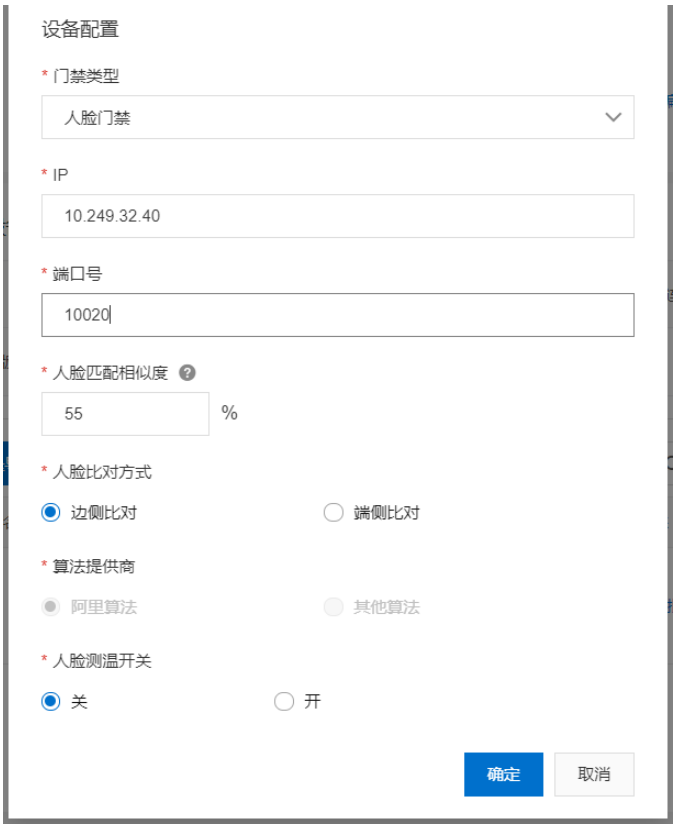
   - 如图所示，在官方协议下新增人脸门禁设备。其中：
     - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，".1"标识人脸门禁设备

- **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028**功能定义**：**太川**门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"



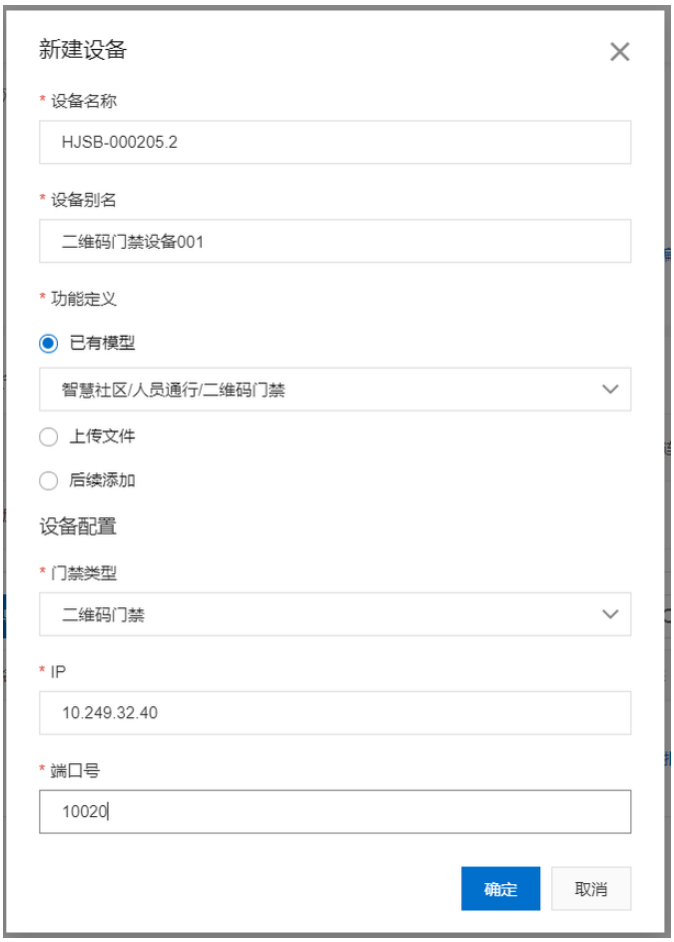
- **门禁类型**：选择“**人脸门禁**”
- **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
- **端口号**：**太川**门禁设备对外提供的网络访问端口号固定是10020
- **人脸匹配相似度**：输入**55%**。
- **人脸比对方式**：**太川**人脸比对方式为**边侧比对**

- **人脸测温开关**：太川门禁设备不支持人脸测温，设置为关



- 如图所示，在官方协议下新增二维码门禁设备。其中：
  - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备
  - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001

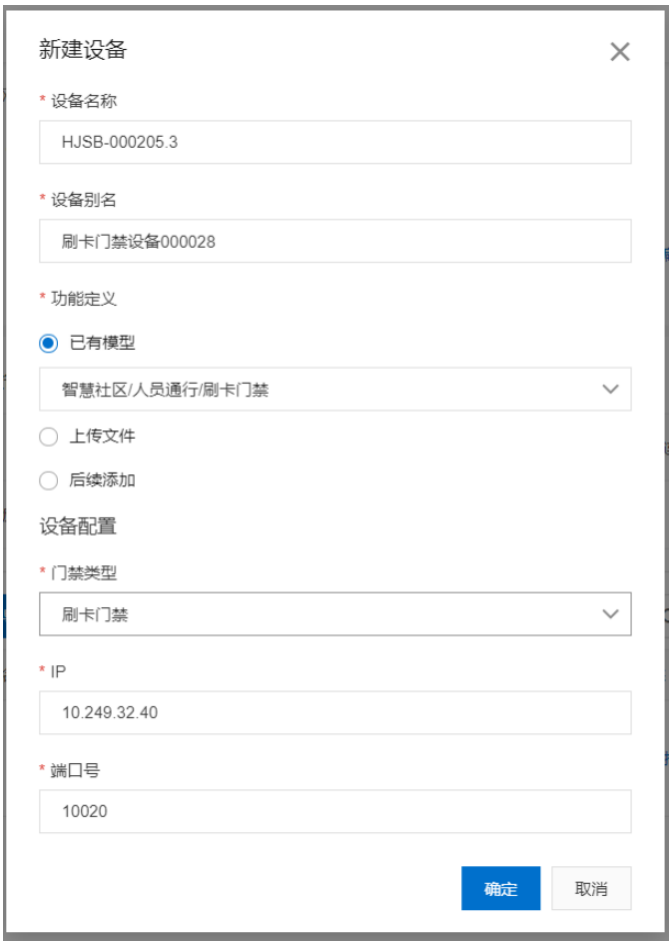
- **功能定义**：**太川**门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



- **门禁类型**：选择“二维码门禁”
- **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
- **端口号**：**太川**门禁设备对外提供的网络访问端口号固定是10020

- 如图所示，在官方协议下新增刷卡门禁设备。其中：
  - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中"**HJSB-000205**"是该设备的IoT编码，".3"标识刷卡门禁设备
  - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028

- **功能定义**：**太川**门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-刷卡门禁。



- **门禁类型**：选择“刷卡门禁”
- **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40
- **端口号**：**太川**门禁设备对外提供的网络访问端口号固定是10020

- 如图所示，在官方协议下新增可视对讲设备。其中：
  - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。可视对讲设备配置示例为：HJSB-000205.5。其中"**HJSB-000205**"是该设备的IoT编码，"**.5**"标识可视对讲设备
  - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：可视对讲设备000028

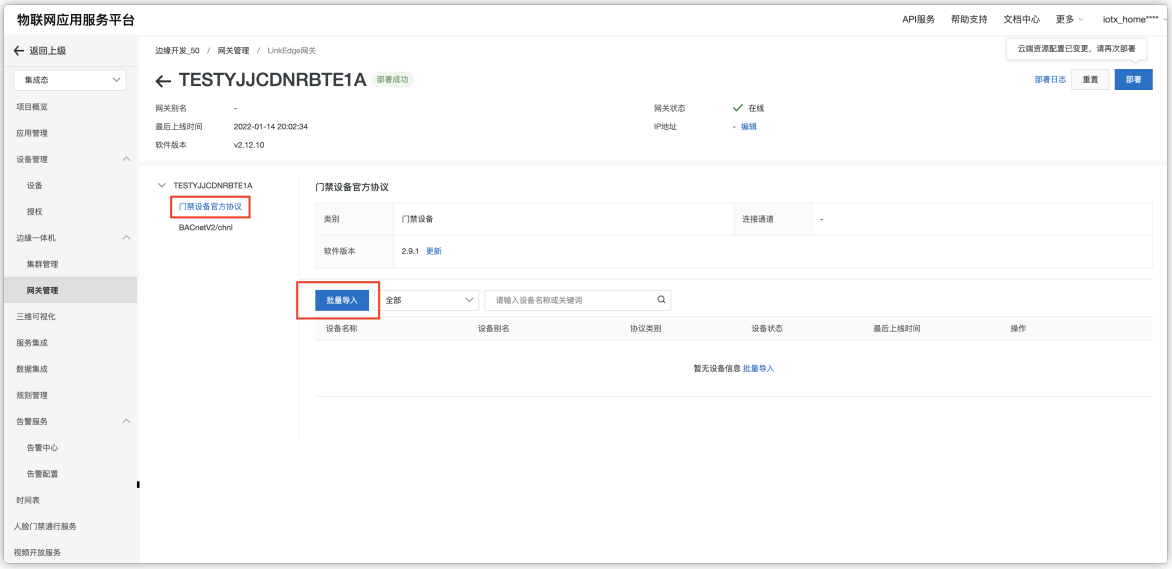
- **功能定义**：选择已有模型，对应设备名称选择智慧社区-人员通行-可视对讲。



- **门禁类型**：选择“可视对讲”
- **IP**:参照甲方提供的<设备信息表>中的该可视对讲设备IP。输入示例为10.249.32.40
- **端口号**：**太川**门禁设备对外提供的可视对讲网络访问端口号固定是**10020**

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

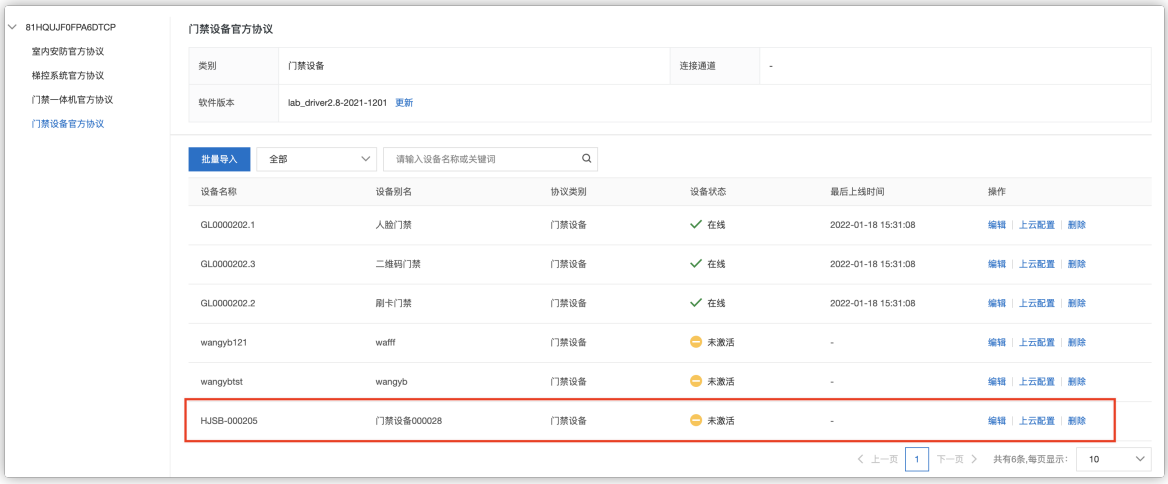
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.13. 悉尔人行设备

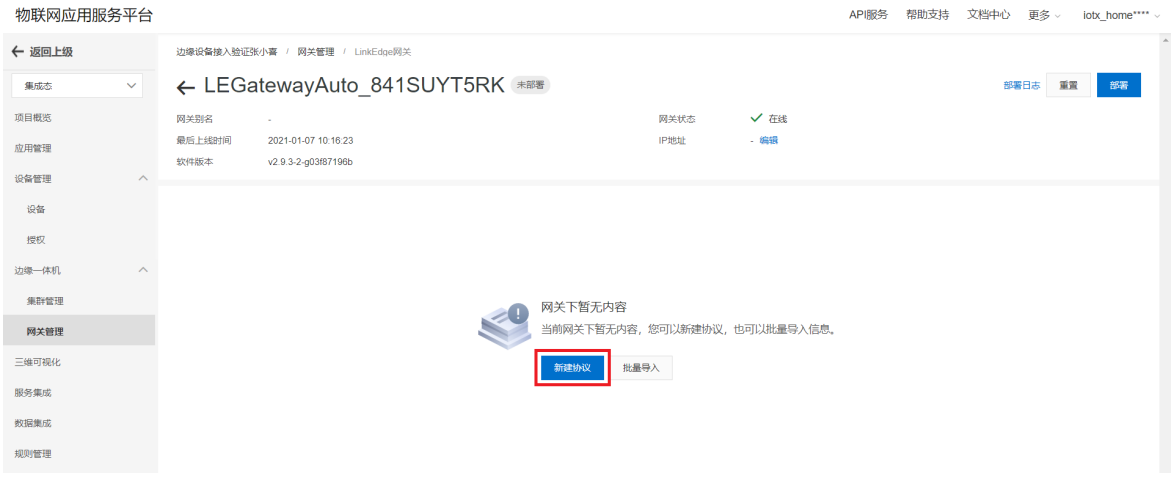
说明

悉尔支持的型号

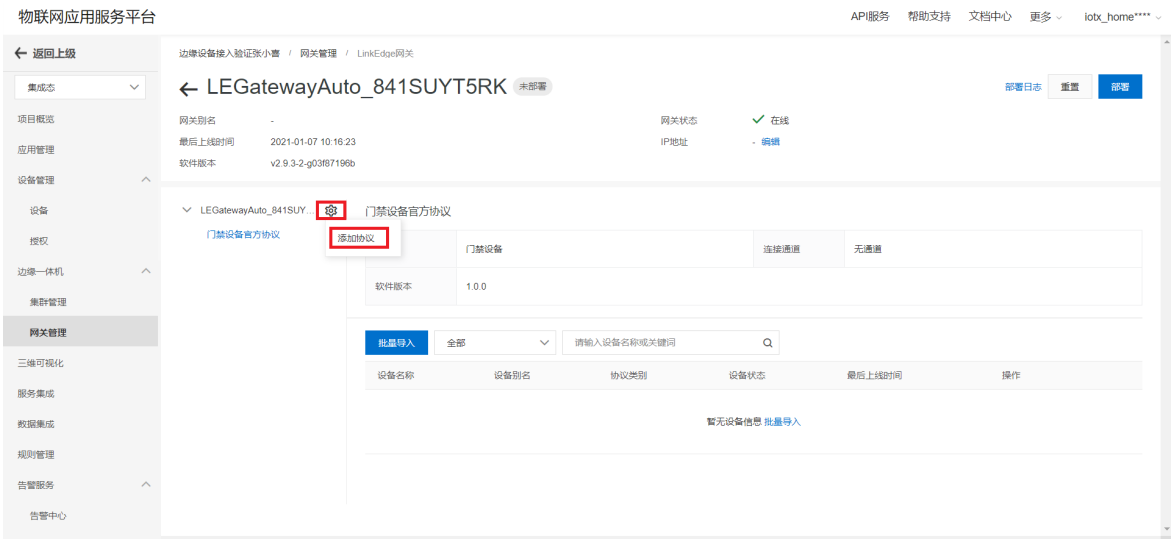
对接型号支持	A18010
接入模式	门禁设备
支持功能	人脸/二维码
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	N/A

门禁设备集成

1. 添加门禁协议
   - 第一次添加协议，可以点击**新建协议**按钮，如下图所示



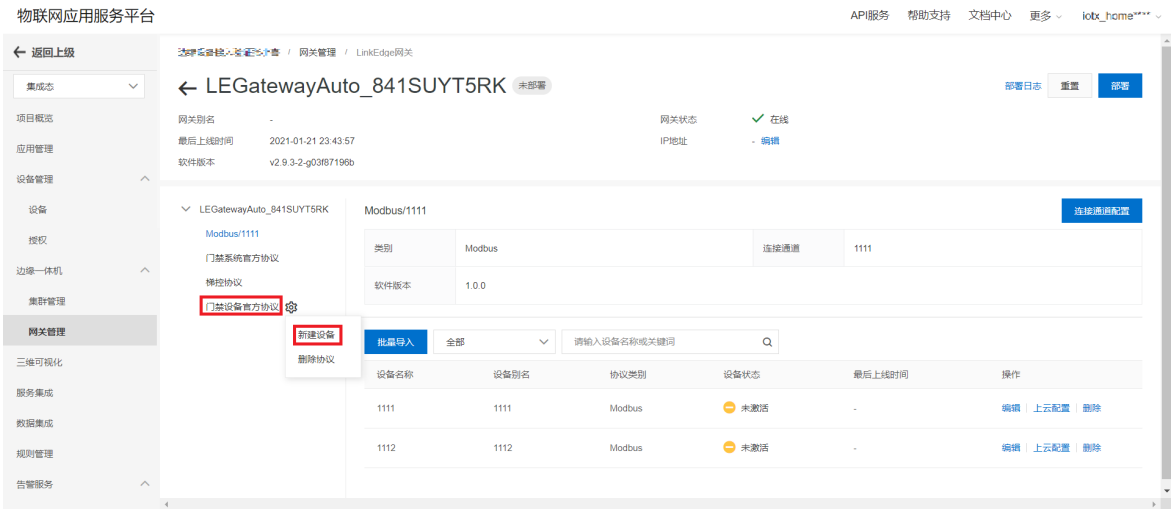
   - 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



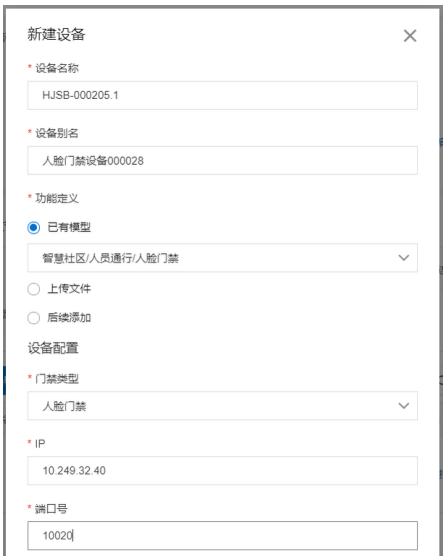
- 悉尔的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示



2. 添加门禁设备
   - 如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。

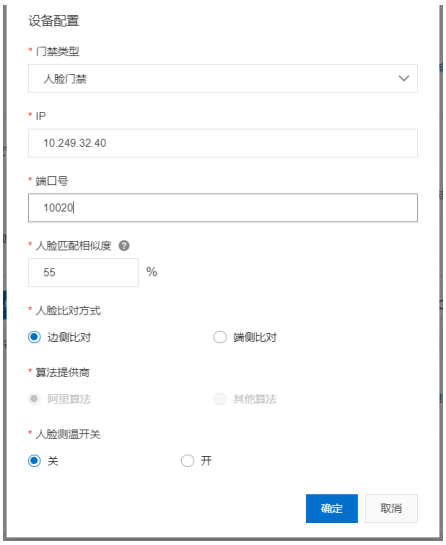


   - 如图所示，在官方协议下新增人脸门禁设备。其中：
     - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，".1"标识人脸门禁设备
     - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028
     - **功能定义**：**悉尔**门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"

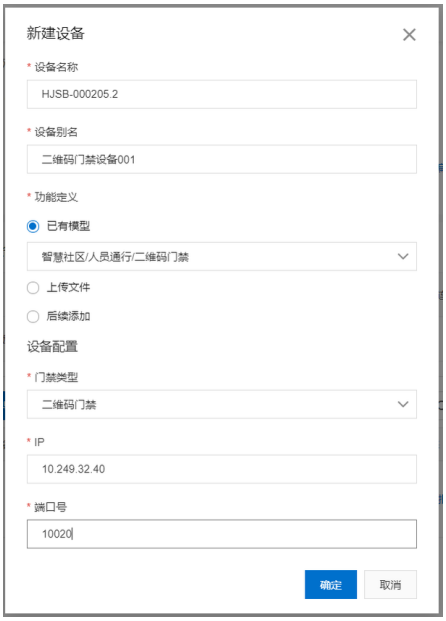


     - **门禁类型**：选择“**人脸门禁**”
     - **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
     - **端口号**：悉尔人脸是在边侧比对，这里端口号可以填写10020
     - **人脸匹配相似度**：输入**55%**。
     - **人脸比对方式**：**悉尔**人脸比对方式为**边侧比对**

- **人脸测温开关：悉尔**门禁设备不支持人脸测温，设置为关



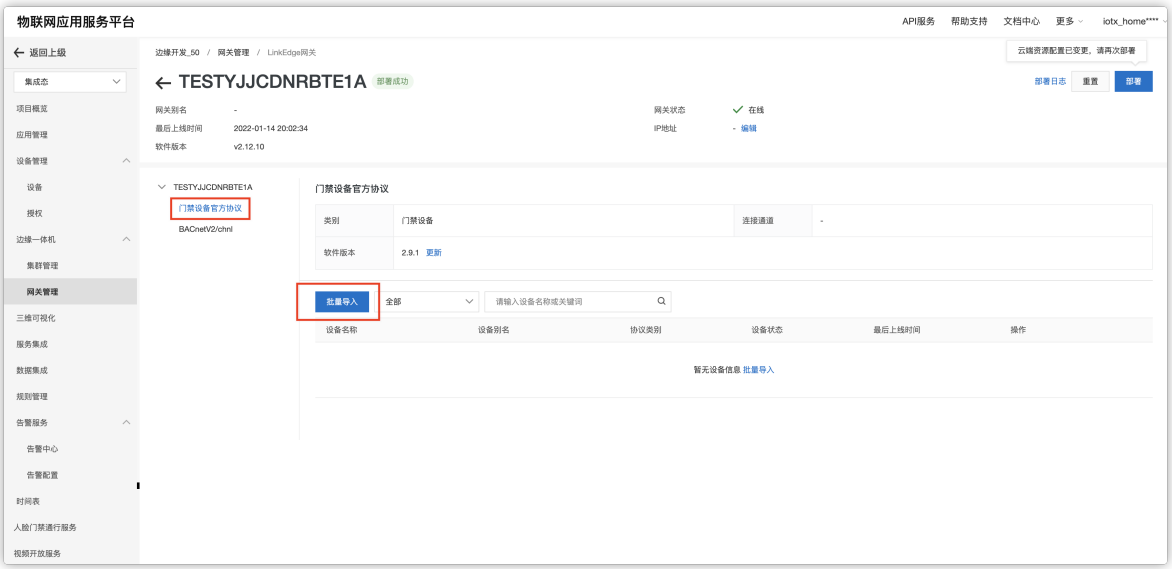
  - 如图所示，在官方协议下新增二维码门禁设备。其中：
    - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，".**2**"标识二维码门禁设备
    - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001
    - **功能定义**：**悉尔**门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



    - **门禁类型**：选择“二维码门禁”
    - **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
    - **端口号**：**悉尔**二维码是在边侧比对，这里端口号可以填写10020

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

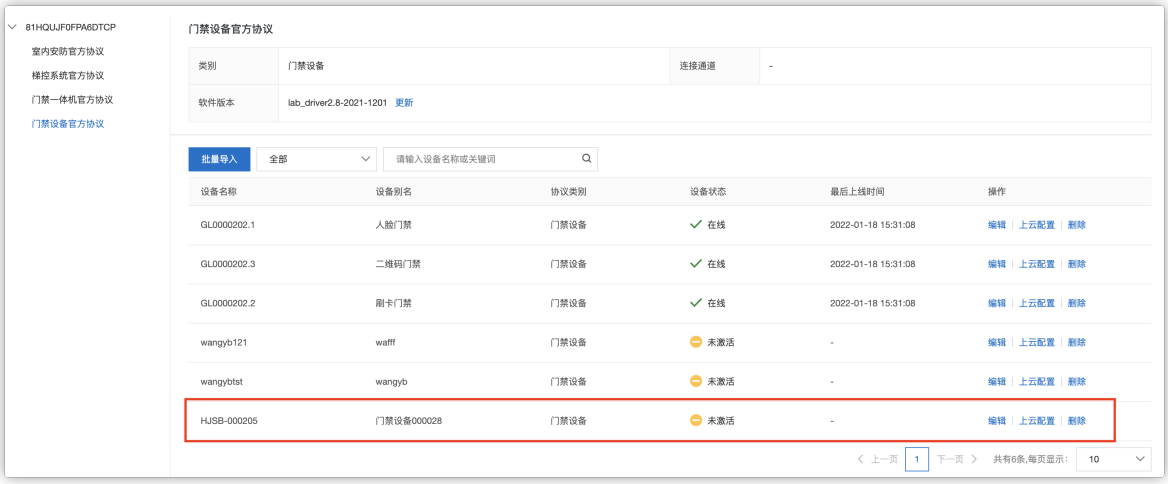
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.14. 通力人行设备

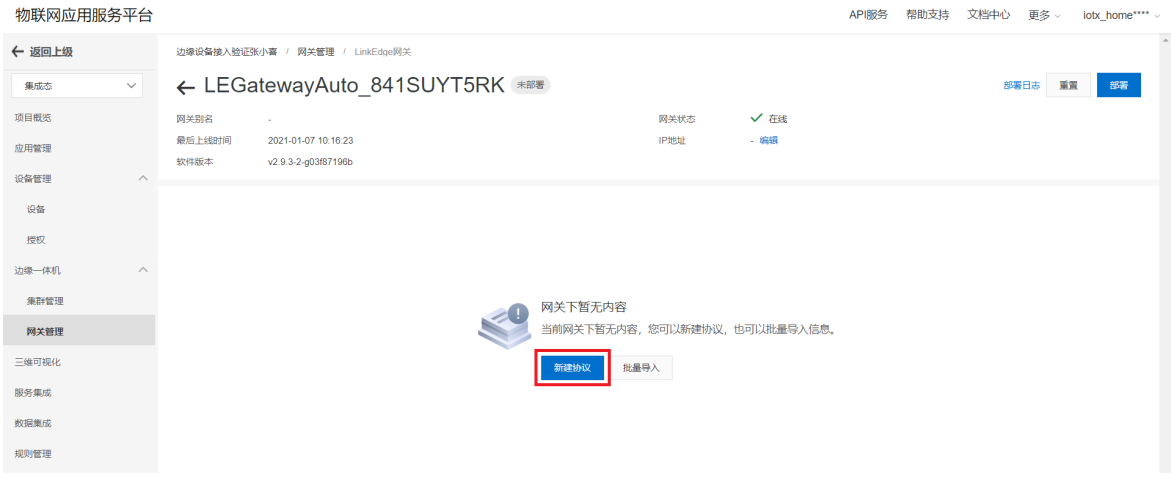
说明

通力支持的型号

对接型号支持	DID-101
接入模式	门禁系统
支持功能	人脸/二维码
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	N/A

门禁设备集成

1. 添加门禁协议
   - 第一次添加协议，可以点击**新建协议**按钮，如下图所示



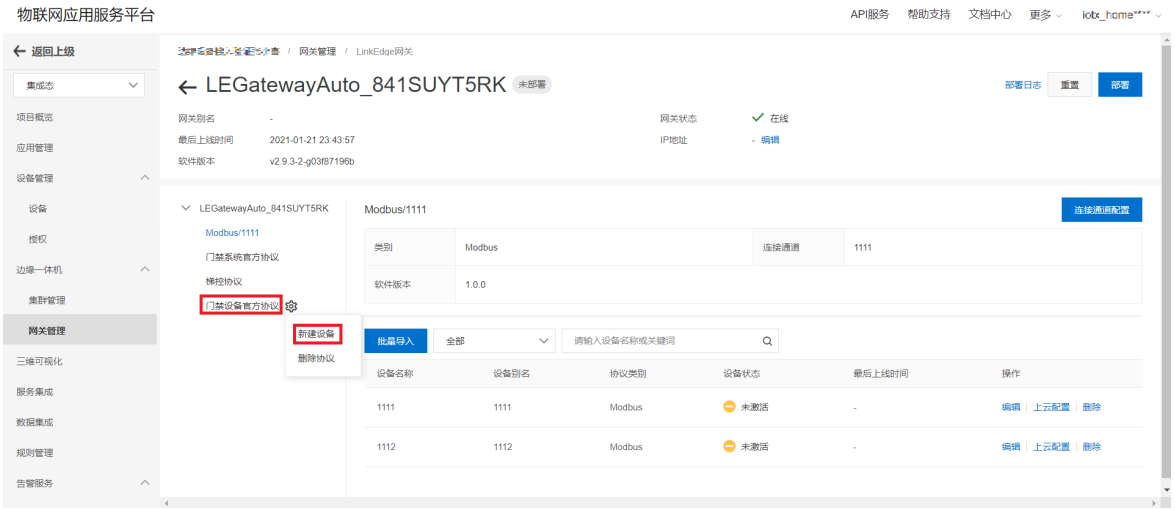
   - 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



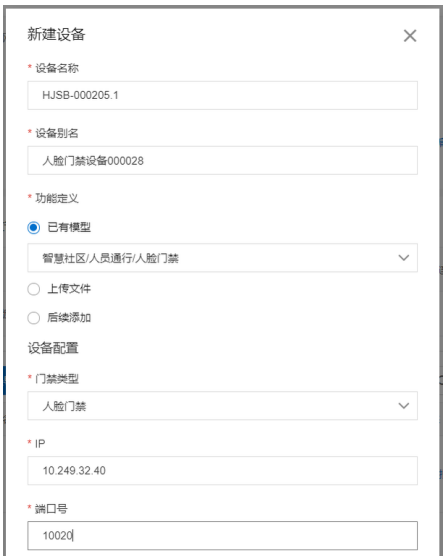
- **通力**的接入模式是系统接入，在弹出的对话框里，选择**门禁系统官方协议**，如下图所示



2. 添加门禁设备
   - 如下图所示，鼠标移动到到刚添加的**门禁系统官方协议**，点击**新建设备**，进行人行设备的添加。



   - 如图所示，在官方协议下新增人脸门禁设备。其中：
     - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，".1"标识人脸门禁设备
     - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028
     - **功能定义**：**通力**门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"

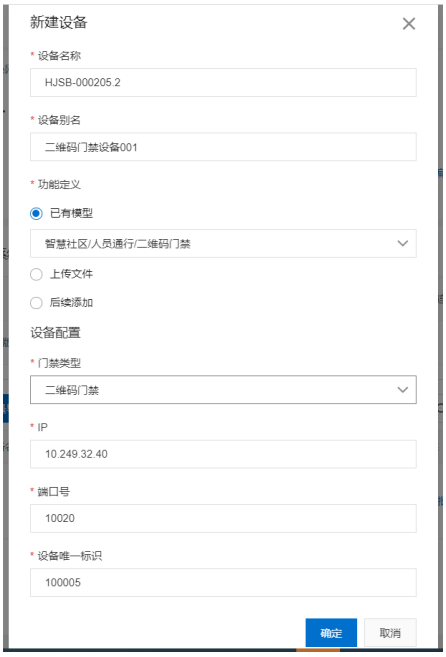


     - **门禁类型**：选择“**人脸门禁**”
     - **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
     - **端口号**：**通力**人脸是在边侧比对，这里端口号可以填写10020
     - **人脸匹配相似度**：输入**55%**。
     - **设备唯一标识**：参照甲方提供的<设备信息表>中的该门禁**设备唯一标识**。输入示例为100005
     - **人脸比对方式**：**通力**人脸比对方式为**边侧比对**

- **人脸测温开关：通力**门禁设备不支持人脸测温，设置为关



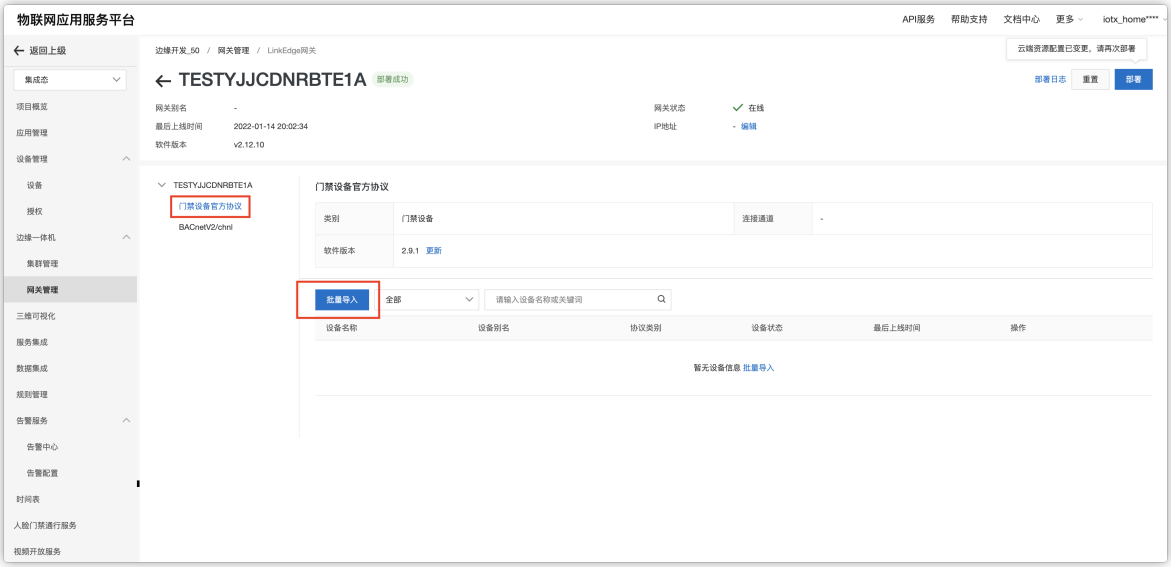
  - 如图所示，在官方协议下新增二维码门禁设备。其中：
    - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备
    - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001
    - **功能定义**：**通力**门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



    - **门禁类型**：选择“二维码门禁”
    - **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
    - **端口号**：**通力**二维码是在边侧比对，这里端口号可以填写10020
    - **设备唯一标识**：参照甲方提供的<设备信息表>中的该门禁**设备唯一标识**。输入示例为100005

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

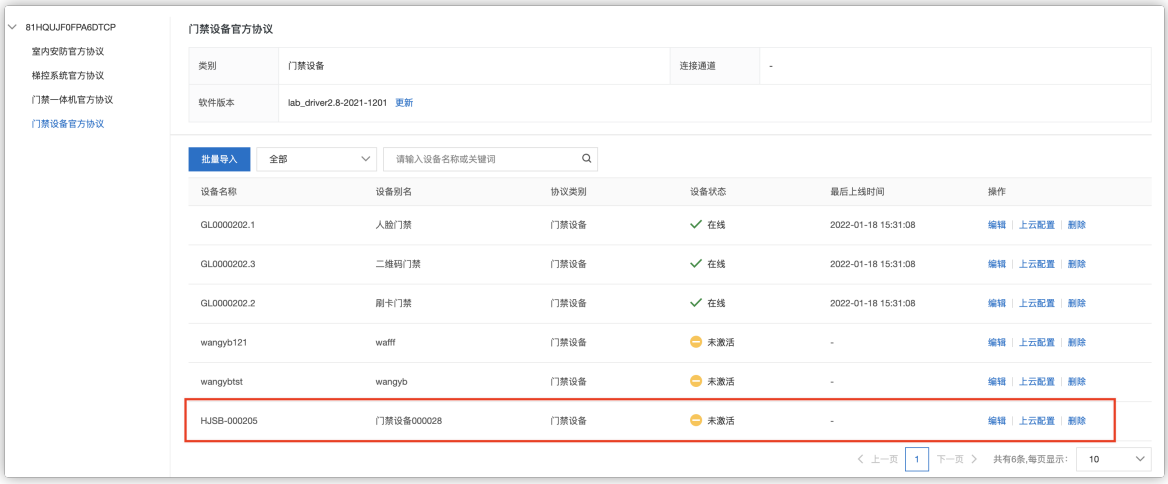
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.15. 魔点人行设备

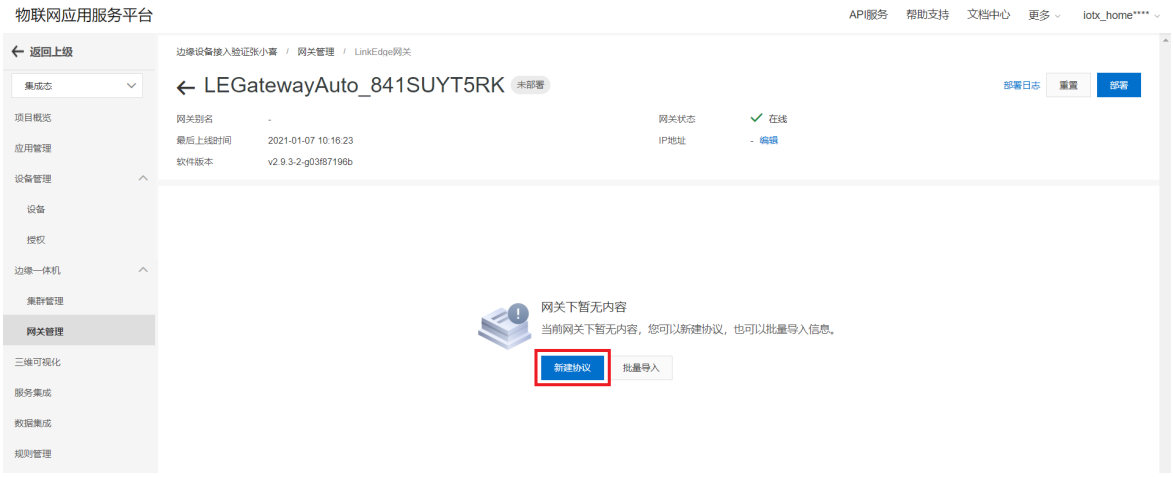
说明

魔点支持的型号

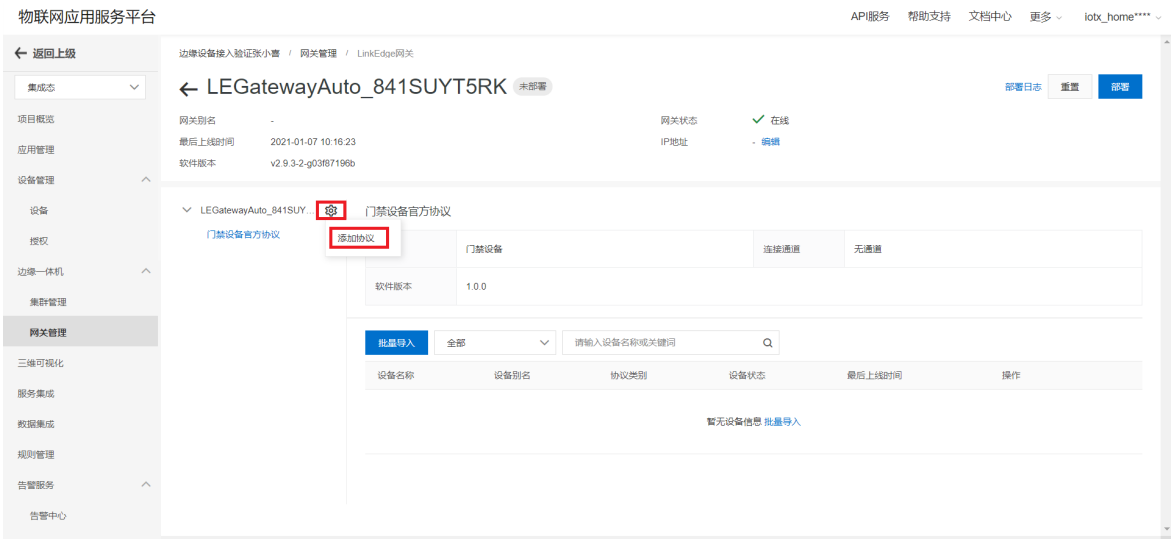
对接型号支持	M1
接入模式	门禁设备
支持功能	人脸
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	无

门禁设备集成

1. 添加门禁协议
   - 第一次添加协议，可以点击**新建协议**按钮，如下图所示



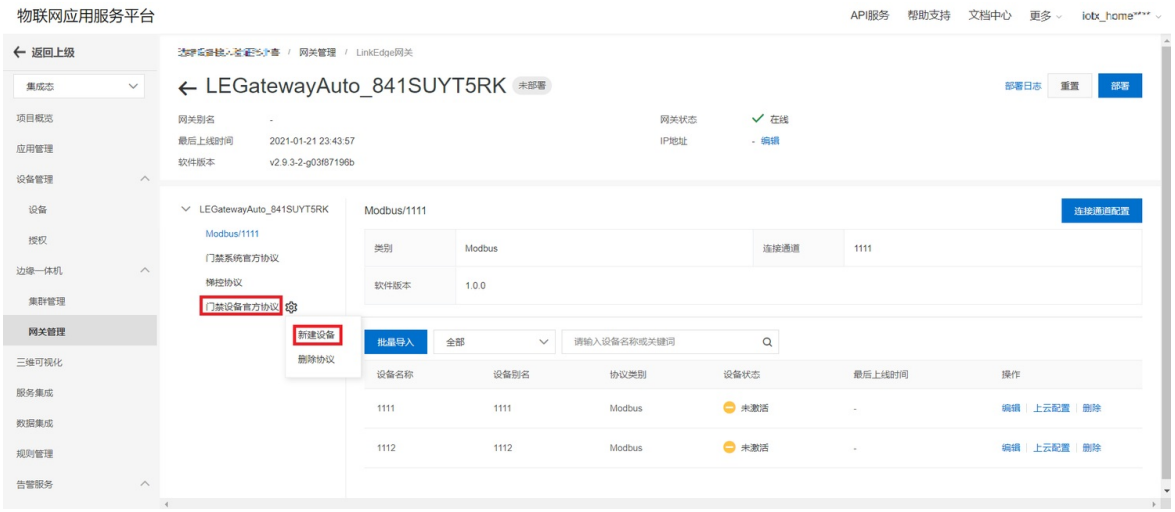
   - 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



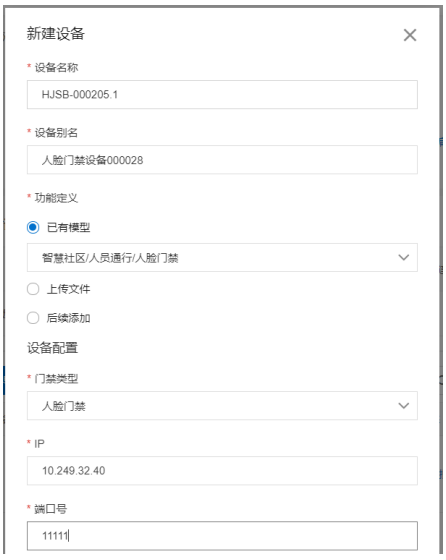
- **魔点**的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示



2. 添加门禁设备
   - 如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。

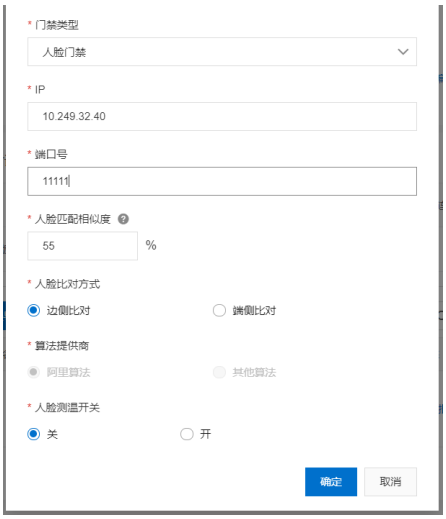


   - 如图所示，在官方协议下新增人脸门禁设备。其中：
     - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，".1"标识人脸门禁设备
     - **设备别名**：参照甲方提供的<设备信息表>中的设备名称。配置示例为：人脸门禁设备000028
     - **功能定义**：门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-人脸门禁。



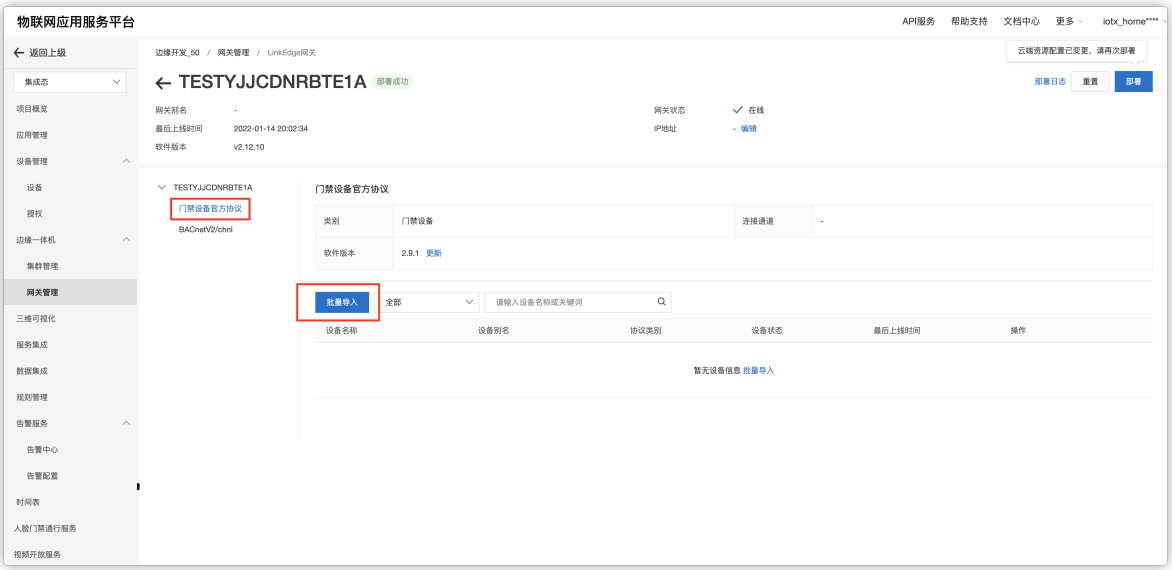
     - **门禁类型**：选择“人脸门禁”
     - **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40
     - **端口号**：**魔点**人脸在边侧识别，不需要设备端口号。这里固定填写11111
     - **人脸匹配相似度**：输入55%
     - **人脸比对方式**：**魔点**人脸比对方式为边侧比对

- **人脸测温开关：魔点**门禁设备不支持人脸测温，设置为关



3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

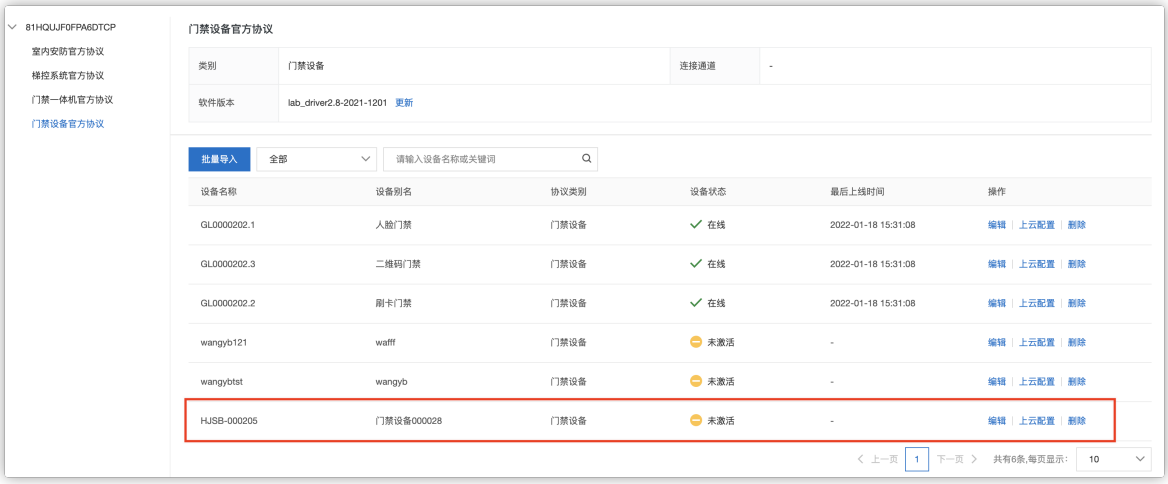
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



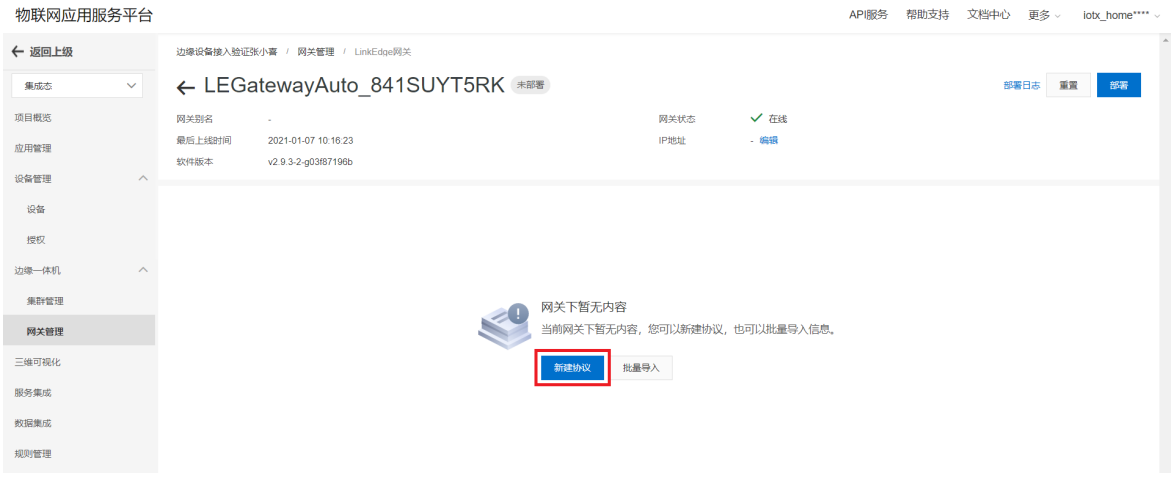
3.3.5.3.16. 安居宝人行设备

> 说明
>
> 安居宝支持的型号

对接型号支持	AJB-ZJ15CCR(AM)IPAJB-ZJ18AAC(AM)IPAJB-ZJ15CAC(AM)IP
接入模式	门禁设备
支持功能	人脸/二维码
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	N/A

门禁设备集成

1. 添加门禁协议
   - 第一次添加协议，可以点击**新建协议**按钮，如下图所示



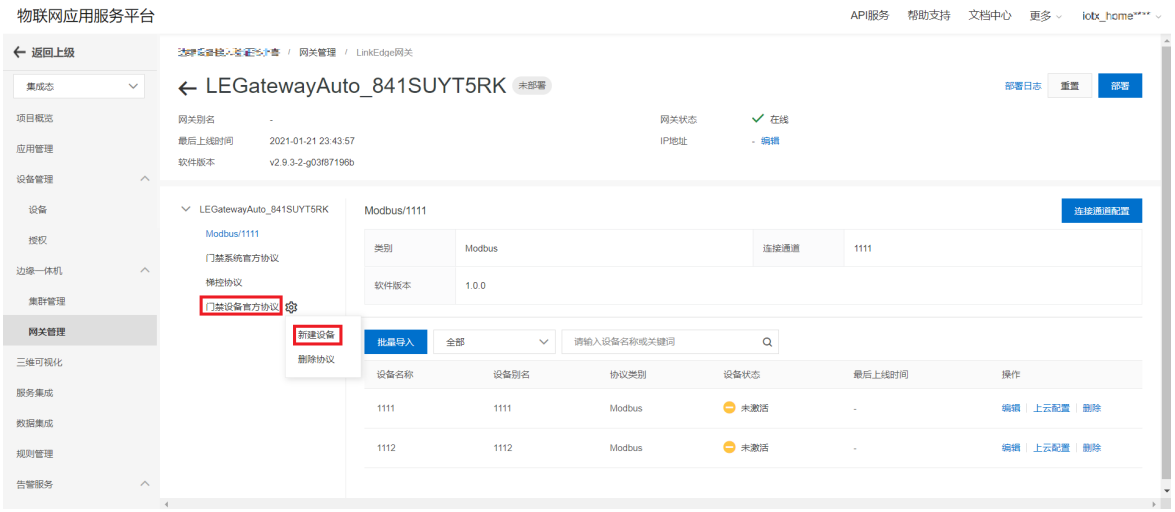
   - 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



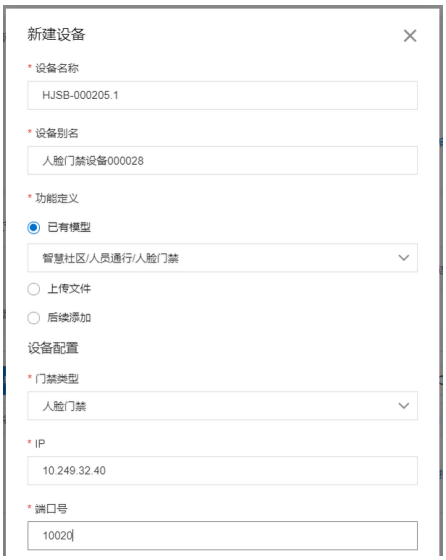
- 安居宝的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示



2. 添加门禁设备
   - 如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。

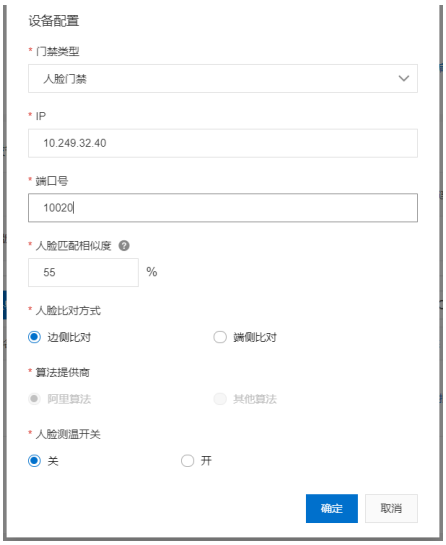


   - 如图所示，在官方协议下新增人脸门禁设备。其中：
     - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，".1"标识人脸门禁设备
     - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028
     - **功能定义**：安居宝门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"

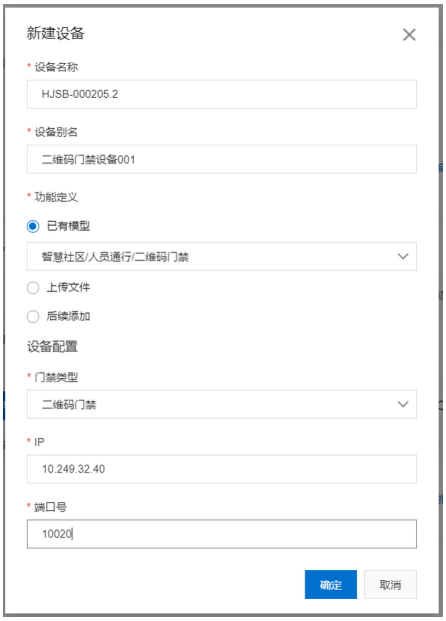


     - **门禁类型**：选择“**人脸门禁**”
     - **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
     - **端口号**：悉尔人脸是在边侧比对，这里端口号可以填写10020
     - **人脸匹配相似度**：输入**55%**。
     - **人脸比对方式**：安居宝人脸比对方式为**边侧比对**

- **人脸测温开关**：安居宝门禁设备不支持人脸测温，设置为关



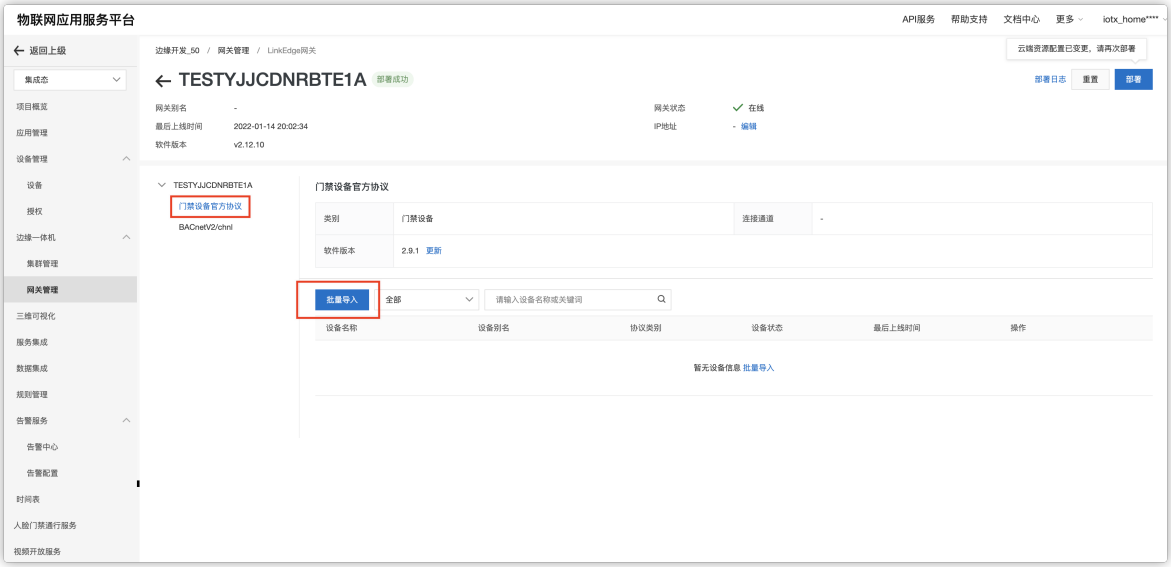
  - 如图所示，在官方协议下新增二维码门禁设备。其中：
    - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，".**2**"标识二维码门禁设备
    - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001
    - **功能定义**：安居宝门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



    - **门禁类型**：选择“二维码门禁”
    - **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
    - **端口号**：安居宝二维码是在边侧比对，这里端口号可以填写10020

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

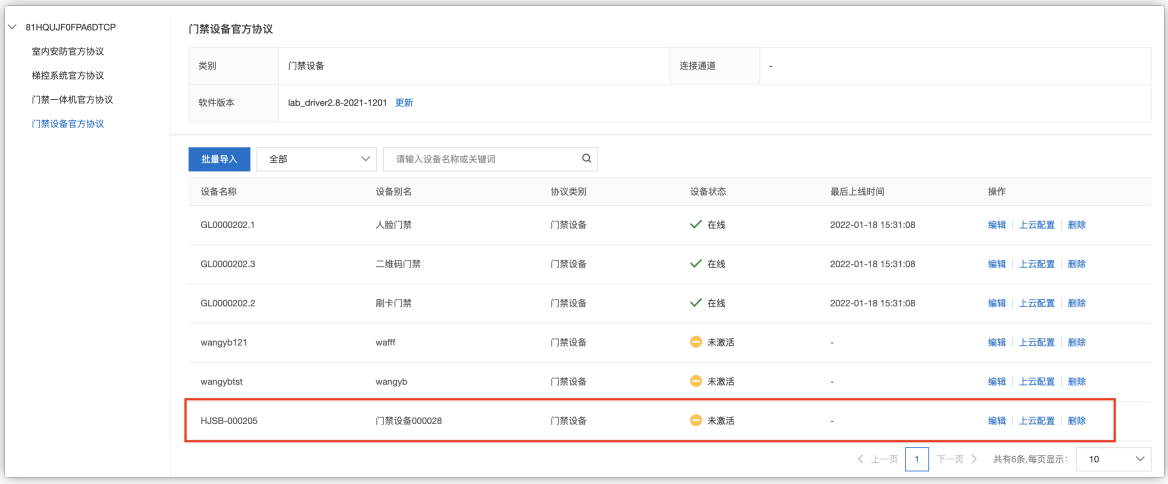
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.17. 霍尼韦尔人行设备

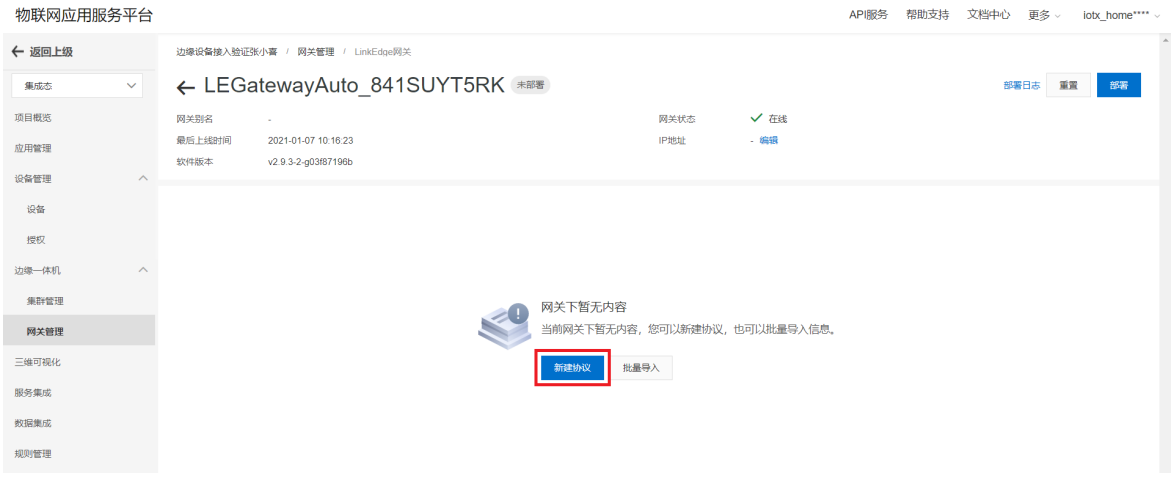
说明

霍尼韦尔支持的型号

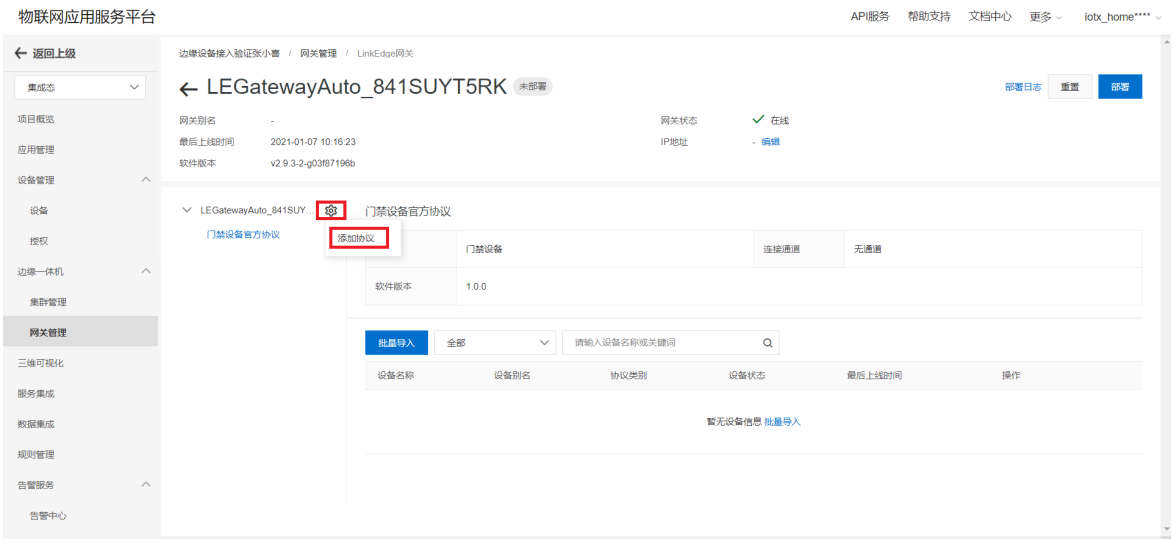
对接型号支持	HLP-6050F
接入模式	门禁设备
支持功能	人脸/刷卡
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	10000

门禁设备集成

1. 添加设备协议
   - 第一次添加协议，可以点击**新建协议**按钮，如下图所示



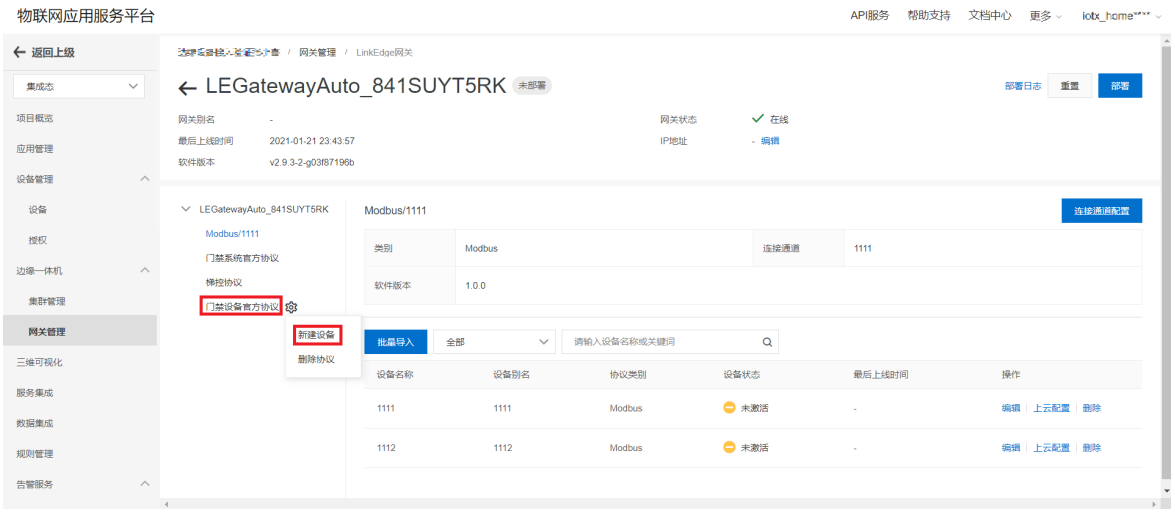
   - 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



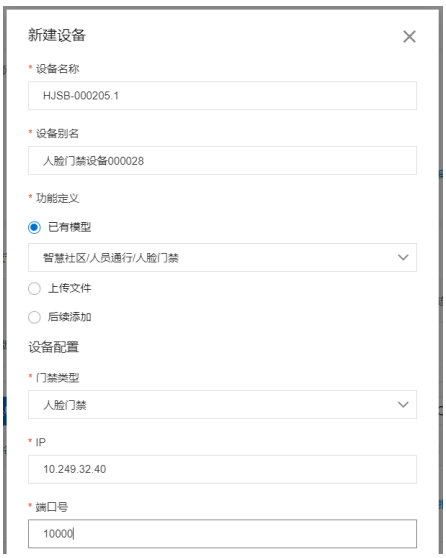
- 接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示



2. 添加门禁设备
   - 如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。

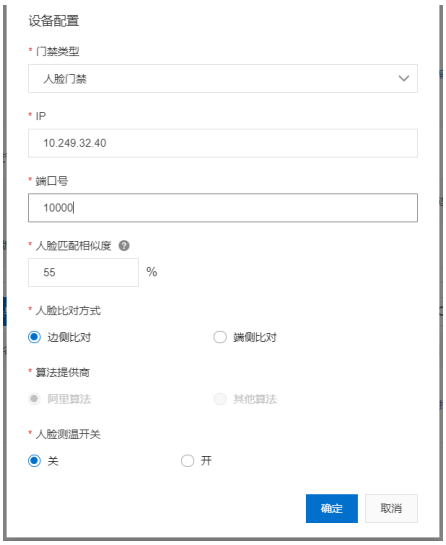


   - 如图所示，在官方协议下新增人脸门禁设备。其中：
     - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，".1"标识人脸门禁设备
     - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028
     - **功能定义**：霍尼韦尔门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"

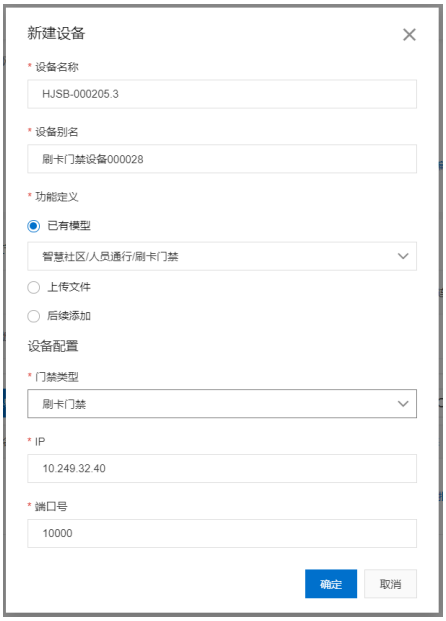


     - **门禁类型**：选择“**人脸门禁**”
     - **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
     - **端口号**：霍尼韦尔门禁设备对外提供的网络访问端口号固定是**10000**
     - **人脸匹配相似度**：输入**55%**。
     - **人脸比对方式**：霍尼韦尔人脸比对方式为**边侧比对**

- **人脸测温开关**：霍尼韦尔门禁设备不支持人脸测温，设置为关



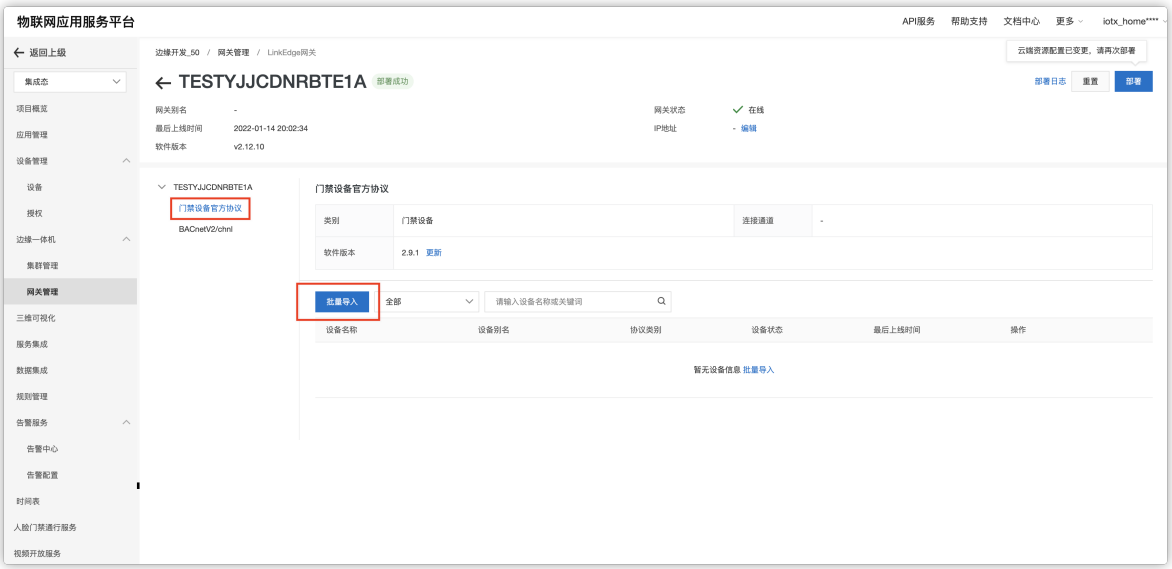
  - 如图所示，在官方协议下新增刷卡门禁设备。其中：
    - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中"**HJSB-000205**"是该设备的IoT编码，".3"标识刷卡门禁设备
    - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028
    - **功能定义**：霍尼韦尔门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-刷卡门禁。



    - **门禁类型**：选择“刷卡门禁”
    - **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40
    - **端口号**：霍尼韦尔门禁设备对外提供的网络访问端口号固定是10000

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

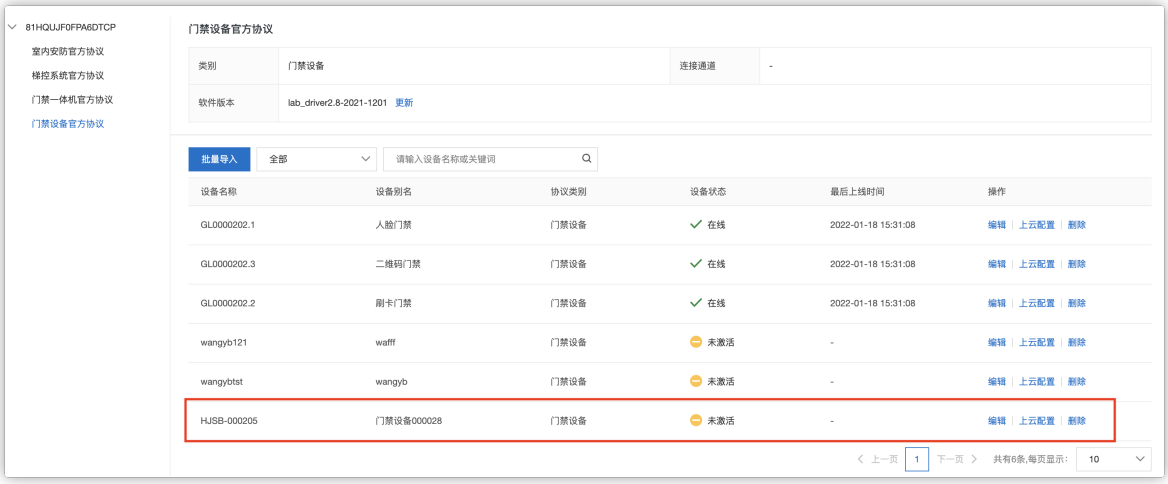
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.18. 富士人行设备

说明

设备对接支持的型号

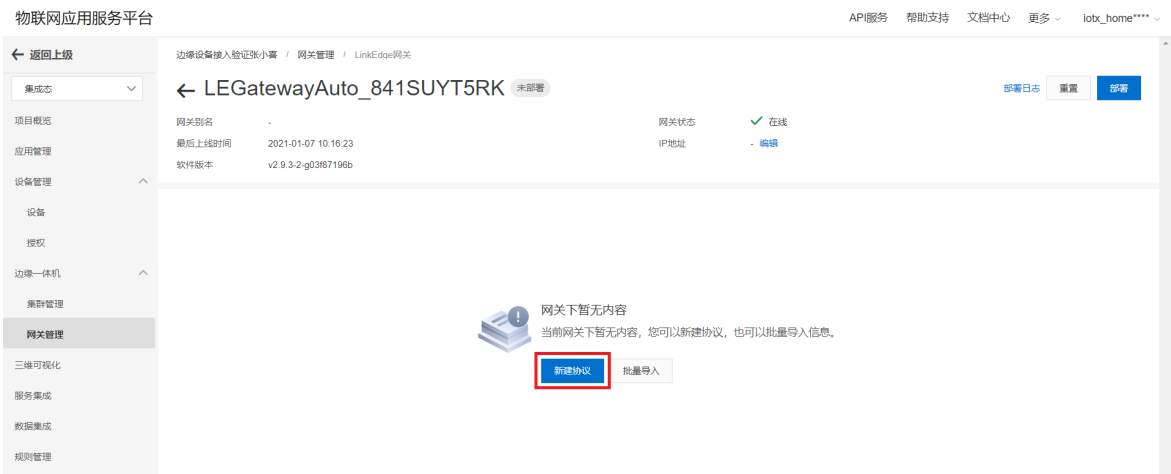
对接型号支持	FJC-ZRS632
接入模式	门禁设备
支持功能	人脸/二维码
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	N/A

系统对接支持的型号

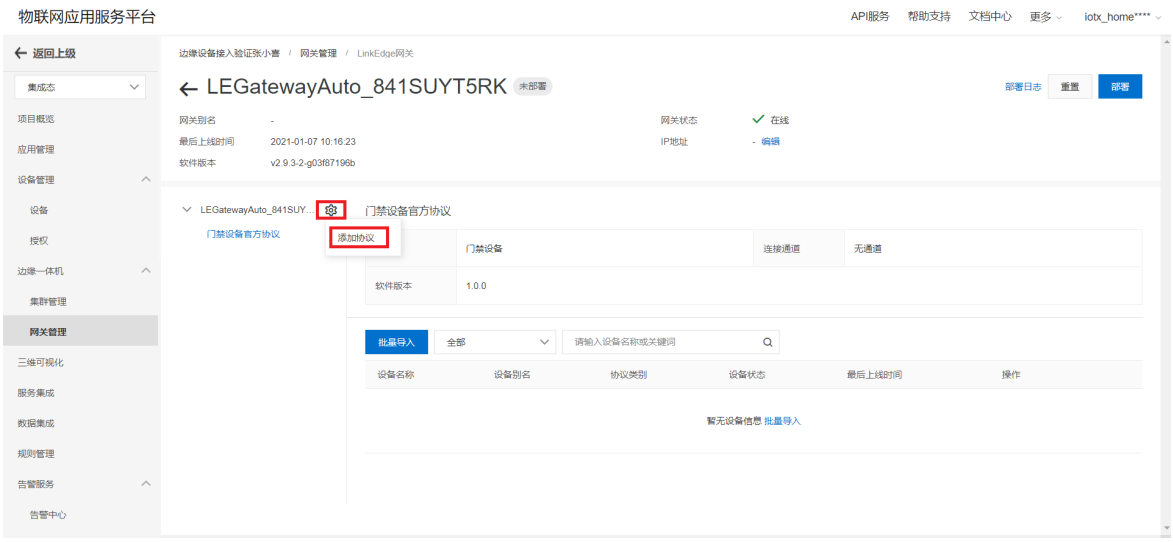
对接型号支持	N/A
接入模式	门禁系统
支持功能	刷卡/远程开门
硬件型号	分体门禁读卡器（IC） FJC-MG613TCP/IP嵌入式门禁读卡器（IC） FJC-MAT612双门控制器 FJC-MFT211
系统版本	V8.1.2.0
门禁端口号	8008

通过直连设备对接

1. 添加门禁协议
   - 第一次添加协议，可以点击**新建协议**按钮，如下图所示



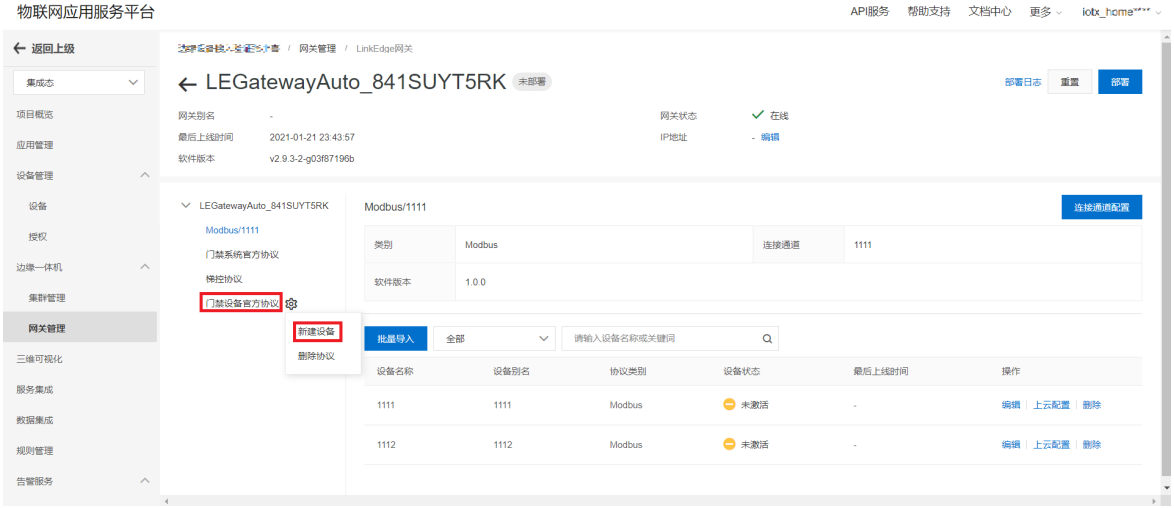
- 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



- 富士的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示

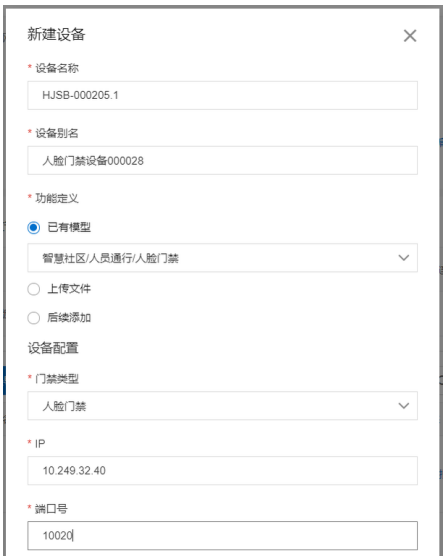


2. 添加门禁设备
   - 如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。

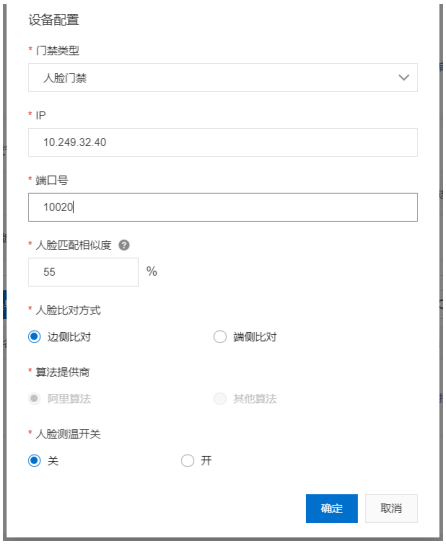


   - 如图所示，在官方协议下新增人脸门禁设备。其中：
     - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，"**.1**"标识人脸门禁设备
     - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028

- **功能定义**：富士门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"



- **门禁类型**：选择“**人脸门禁**”
- **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
- **端口号**：富士人脸是在边侧比对，这里端口号可以填写10020
- **人脸匹配相似度**：输入**55%**。
- **人脸比对方式**：富士人脸比对方式为**边侧比对**
- **人脸测温开关**：富士门禁设备不支持人脸测温，设置为关



- 如图所示，在官方协议下新增二维码门禁设备。其中：
  - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备
  - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001

- **功能定义**：富士门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



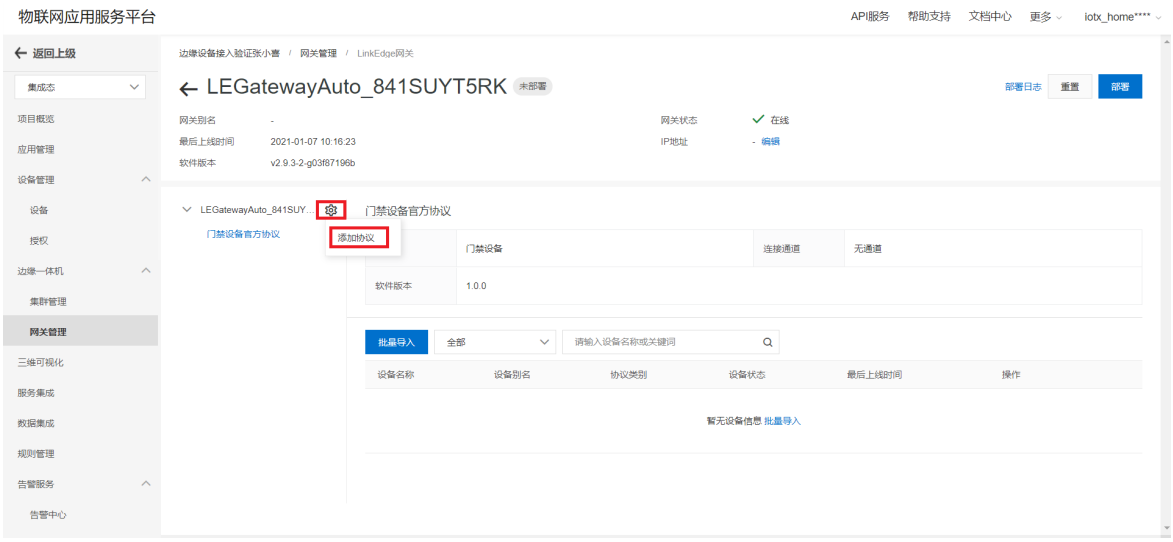
- **门禁类型**：选择“二维码门禁”
- **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
- **端口号**：富士二维码是在边侧比对，这里端口号可以填写10020

通过门禁系统对接

1. 添加门禁协议
   - 网关里首次添加协议，选择新建协议



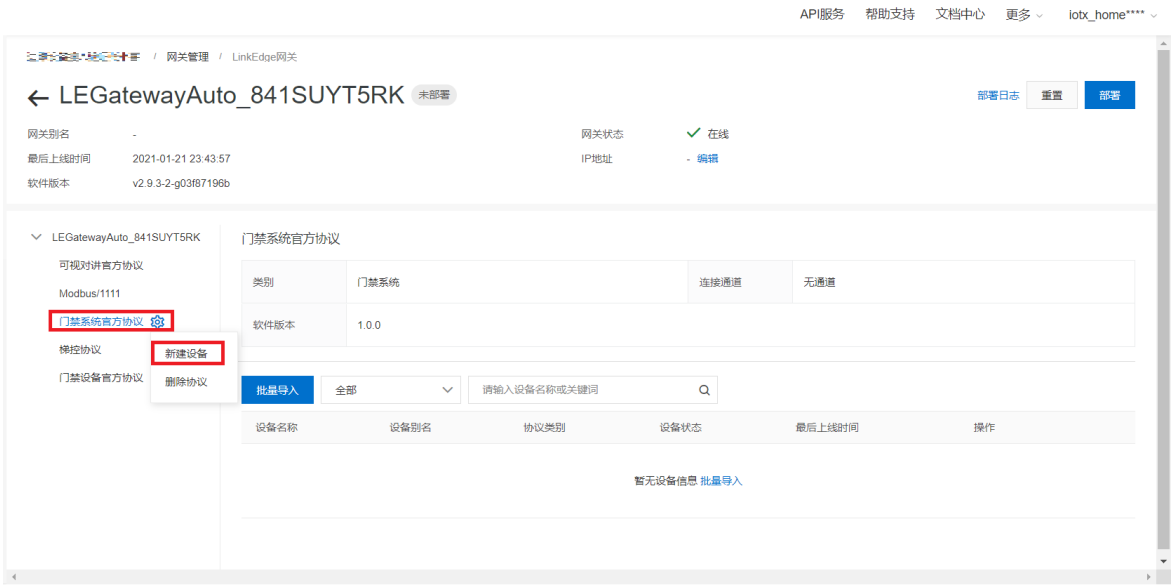
- 网关里已添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



- 富士系统对接的接入模式是系统接入，在弹出的对话框里，选择**门禁系统官方协议**，如下图所示

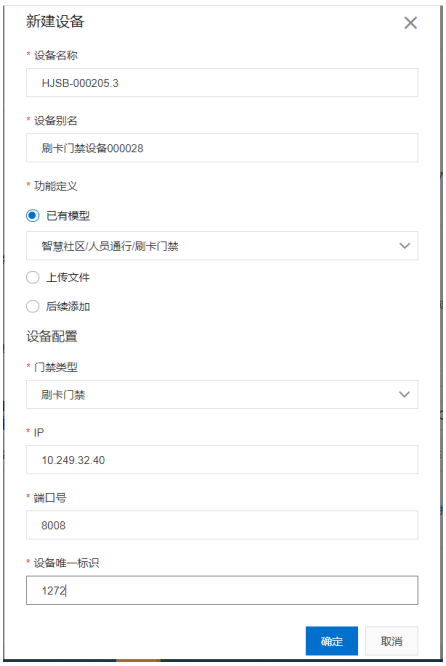


2. 添加门禁设备
   - 如下图所示，鼠标移动到到刚添加的**门禁系统官方协议**，点击**新建设备**，进行人行设备的添加。



   - 如图所示，在官方协议下新增刷卡门禁设备。其中：
     - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中"**HJSB-000205**"是该设备的IoT编码，"**.3**"标识刷卡门禁设备
     - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028

- **功能定义**：选择已有模型，对应设备名称选择智慧社区-人员通行-刷卡门禁。

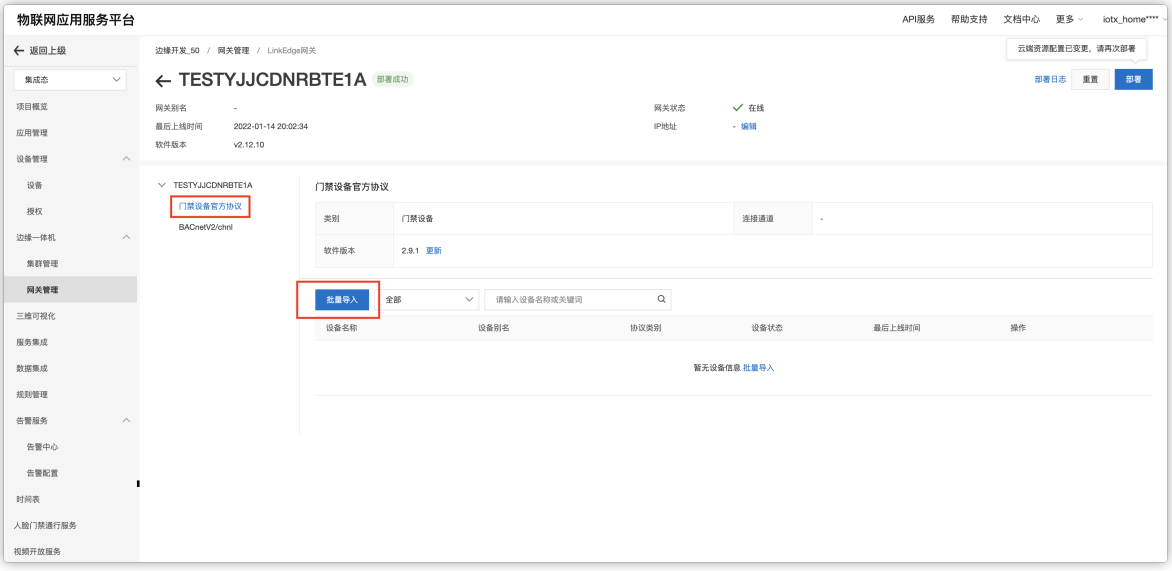


- **门禁类型**：选择“刷卡门禁”
- **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40
- **端口号**：富士门禁系统对外提供的端口号为8008
- **设备唯一标识**：参照甲方提供的<设备信息表>中的该门禁设备的**设备唯一编码**。输入示例为1272

> **警告**
> 特别注意：每个设备唯一编码必须是唯一的，如果有重复的，部署后将导致所有富士门禁设备离线

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

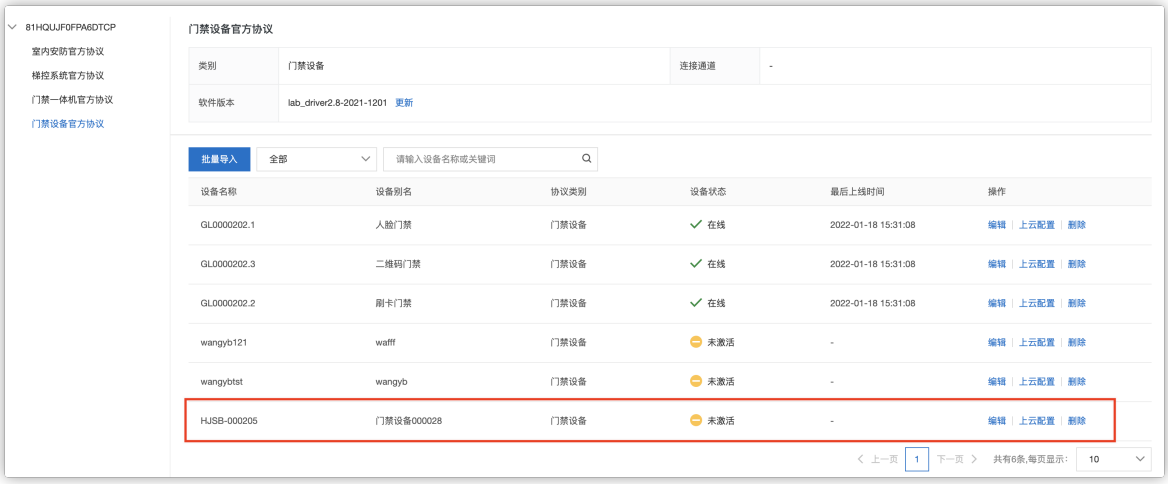
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.19. 慧锐通人行设备

1 慧锐通支持对接的型号说明

对接型号及功能	型号：B2M-801S5；软件版本：9.00.07；硬件版本：9.00.00；支持功能：人脸/二维码/可视对讲
对接型号及功能	型号：BKM-801S7；软件版本：4.14；硬件版本：JA308；支持功能：人脸/二维码
对接型号及功能	型号：B2M-801S5；软件版本：9.00.03；硬件版本：A3；支持功能：人脸/二维码/可视对讲
接入模式	门禁设备
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	10000

接入模式包含下面两种：

门禁设备：代表设备与阿里边缘服务器进行直连

门禁系统：代表设备接入集成方的服务器，集成方服务器再与阿里边缘服务器连接

人脸识别方式包含下面两种：

端侧识别：人脸特征值下发到设备端，人脸识别直接在设备端进行

边侧识别：人脸特征值保存在边缘服务器，人脸照片需要上传到边缘服务器，人脸识别在边缘服务器进行

2 人行设备接入操作

2.1 登录应用服务平台

2.1.1 选择项目

登录阿里云应用服务平台，找到需要配置的项目，该项目已由项目方创建，如果未创建，请联系项目方。



2.1.2 选择对应网关的设备集成

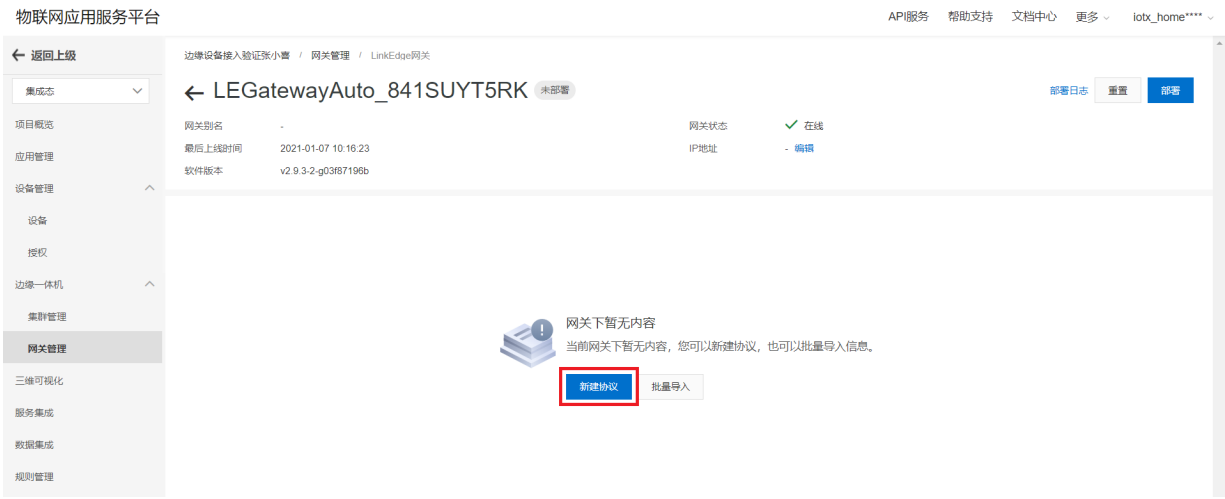
进入项目内，找到**边缘一体机-网关管理**，如下图所示。点击网关对应的**设备集成**



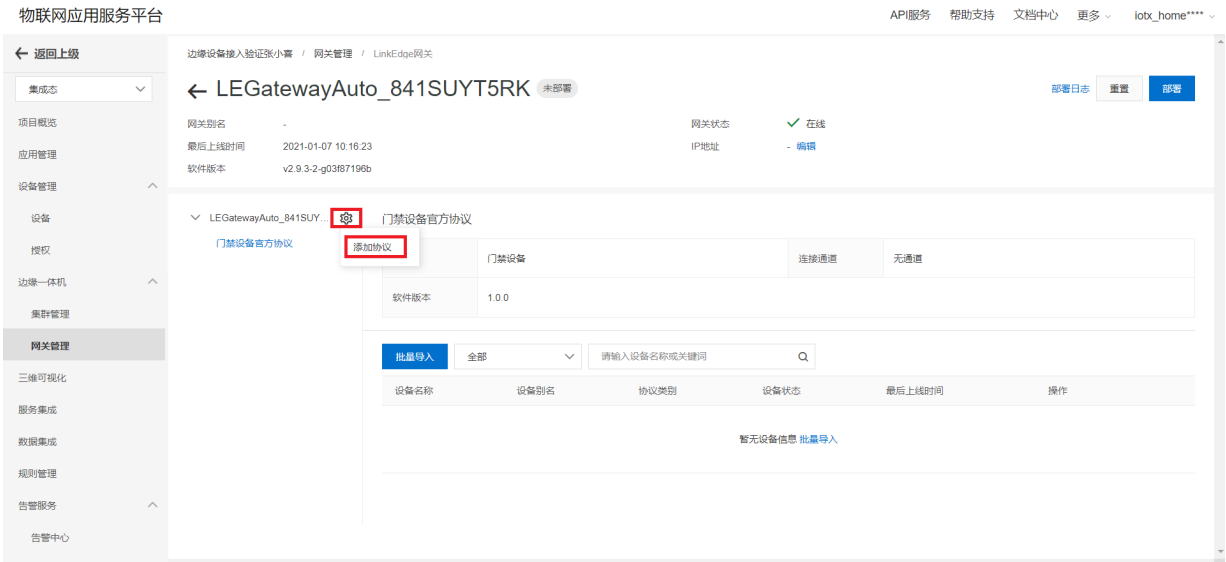
2.2 门禁设备集成

2.2.1 添加门禁协议

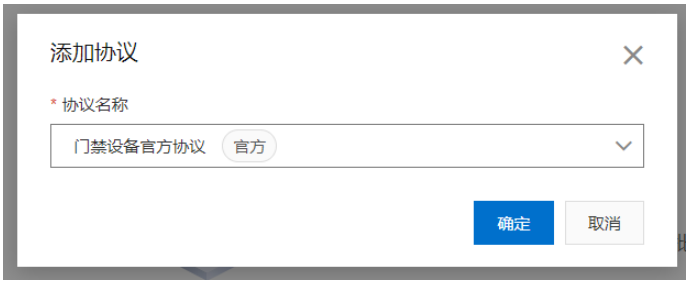
第一次添加协议，可以点击**新建协议**按钮，如下图所示



如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**

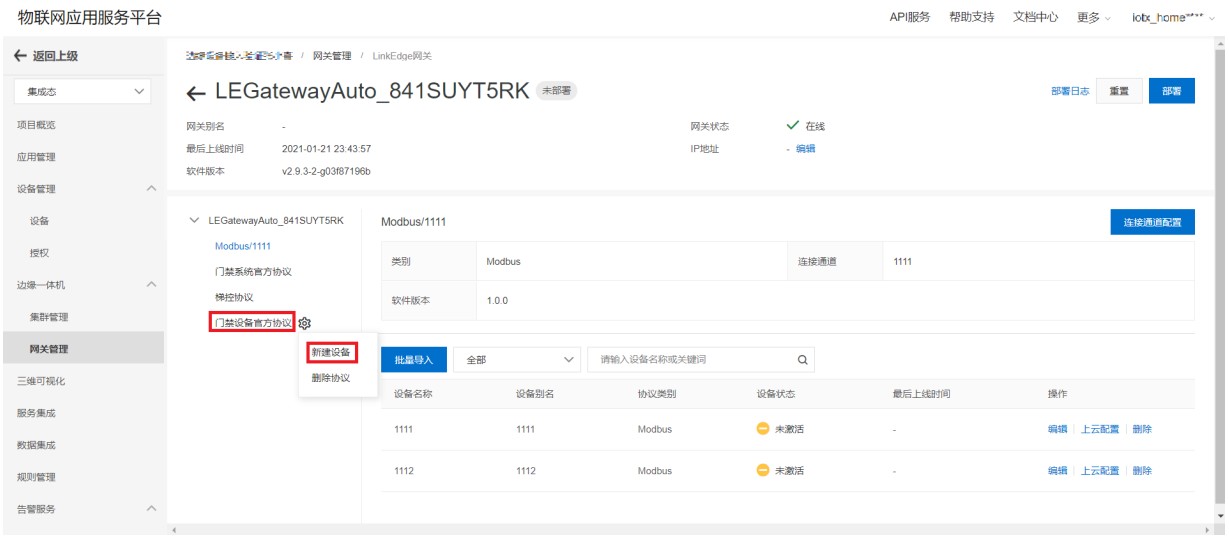


慧锐通的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示



2.2.2 添加门禁设备

如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。



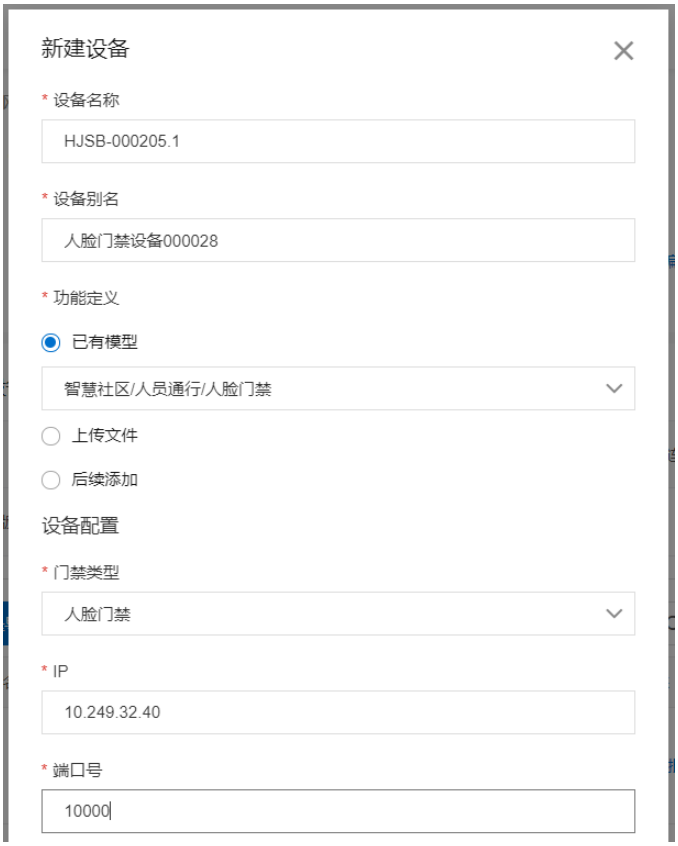
2.2.2.1 人脸门禁设备及配置

如图所示，在官方协议下新增人脸门禁设备。其中：

设备名称：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，"**.1**"标识人脸门禁设备

设备别名：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028

功能定义：门禁设备为已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"



门禁类型：选择“**人脸门禁**”

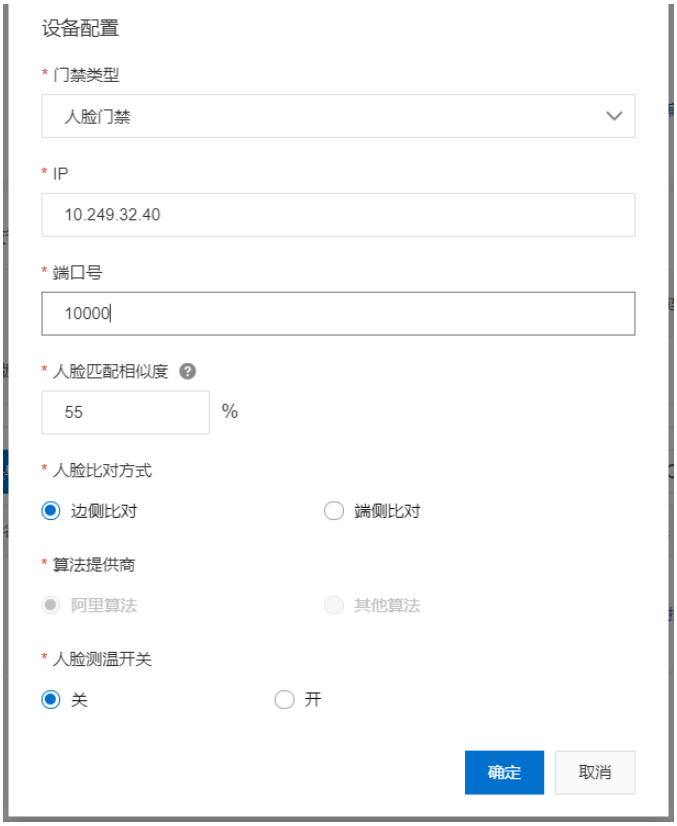
IP:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

端口号：慧锐通门禁设备对外提供的网络访问端口号固定是**10000**

人脸匹配相似度：输入**55%**。

人脸比对方式：慧锐通人脸比对方式为**边侧比对**

人脸测温开关：慧锐通门禁设备不支持人脸测温，设置为关



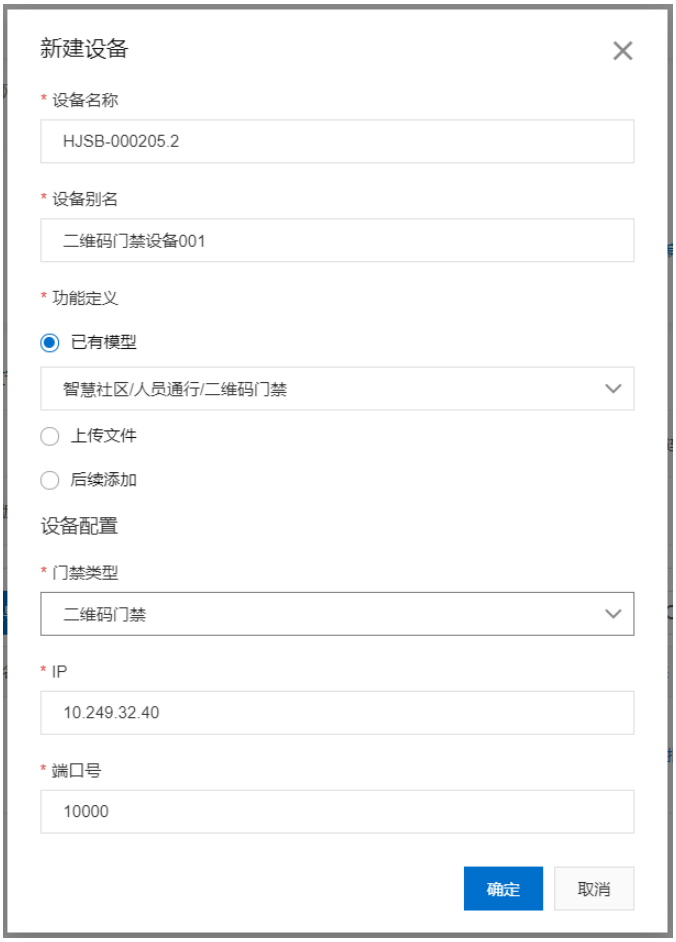
2.2.2.2 二维码门禁设备及配置

如图所示，在官方协议下新增二维码门禁设备。其中：

设备名称：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，"**.2**"标识二维码门禁设备

设备别名：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001

功能定义：门禁设备为已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



门禁类型：选择“二维码门禁”

IP:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40

端口号：慧锐通门禁设备对外提供的网络访问端口号固定是10000

2.2.2.3 可视对讲设备及配置

如图所示，在官方协议下新增可视对讲设备。其中：

设备名称：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。可视对讲设备配置示例为：HJSB-000205.5。其中"**HJSB-000205**"是该设备的IoT编码，"**.5**"标识可视对讲设备

设备别名：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：可视对讲设备000028

功能定义：选择已有模型，对应设备名称选择智慧社区-人员通行-可视对讲。



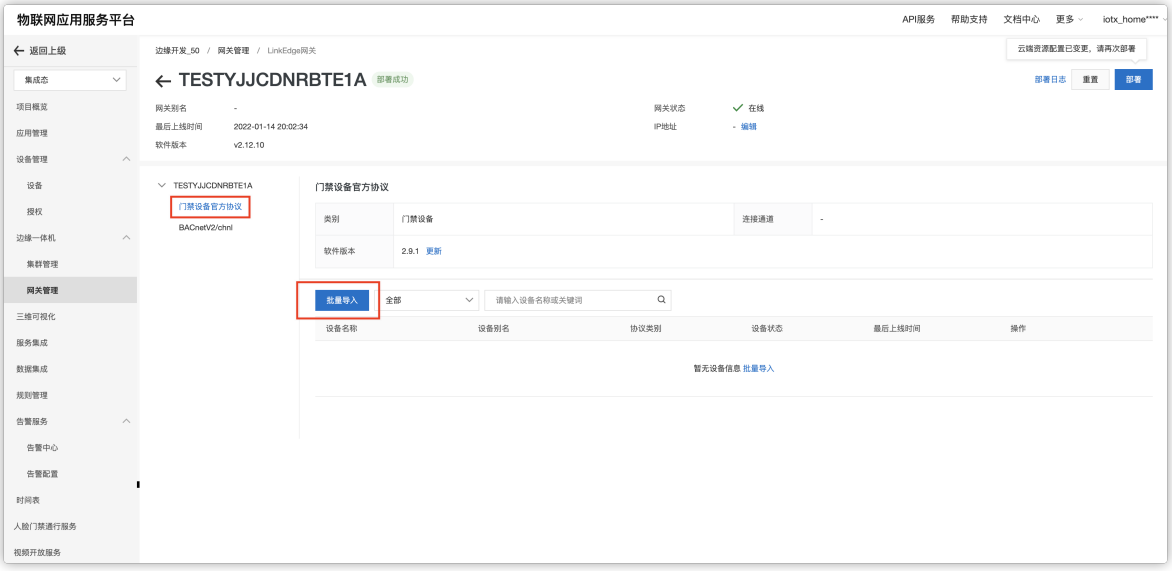
门禁类型：选择“可视对讲”

IP:参照甲方提供的<设备信息表>中的该可视对讲设备IP。输入示例为10.249.32.40

端口号：慧锐通门禁设备对外提供的网络访问端口号固定是10000

2.2.3批量导入门禁设备

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

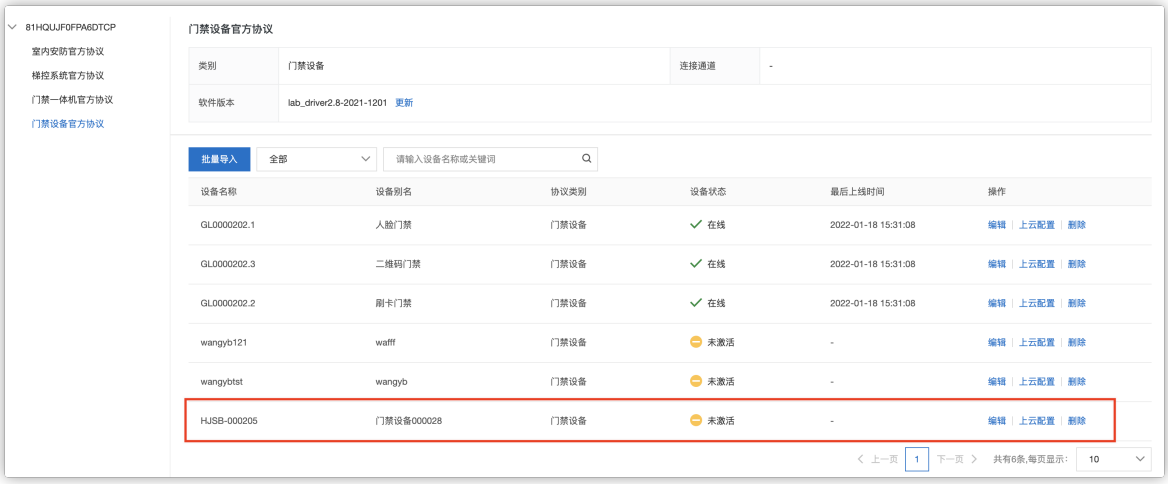
其中**设备名称、设备别名、设备品类、门禁类型、IP、端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



2.4 部署设备配置

在完成设备的配置后，点击部署，进行设备配置的协议下发到边缘服务器。如下图所示。

特别提醒：驱动部署时，网关侧对应驱动将重启，该驱动下的所有设备将出现离线后在线。这期间设备不可用，请部署人员提前与甲方沟通部署时间



点击部署后可以查看设备配置下发的状态和部署进度。



说明：如果某个设备有新更新的配置信息，重新点击部署后，只会下发有改动的设备配置协议，其余未修改的配置信息不会再次下发到边缘服务器，部署的时间根据修改的设备数量有关。

2.5 设备联通性验证

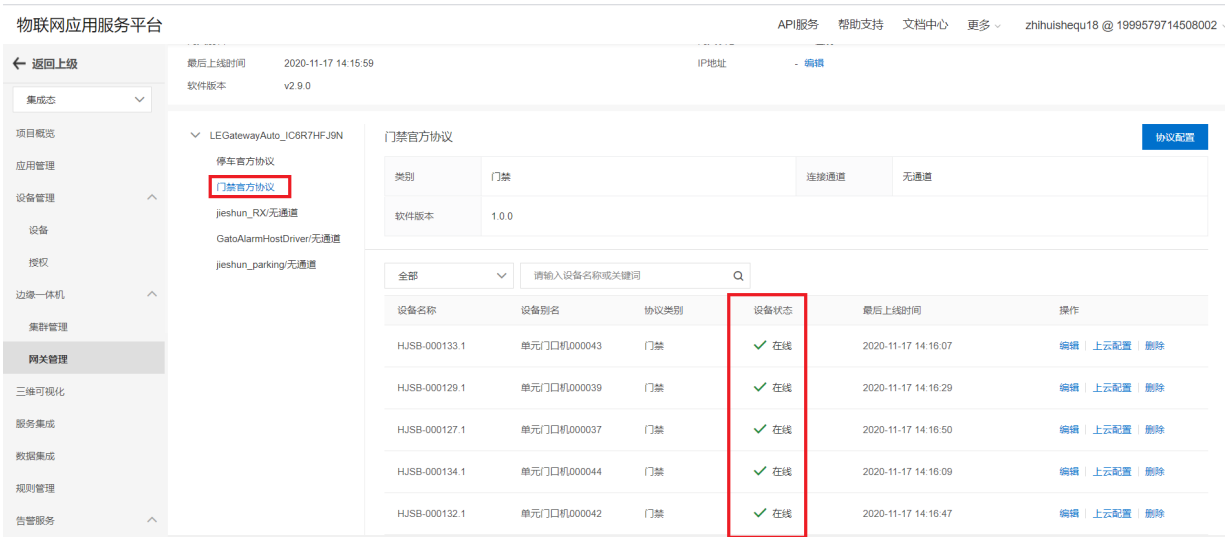
在进行设备联通性验证前，需要首先查验厂商的设备状态

1、厂商设备的软件版本已经升级到完成与阿里物业一体机对接过的版本。

2、厂商的设备配置正确，例如配置了物业一体机的IP地址和端口号。通过设备对接方式接入物业一体机，端口号是10000；通过服务器对接方式接入物业一体机，端口号是10020

3、厂商的设备IP地址设置正确，物业一体机可以正常ping通厂商设备

设备添加并部署完成后，可以点击**边缘一体机-网关管理-门禁官方协议**，逐个查看上面添加的所有设备状态，验证设备的联通性，如下图所示。如果状态是**在线**，则说明设备成功与边缘一体机连接成功；如果是未激活，说明设备与边缘一体机没有连接成功，这时可以排查的点：(1) 查看2.1.2里设备的配置是否填写正确。如果存在问题，可以修正后按照2.3节的方法重新部署 (2) 可以远程登录物业一体机，尝试ping设备IP地址。如果无法ping通，说明设备IP地址有问题，或者内网存在问题，需要找项目的弱电施工方解决；如果可以ping通，需要找技术同学进一步排查解决



3.3.5.3.20. 京希人行设备

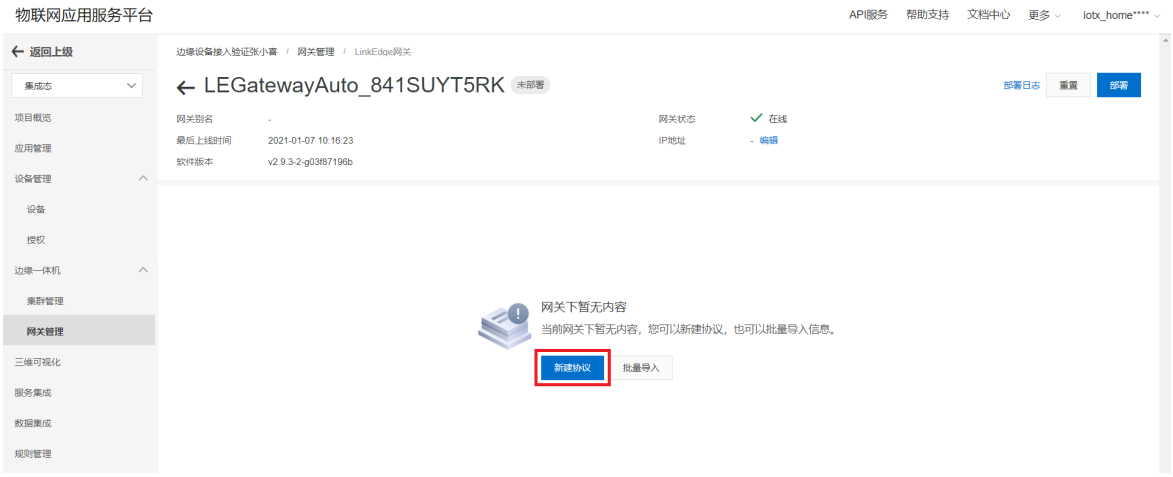
说明

京希支持的型号

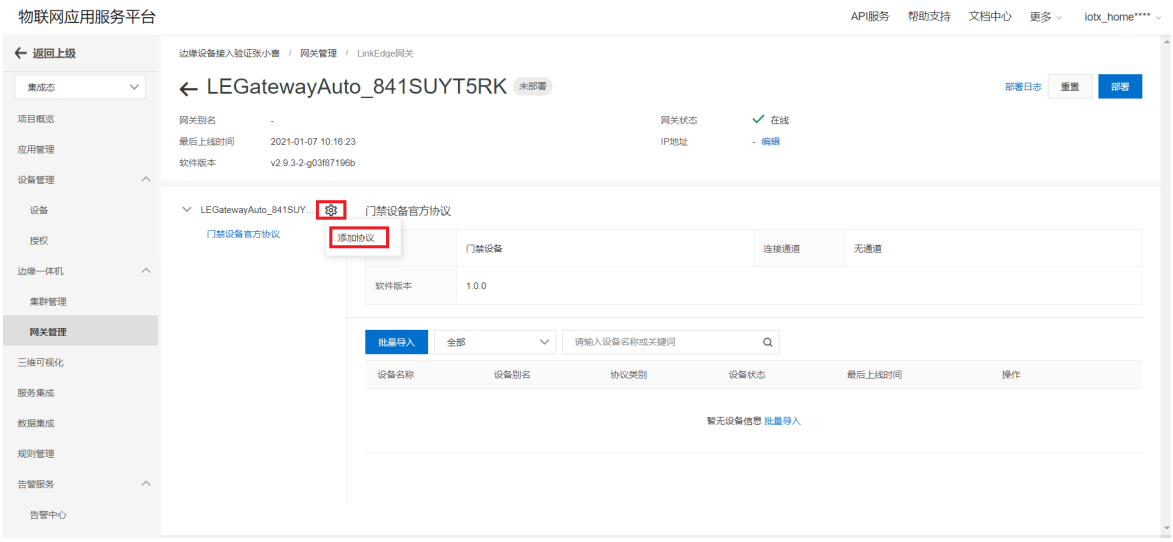
对接型号支持	BH9840L10A-FDS5
接入模式	门禁设备
支持功能	人脸/二维码/刷卡
系统版本	N/A
人脸识别方式	边侧识别
门禁端口号	12348

门禁设备集成

1. 添加门禁协议
   - 第一次添加协议，可以点击**新建协议**按钮，如下图所示



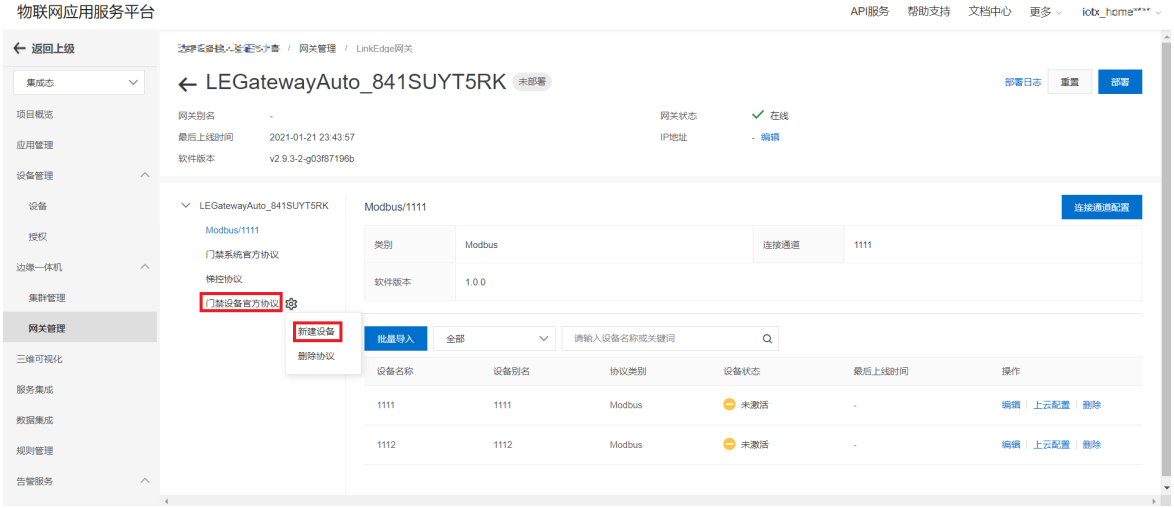
- 如果已经添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



- 京希的接入模式是设备接入，在弹出的对话框里，选择**门禁设备官方协议**，如下图所示

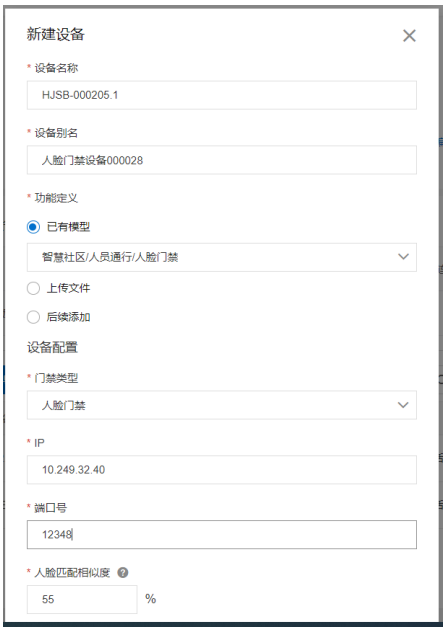


2. 添加门禁设备
   - 如下图所示，鼠标移动到到刚添加的**门禁设备官方协议**，点击**新建设备**，进行人行设备的添加。

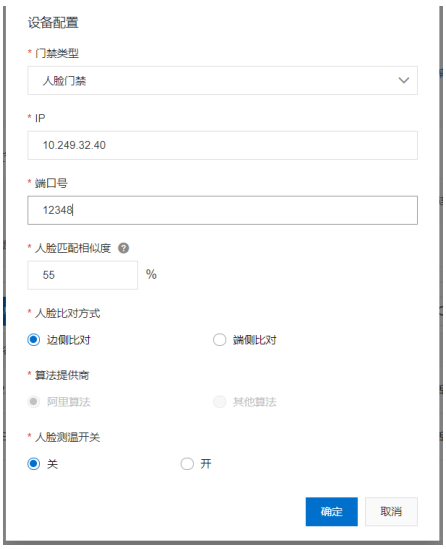


- 如图所示，在官方协议下新增人脸门禁设备。其中：
  - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。人脸门禁配置示例为：HJSB-000205.1。其中"**HJSB-000205**"是该设备的IoT编码，"**.1**"标识人脸门禁设备
  - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：人脸门禁设备000028

- **功能定义**：选择已有模型，对应设备名称选择 "智慧社区-人员通行-人脸门禁"



- **门禁类型**：选择“**人脸门禁**”
- **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
- **端口号**：**京希**门禁设备对外提供的网络访问端口号固定是12348
- **人脸匹配相似度**：输入**55%**。
- **人脸比对方式**：**京希**人脸比对方式为**边侧比对**
- **人脸测温开关**：**京希**门禁设备不支持人脸测温，设置为关



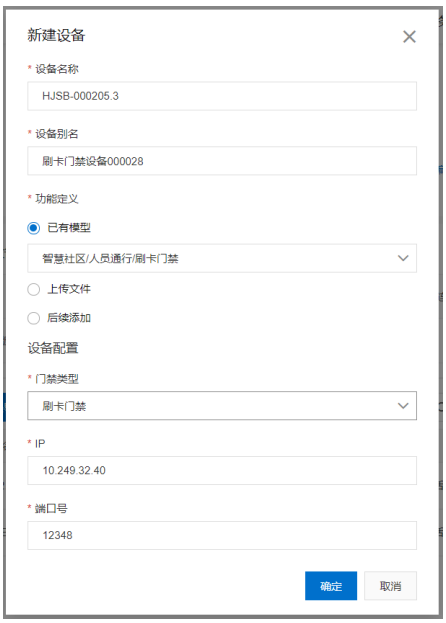
- 如图所示，在官方协议下新增二维码门禁设备。其中：
  - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。二维码门禁配置示例为：HJSB-000205.2。其中"**HJSB-000205**"是该设备的IoT编码，".2"标识二维码门禁设备
  - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：二维码门禁设备001

- **功能定义**：选择已有模型，对应设备名称选择智慧社区-人员通行-二维码门禁。



- **门禁类型**：选择“二维码门禁”
- **IP**:参照甲方提供的<设备信息表>中的该门禁**设备IP**。输入示例为10.249.32.40
- **端口号**：**京希**门禁设备对外提供的网络访问端口号固定是12348

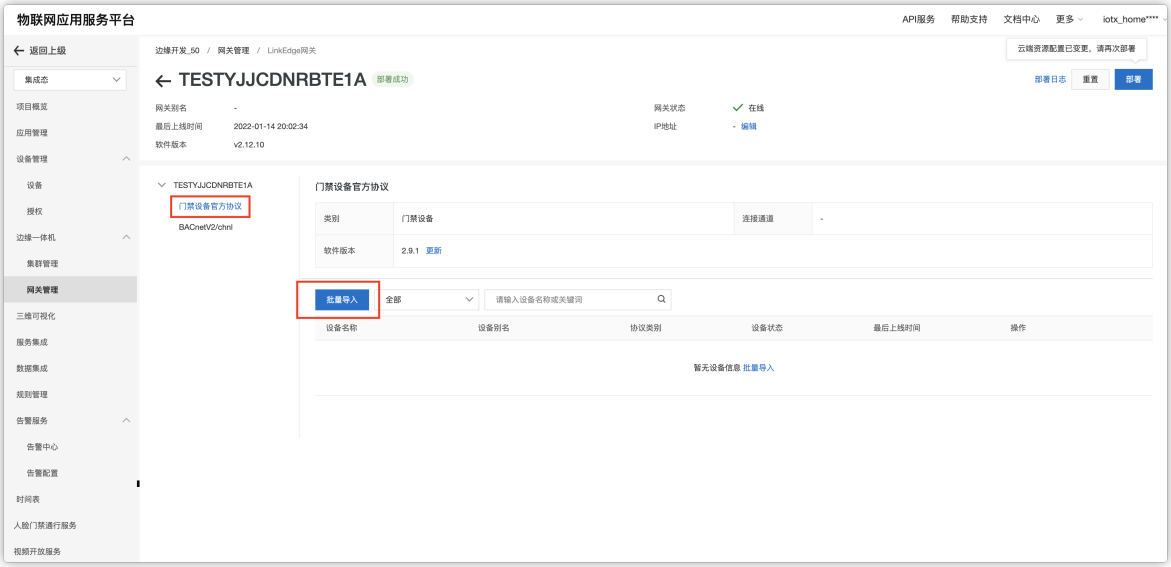
- 如图所示，在官方协议下新增刷卡门禁设备。其中：
  - **设备名称**：参照甲方提供的<设备信息表>中的门禁设备的**IoT编码**，每个门禁设备的IoT编码不同。刷卡门禁配置示例为：HJSB-000205.3。其中**"HJSB-000205"**是该设备的IoT编码，**".3"**标识刷卡门禁设备
  - **设备别名**：参照甲方提供的<设备信息表>中的**设备名称**。配置示例为：刷卡门禁设备000028
  - **功能定义**：选择已有模型，对应设备名称选择智慧社区-人员通行-刷卡门禁。



  - **门禁类型**：选择“刷卡门禁”
  - **IP**:参照甲方提供的<设备信息表>中的该门禁设备IP。输入示例为10.249.32.40
  - **端口号**：**京希**门禁设备对外提供的网络访问端口号固定是12348

3、**批量导入门禁设备**

1. 选择门禁设备官方协议，出现如下界面，点击批量导入



此时会弹出批量导入界面，在该界面点击下载批量导入模板文件



下载成功后，打开该文件，该模板文件是excel表格，每一列都会有填写说明，如下图所示，



设备名称	设备别名	设备品类	门禁类型	IP	端口	人脸匹配相识度	人脸比对方式	人脸识别算法服务地址IP	人脸识别算法服务地址端口号	人脸测温开关	人脸测温下限(C°)	人脸测温上限(C°)
HJSB-000205	门禁设备000028	测温人脸门禁	人脸门禁	192.168.2.3	4000	55	端侧对比	192.168.21.45	8081	开	36.5	39.8

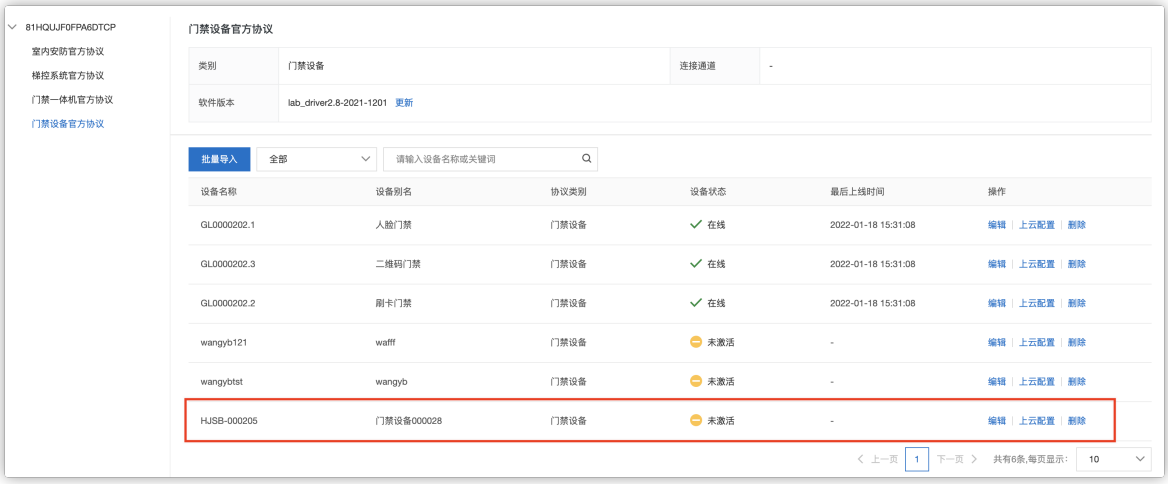
其中**设备名称**、**设备别名**、**设备品类**、**门禁类型**、**IP**、**端口**为必填项，**设备品类**和**门禁类型**，**人脸比对方式**，**人脸测温开关**提供下拉框选项

填入设备信息并进行保存，保存成功后，在批量导入窗口点击上传附件，上传成功后点击确认，进行导入





导入成功后，会自动刷新设备列表，如果导入失败，界面会有相关错误提示信息



3.3.5.3.21. 门禁一体机设备接入

1 登录

使用阿里云账号登录物联网应用服务平台，点击项目管理，选择项目：



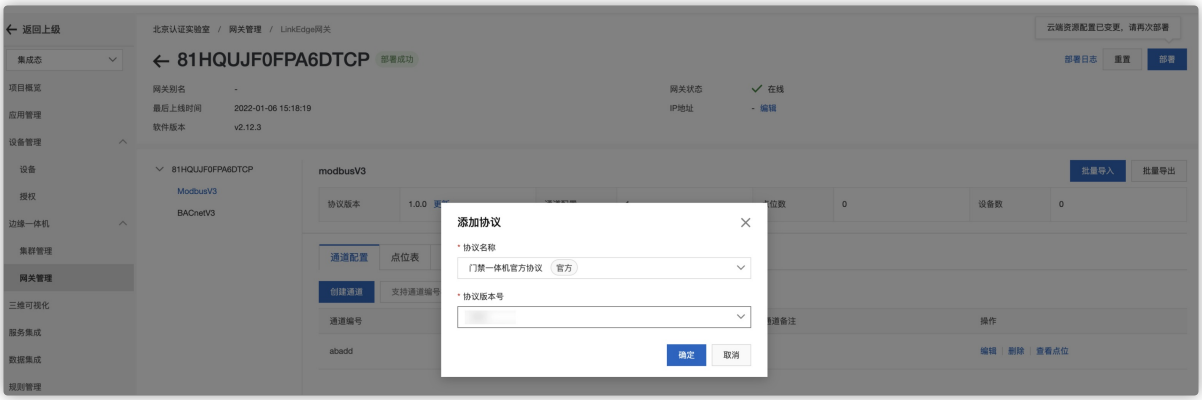
选择**边缘一体机>网关管理>设备集成**：



2 设备集成

2.1 添加门禁一体机官方协议

首先添加门禁一体机官方协议，并选择最新版本



添加成功以后如下图所示



2.2 添加门禁一体机设备

添加设备分为两种方式，单个添加和批量添加，该方式为单个设备添加

鼠标放在左侧门禁一体机官方协议上，会弹出添加设备的菜单，



点击新建设备，则进入如下界面，填写对应信息



设备名称：门禁设备名称，每个门禁设备名称要求不能相同。例如：HJSB-000205

设备别名：设备别名一般可以填写设备相关的备注信息。例如：门禁设备000028

品牌型号：根据项目采用设备的具体情况，选择相应的品牌和型号。

设备配置：IP:门禁设备的IP地址填写，输入示例为192.168.1.110

点击品牌型号，会弹出品牌型号下拉列表，可通过该下拉框进行品牌型号的选择，下拉框同时提供搜索功能，可按照品牌或者型号关键字输入查询(大小写敏感)，当鼠标放在具体型号上的时候，会有浮动窗口显示该型号设备具备的能力，如下图所示



当选择的型号支持边侧人脸的时候，会在界面自动增加算法配置项，如下所示



算法配置:IP：端侧设备人脸算法服务器的IP地址。输入示例为192.168.1.110

算法配置:端口：端侧设备人脸算法服务器的端口。输入示例为80

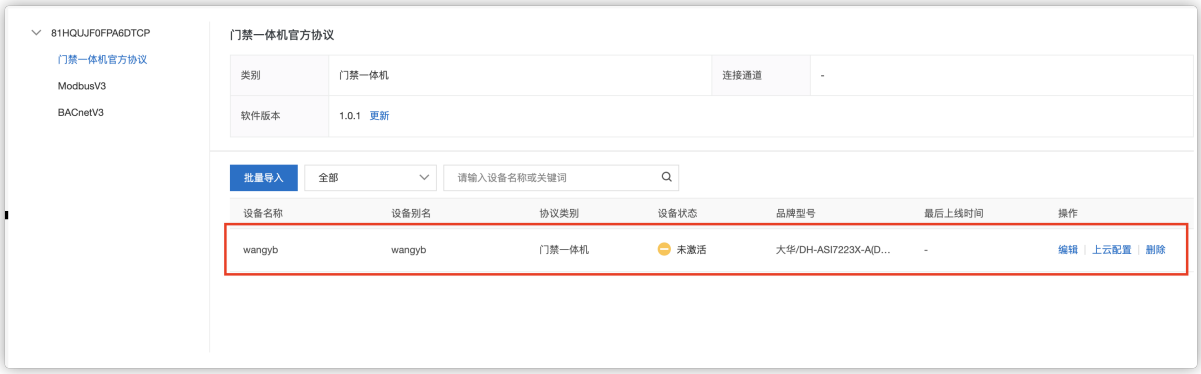
算法配置:人脸匹配度：人脸比对判定标准。输入示例为55%

当选择的型号包含梯控能力的时候，会增加如下配置项，如下图所示：



如果门禁设备已经连接梯控则勾选，否则不勾选；设备部署完成后如又连接上梯控，可将该选型勾选上，重新进行部署。

填入相关信息后，点击确定会进行设备创建，创建成功则会在设备列表显示



2.3 批量导入门禁一体机设备

2.2节为单独添加设备。本节为批量导入设备，即一次性添加多台设备的操作。

点击批量导入按钮会弹出批量导入的对话框，如下图所示





下载导入模板，并填写相关信息，导入模板及填写示例如下图所示：



A	B	C	D	E	F	G	H	I	J	K	L
设备名称 (必填，支持英文字母、数字和特殊字符"_"、"-"、@、"."、"-"，长度限制4~32)	设备别名 (必填，支持中文、英文字母、数字和下划线，4~64个字符，中文算两个)	品牌 (必填，只能在下拉列表中选择)	型号 (必填，只能在下拉列表中选择)	IP (必填，设备接入IP)	算法服务器IP (当人脸识别算法为端侧识别时必填，填写算法服务器的IP)	算法服务器端口 (当人脸识别算法为端侧时必填，填写算法服务器的端口)	人脸匹配相识度 (当设备包含人脸识别功能时必填，范围是0-100的整数)	人脸测温开关 (当设置包含人脸测温功能时必填，只能从下拉框选择)	人脸测温下限(C°) (当人脸测温开关打开时必填，范围是30.0°C-45.0°C)	人脸测温上限(C°) (当人脸测温开关打开时必填，范围是30.0°C-45.0°C)	是否包含梯控能力 (包含填1,不包含填0)
HJSB-000205	东南角	大华	DH-ASI7213Y-V3(DUAN)	192.168.1.2	192.168.1.34	90	55				
HJSB-000206	西北角	立方	RF-FX300-H823TC	192.168.1.3			55				
HJSB-000207	5单元	通用	电梯网关	192.168.1.4			55				1
HJSB-000208	6单元	通用	电梯规则网关	192.168.1.5			55				0

捷顺
富士
立方
通力
悉尔
安居宝
宇视
立林
海康威视
霍尼韦尔
太川
华安视讯

其中**设备名称**、**设备别名**、**IP**字段为必填项，其他字段含义如下：

品牌：提供下拉框选择，该项必填。

型号：型号提供下拉框选项，选择品牌后，型号下拉框会自动更新为该品牌支持的型号，该项必填。

算法IP：当选择的型号支持端侧人脸识别时，该字段填写算法服务器的IP地址，程序会校验该字段信息，如果没有填，则会在进行导入的时候弹出错误信息。

算法端口：该字段为算法服务器的端口，规则同算法IP，范围是0-65535。

人脸匹配相识度：当选择的设备型号支持人脸功能时，该项必填，范围是0-100的整数，否则导入失败。

人脸测温开关：当选择的设备型号支持人脸测温功能时，该项必填，否则导入失败。

人脸测温上限、下限：当人脸测温开关为开时，上限和下限为必填项，范围是30.0-45.0，支持一位小数，且下限要低于上限，否则导入失败。

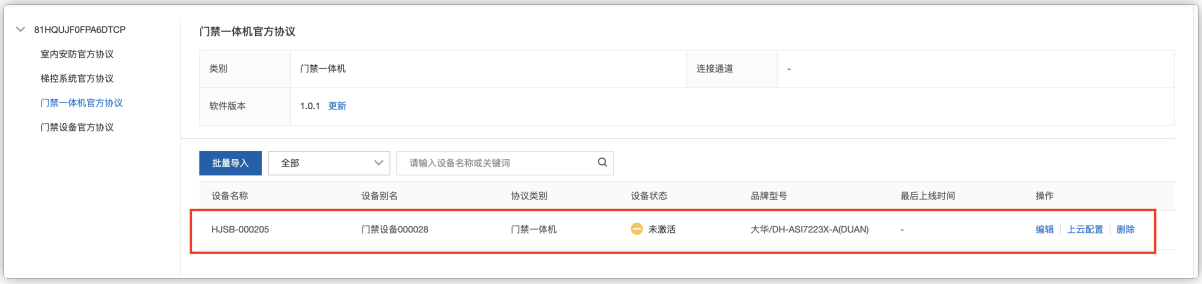
是否包含梯控能力：当设备型号支持梯控能力的时候，该项必填，如果门禁机已经连接梯控则选择1，否则选择0

选择品牌型号时，可以参考文档https://help.aliyun.com/document_detail/408806.html，查看品牌型号和能力的对应关系。

填写完设备信息后，上传文件并点击确定，开始导入，导入过程中不要刷新页面，否则会丢失进度显示，但是导入过程会正常进行



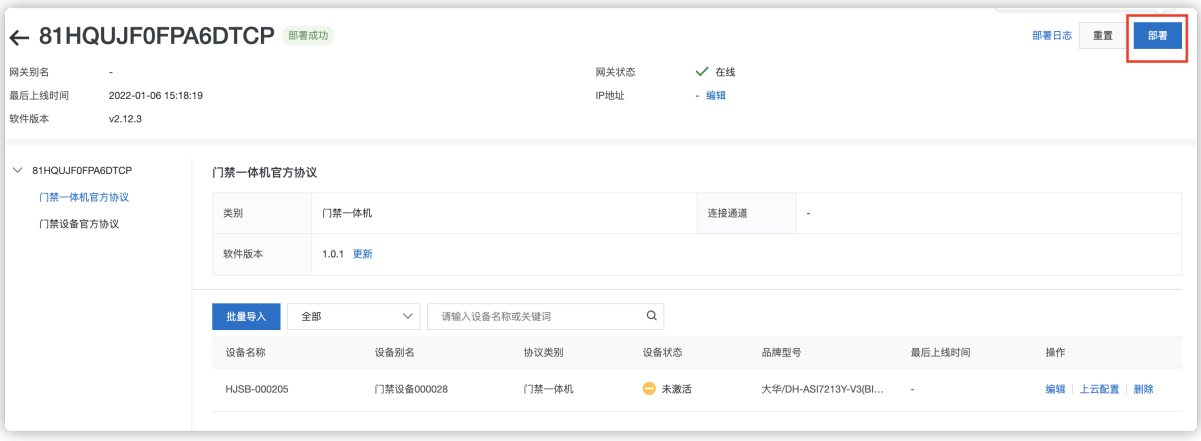
导入成功后会自动刷新设备列表，如出现导入的设备，则表示批量导入成功



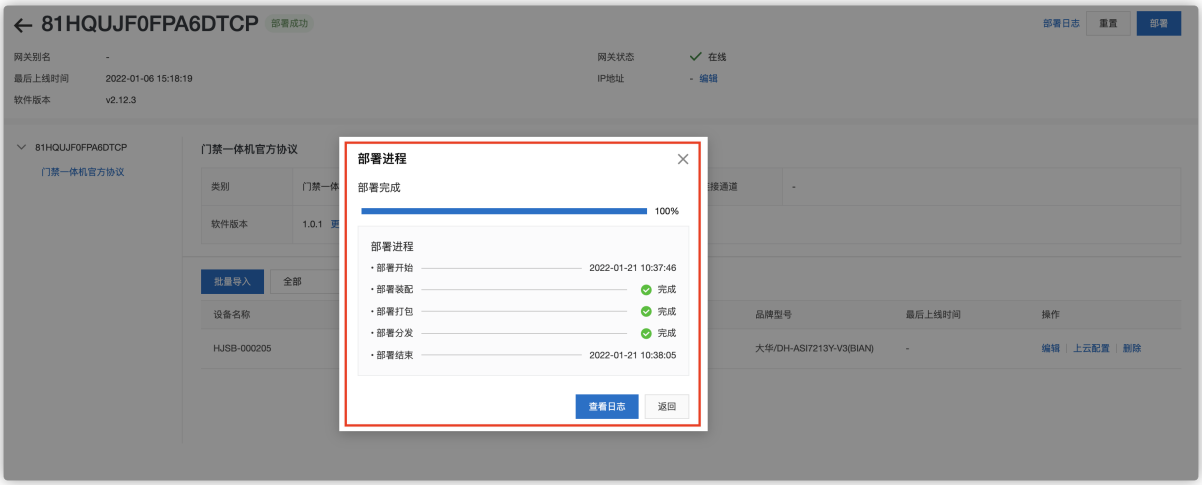
3 部署设备配置

在完成设备的配置后，点击部署，进行设备配置的协议下发到边缘服务器。如下图所示。

特别提醒：驱动部署时，网关侧对应驱动将重启，该驱动下的所有设备将出现离线后在线。这期间设备不可用，请部署人员提前与项目方沟通部署时间



点击部署后可以查看设备配置下发的状态和部署进度。



说明：如果某个设备有新更新的配置信息，重新点击部署后，只会下发有改动的设备配置协议，其余未修改的配置信息不会再次下发到边缘服务器，部署的时间根据修改的设备数量有关。

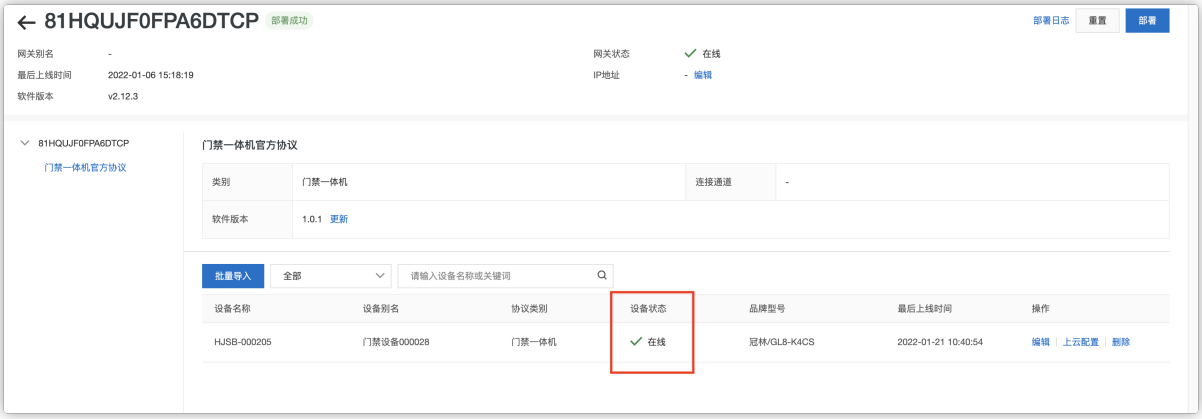
4 设备联通性验证

在进行设备联通性验证前，需要首先查验厂商的设备状态

1、厂商设备的软件版本已经升级到完成与阿里物业一体机对接过的版本。

2、厂商的设备IP地址设置正确，物业一体机可以正常ping通厂商设备。

设备添加并部署完成后，可以点击**边缘一体机-网关管理-门禁一体机官方协议**，逐个查看上面添加的所有设备状态，验证设备的联通性，如下图所示。如果状态是**在线**，则说明设备成功与边缘一体机连接成功；如果是未激活，说明设备与边缘一体机没有连接成功，这时可以排查的点：(1) 查看步骤2里设备的配置是否填写正确。如果存在问题，可以修正后点击重新部署； (2) 可以远程登录物业一体机，尝试ping设备IP地址。如果无法ping通，说明设备IP地址有问题，或者内网存在问题，需要找项目的弱电施工方解决；如果可以ping通，需要找技术同学进一步排查解决



3.3.5.4. 车行设备接入

3.3.5.4.1. 立方停车系统接入

1 立方停车系统支持信息说明

1.1 支持的型号信息

对接型号支持	立方停车系统版本：acs 4.2.0道闸型号：b35k 相机型号：rv140b

1.2 支持的停车场系统领域服务-服务模型

支持的服务	1.1 查询停车场信息1.2 查询停车场一级区域ID1.3 查询停车场道闸信息 2.1 查询停车区域信息2.2 查询区域下的道闸列表 3.1 查询道闸信息3.2 手动打开道闸栏杆 5.9 查询车辆入场时间7.1 车辆加入安全黑/白名单 7.2 车辆从黑/白名单移除7.3 车辆属于黑/白名单查询 8.1 新增车辆订单8.2 确认车辆订单

1.3 支持的停车场系统领域服务-数据模型

1.3.1 车辆通行(iot_park_pass_record)

字段	数据类型	立方停车系统是否支持
direction	Integer	支持
openType	Integer	不支持
plateNumber	String	支持
plateNumberImage	String	不支持
typePermission	Integer	不支持
plateColor	Integer	不支持
plateType	Integer	不支持
vehicleColor	Integer	不支持
vehicleType	Integer	不支持
barrierId	String	支持
barrierName	String	支持
parkingLotId	String	支持
areaId	String	不支持
orderNumber	String	支持
recordId	String	不支持
eventTime	String	不支持

1.3.2 收费事件(iot_park_fee)

字段	数据类型	立方停车系统是否支持
outBarrierName	String	不支持
outBarrierId	String	不支持
inBarrierName	String	不支持
inBarrierId	String	不支持
payStatus	Integer	不支持
orderNumber	String	支持
amount	Float	支持

outTime	String	不支持
inTime	String	支持
plateNumber	String	支持
outImage	String	不支持
inImage	String	不支持
areaId	String	不支持
parkingLotId	String	支持
eventTime	String	不支持

1.3.3 黑名单车辆识别(iot_park_blacklist_detect)

字段	数据类型	立方停车系统是否支持
direction	Integer	支持
plateNumberImage	String	不支持
plateNumber	String	支持
barrierId	String	支持
barrierName	String	支持
parkingLotId	String	支持
areaId	String	不支持
eventTime	String	不支持

2 立方适配器接入操作

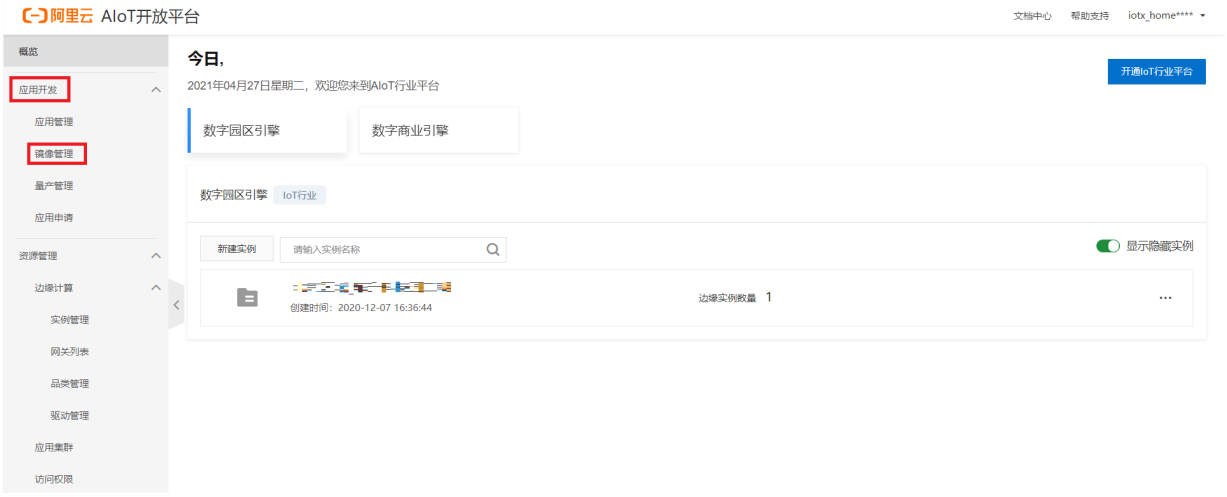
2.1 创建镜像仓库

2.1.1 登录平台

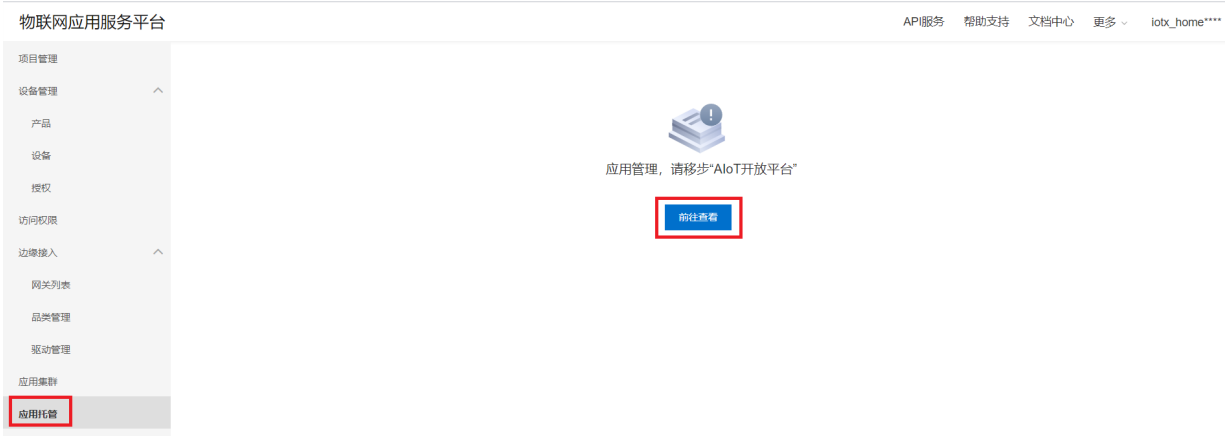
如果项目部署实施的平台是在数字园区引擎平台，使用阿里云账号登录AIOT开放平台，点击**开放平台**，如下图所示



点击**应用开发-镜像管理**，如下图所示：

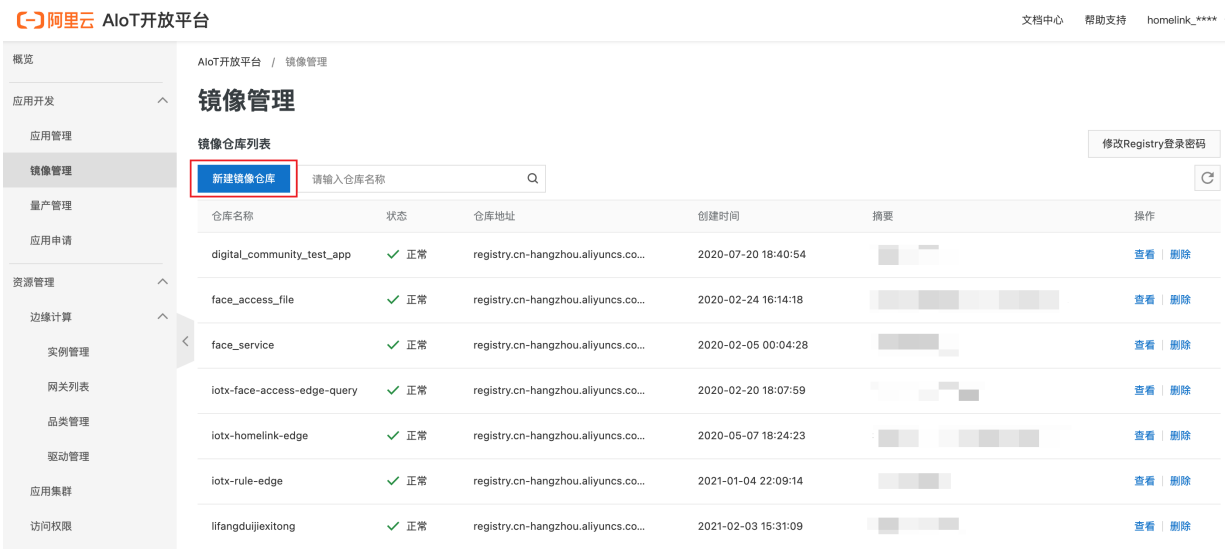


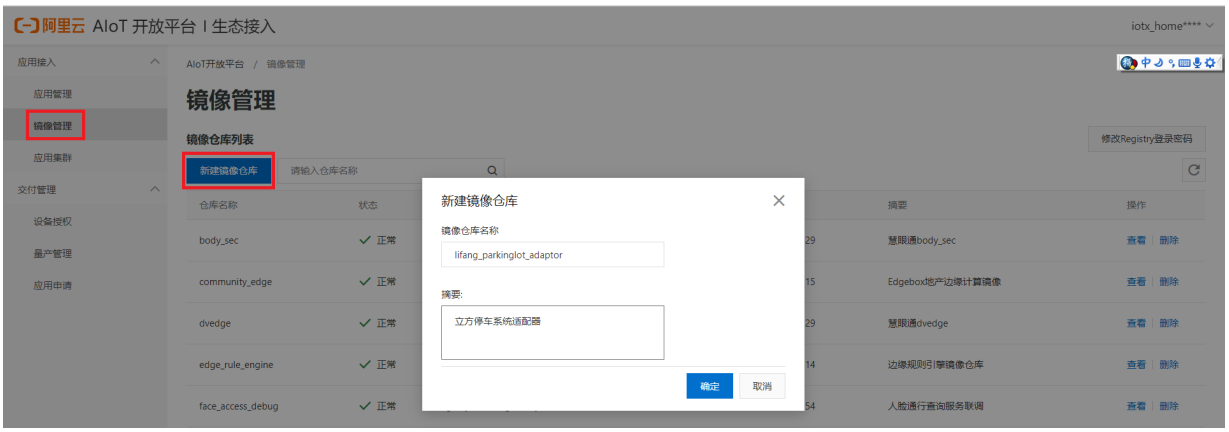
如果项目部署实施的平台是在物联网应用服务平台，使用阿里云账号登录，点击**应用托管 - 前往查看**，如下图所示



2.1.2 创建镜像仓库

选择**新建镜像仓库**，这个仓库用于存放制作好的镜像，每次往仓库推送一次镜像都是一个新的版本。





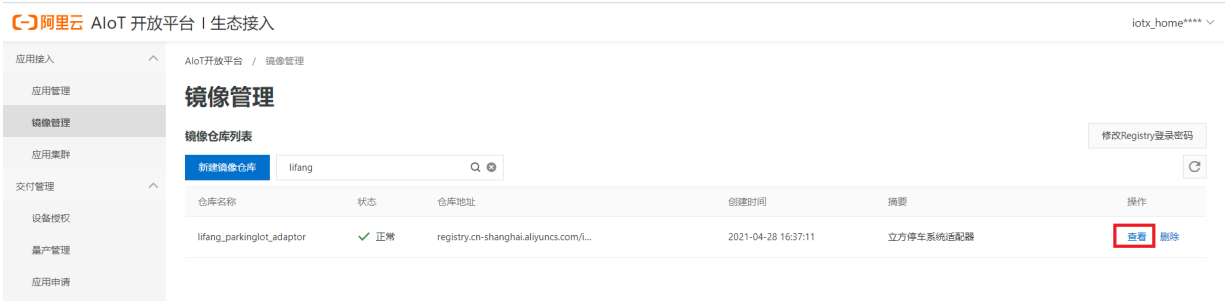
镜像仓库名称：填写： lifang_parkinglot_adaptor

摘要：填写：立方停车系统适配器

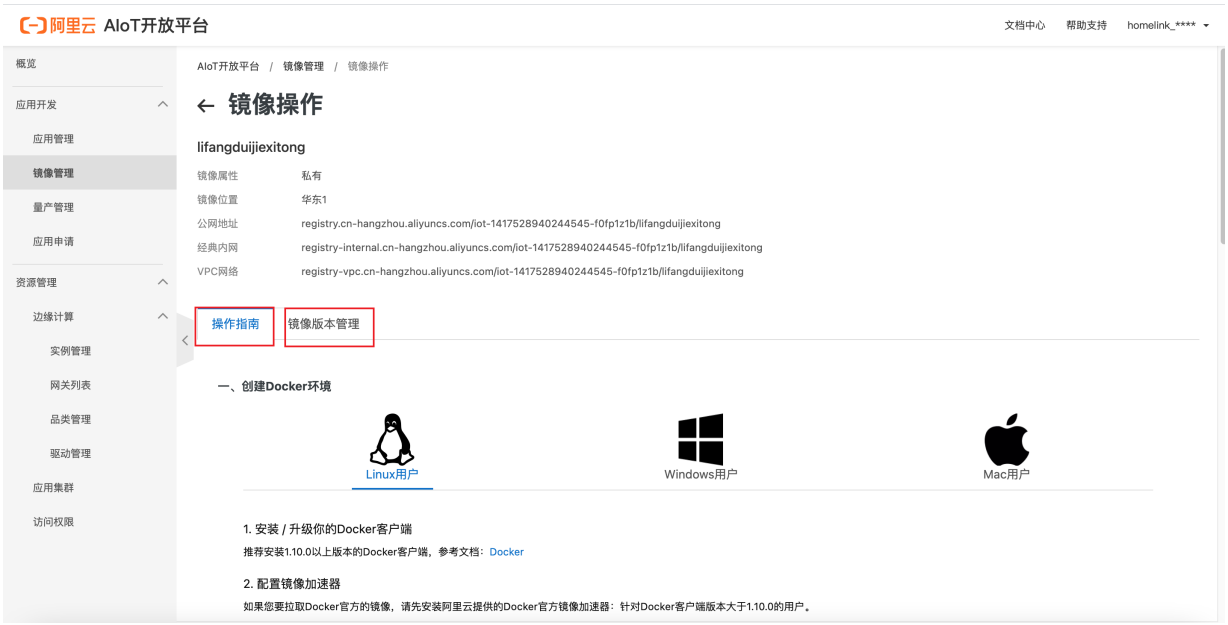
2.1.3 往仓库推送镜像

创建好镜像仓库之后，需要制作镜像并推送到这个镜像仓库。

首先在列表中找到刚新建完成的镜像仓库 lifang_parkinglot_adaptor，然后选择**查看**，如下图所示：



其中**操作指南**指示上传镜像的步骤，**镜像版本管理**列出了仓库中推送上来的所有镜像。



下面操作以Linux操作环境为例

2.1.3.1 制作Docker Image

前置条件1：Linux环境下安装了Docker工具

前置条件2：在Linux环境下将要操作的目录下，将下面两个文件拷贝到该目录。这两个文件从项目组获得

(1) parkinglot_adapter-1.0-SNAPSHOT.jar

(2) Dockerfile

在Linux目录下执行下面命令：

sudo docker build -f Dockerfile -t lifang_adaptor_image:v1.0.

其中：v1.0是版本号； lifang_adaptor_imag是镜像名称

```
[xiaoxi.zxx@iotx-homelinkedge-dev011158129043.na62 /home/xiaoxi.zxx/work/lifang container]
$sudo docker build -f Dockerfile -t lifang_adaptor_image:v1.0 .
[sudo] password for xiaoxi.zxx:
Sending build context to Docker daemon  38.25MB
Step 1/8 : FROM openjdk:8-jre-alpine
 ---> f7a292bbb70c
Step 2/8 : RUN apk add --no-cache tzdata bash tini
 ---> Using cache
 ---> 4b25669a6969
Step 3/8 : ENV TZ Asia/Shanghai
 ---> Using cache
 ---> 8cdede300266
Step 4/8 : RUN rm -rf /tmp/* /var/tmp/* $HOME/.cache
 ---> Using cache
 ---> 380263f73144
Step 5/8 : COPY parkinglot_adapter-1.0-SNAPSHOT.jar /
 ---> 4104d752ffa9
Step 6/8 : EXPOSE 10060
 ---> Running in 810f7d4406f6
Removing intermediate container 810f7d4406f6
 ---> f9e4b7d43e89
Step 7/8 : ENTRYPOINT ["/sbin/tini", "--"]
 ---> Running in 102b4b090a1e
Removing intermediate container 102b4b090a1e
 ---> 2f66a7966329
Step 8/8 : CMD java -XX:+UseContainerSupport -XX:InitialRAMPercentage=50.0 -XX:MaxRAMPercentage=80.0 -jar /pa
nline
 ---> Running in 5c2aba0ce171
Removing intermediate container 5c2aba0ce171
 ---> b176b64a5fad
Successfully built b176b64a5fad
Successfully tagged lifang_adaptor_image:v1.0
```

2.1.3.2 查看lifang_adaptor_image的image id

sudo docker images

```
[xiaoxi.zxx@iotx-homelinkedge-dev011158129043.na62 /home/xiaoxi.zxx/work/lifang_container]
$sudo docker images
REPOSITORY                                                TAG                 IMAGE ID            CREATED             SIZE
lifang_adaptor_image                                      v1.0                b176b64a5fad        22 seconds ago      128MB
linker_build                                              latest              1307696d7c6e        5 weeks ago         623MB
reg.docker.alibaba-inc.com/home-linkedge/linker_build     latest              1307696d7c6e        5 weeks ago         623MB
c-linkedge-sdk                                            latest              13481b898f98        5 weeks ago         651MB
```

找到lifang_adaptor_image对应的image id:b176b64a5fad

2.1.3.3 将镜像推送到Registry

说明：下面命令，必须是本节2.1.3开始找到的镜像仓库lifang_parkinglot_adaptor 页面里找到拷贝出的命令，如下图所示：



(1) 登录registry

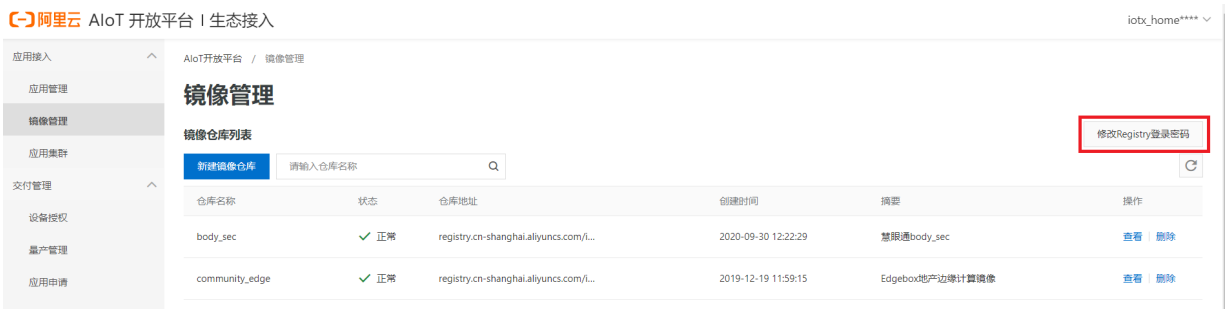
从上面红框里拷贝1对应的命令，最后添加镜像登录密码

sudo docker login --username=iotx_homelink registry.cn-hangzhou.aliyuncs.com -p镜像登录密码

```
[xiaoxi.zxx@iotx-homelinkedge-dev011158129043.na62 /home/xiaoxi.zxx/work/lifang_container]
$ sudo docker login --username=iotx_homelink registry.cn-hangzhou.aliyuncs.com -p[redacted]
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

镜像登录密码是从镜像管理右侧**修改Registry登录密码**设置并获取



(2) 打标签

从上面红框里拷贝2对应的命令，修改image id 和版本号。其中image id在2.1.3.2提供，版本号在2.1.3.1指定

sudo docker tagb176b64a5fadregistry.cn-shanghai.aliyuncs.com/iot-1102025233338260-drzc46k7/lifang_parkinglot_adaptor:1.0

如下图所示：

```
[xiaoxi.zxx@iotx-homelinkedge-dev011158129043.na62 /home/xiaoxi.zxx/work/lifang_container]
$sudo docker tag b176b64a5fad registry.cn-shanghai.aliyuncs.com/iot-1102025233338260-drzc46k7/lifang_parkinglot_adaptor:1.0
```

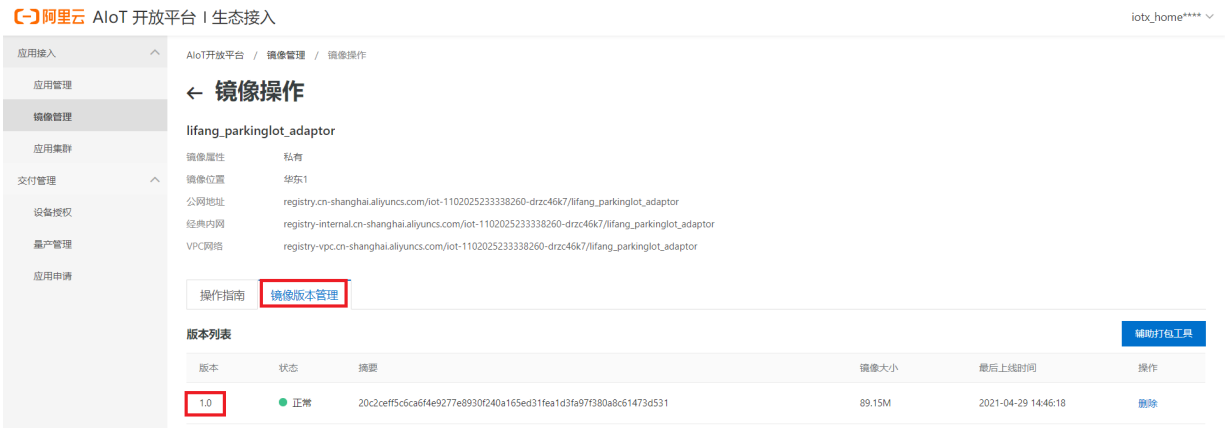
(3) 将镜像推送到Registry

从上面红框里拷贝3对应的命令，修改镜像版本号：

sudo docker push registry.cn-shanghai.aliyuncs.com/iot-1102025233338260-drzc46k7/lifang_parkinglot_adaptor:1.0

```
[xiaoxi.zxx@iotx-homelinkedge-dev011158129043.na62 /home/xiaoxi.zxx/work/lifang_container]
$sudo docker push registry.cn-shanghai.aliyuncs.com/iot-1102025233338260-drzc46k7/lifang_parkinglot_adaptor:1.0
The push refers to repository [registry.cn-shanghai.aliyuncs.com/iot-1102025233338260-drzc46k7/lifang_parkinglot_adaptor]
94e98046c0f4: Pushed
0abb9a65e7f3: Mounted from iot-1102025233338260-drzc46k7/homelinkedge
edd61588d126: Mounted from iot-1102025233338260-drzc46k7/homelinkedge
9b9b7f3d56a0: Mounted from iot-1102025233338260-drzc46k7/homelinkedge
f1b5933fe4b5: Mounted from iot-1102025233338260-drzc46k7/homelinkedge
1.0: digest: sha256:20c2ceff5c6ca6f4e9277e8930f240a165ed31fea1d3fa97f380a8c61473d531 size: 1370
```

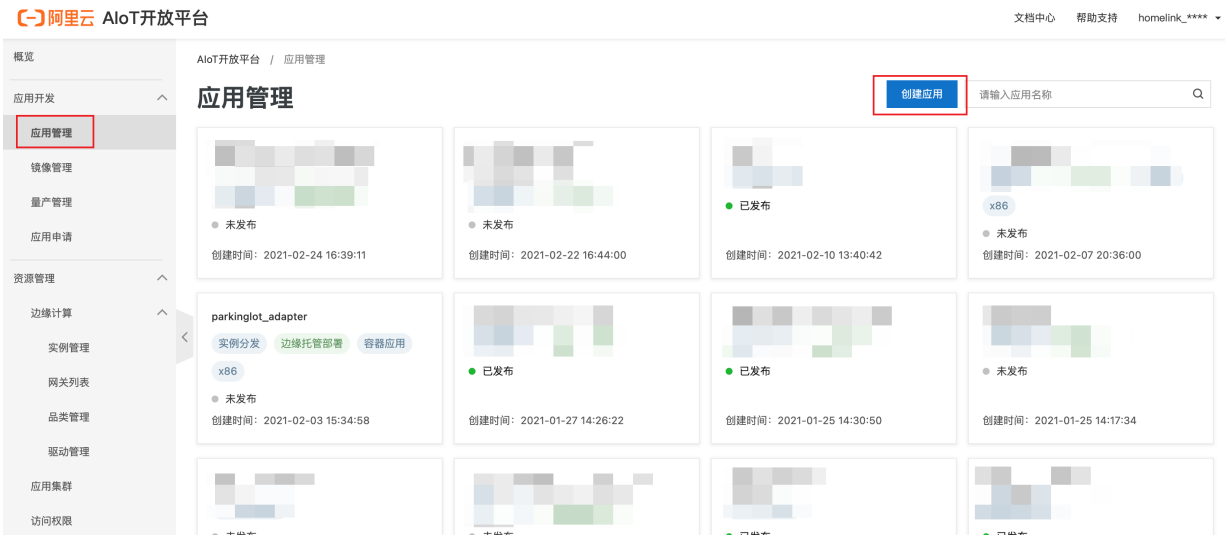
推送成功后，点击该镜像仓库的**镜像版本管理**，可以看见刚才推送成功的1.0版本



2.2 项目部署操作

2.2.1 创建应用

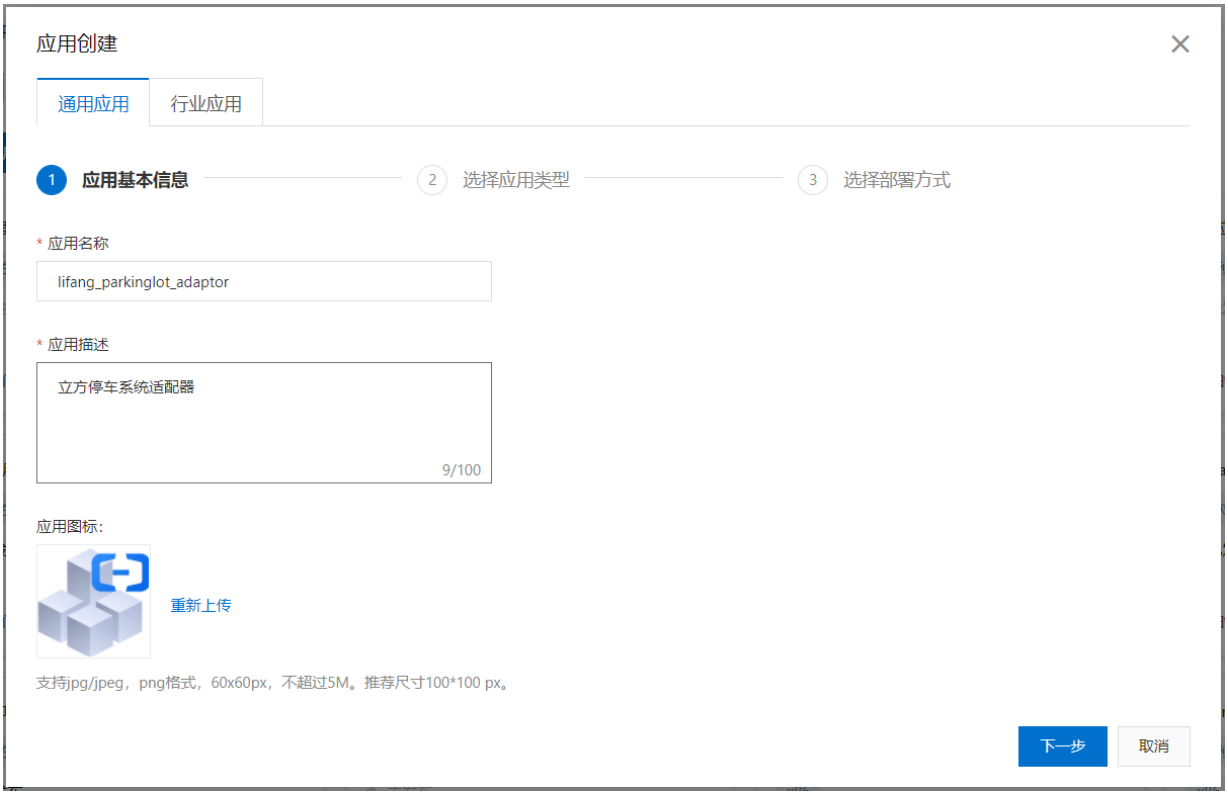
在左边菜单栏选择**应用管理**，然后选择**创建应用**，如下图所示：



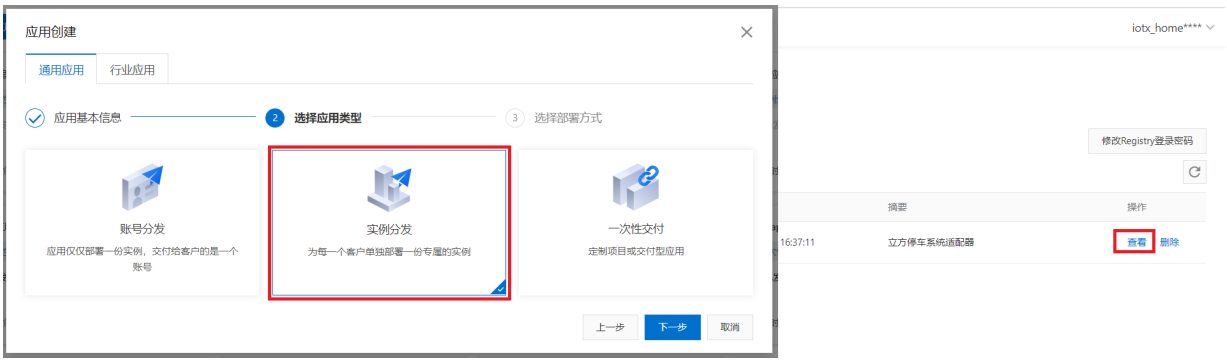
弹出页面如下图所示：

应用名称：填写：lifang_parkinglot_adaptor

应用描述：填写：立方停车系统适配器



下一步，然后选择**实例分发**，如下图所示

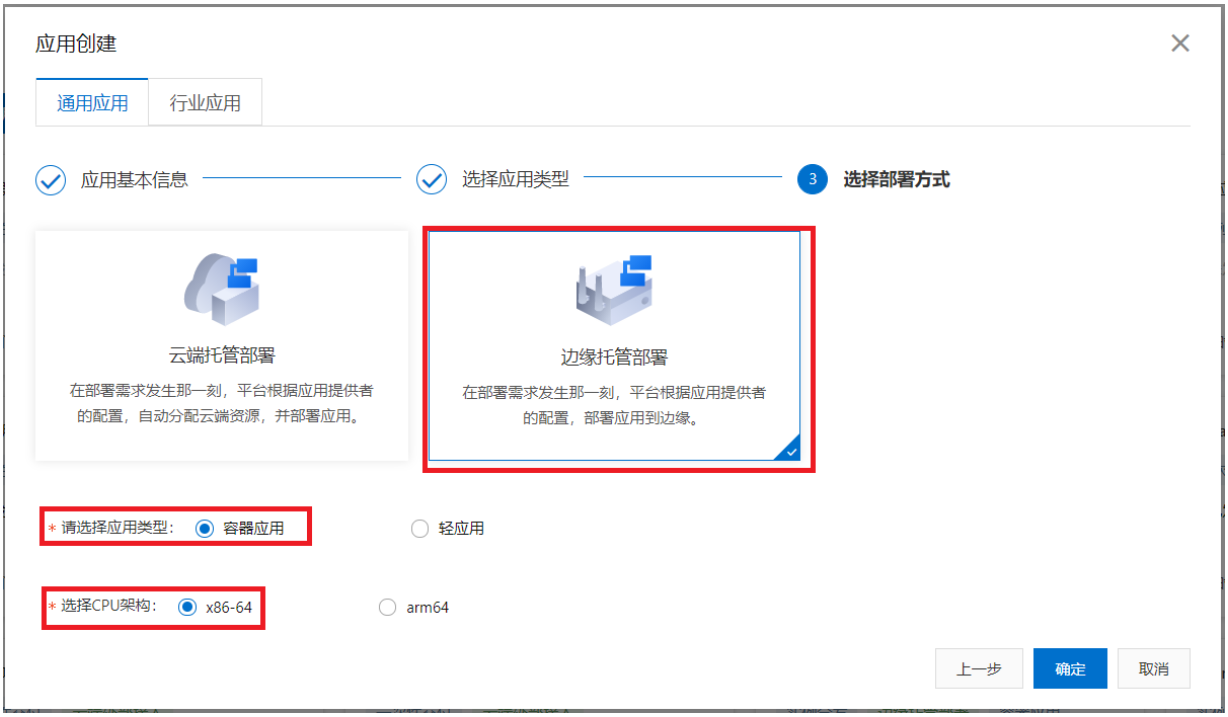


弹出页面如下图：

部署方式选择：**边缘托管部署**

请选择应用类型：**容器应用**

选择CPU架构：**x86-64**



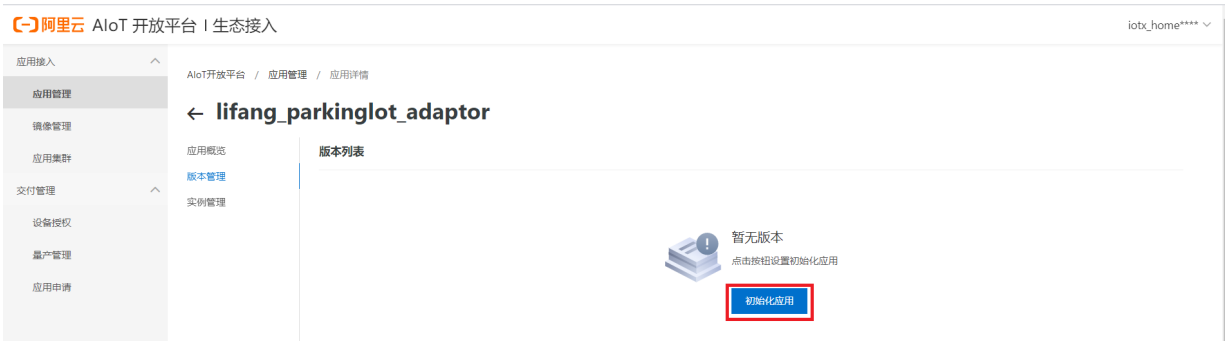
点击确定，完成应用创建

2.2.2 可视化编排

在列表中找到2.2.1 创建好的应用lifang_parkinglot_adaptor，



点击进入该应用，选择**版本管理**。



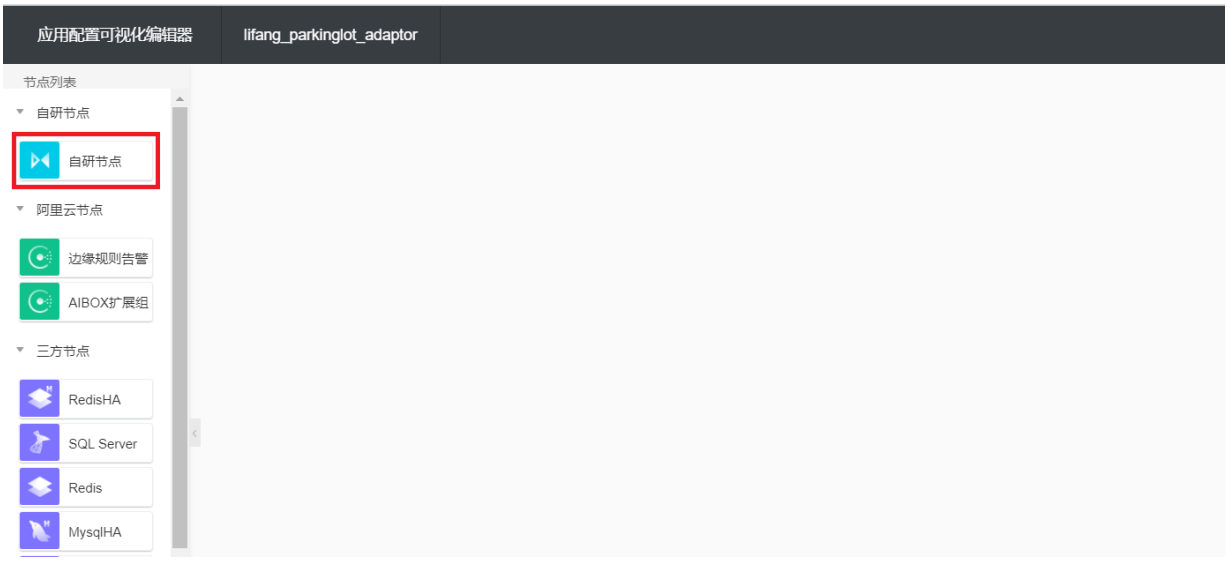
点击**初始化应用**



确定后，在**版本管理**下，点击**可视化编排**，如下图所示：



鼠标左键点中自研节点，拖拽到右侧



下面将填写配置信息：

服务名称：iotxparkingservice

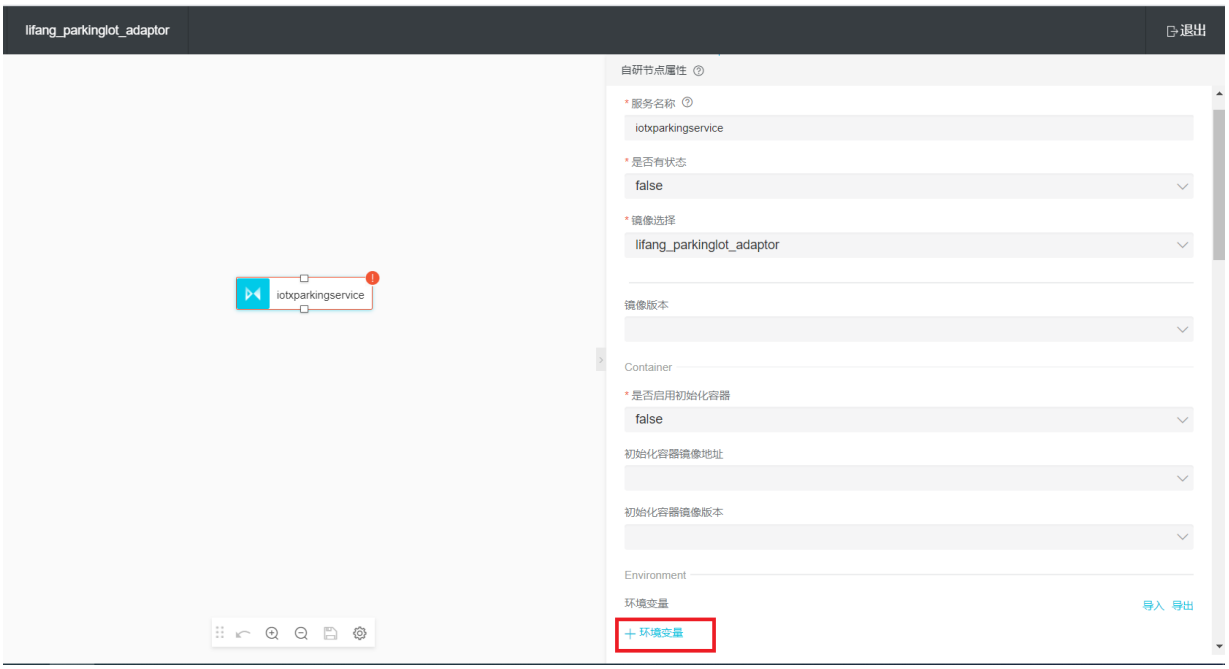
是否有状态：不用更改，使用缺省：**false**

镜像选择：选择2.2.2创建的镜像：lifang_parkinglog_adaptor

镜像版本：可以选择最新版本：1.0

是否启用初始化容器：不用更改，使用缺省：false

如下图所示：



下面进行环境变量配置，点击**环境变量**，如上图所示：

adapter_port： 适配器应用对外端口，固定填写：10060

lfDataBase_ip: 立方停车系统的内网IP地址，需根据项目实际情况填写。这里填写示例为192.168.100.100

lfDataBase_port：立方停车系统数据库对外端口，固定填写：3306

lfDataBase_un：立方数据库访问用户：固定填写：root

lfDataBase_pwd：立方数据库访问密码，固定填写：lf0507

lf_ip：立方停车系统的内网IP地址，需根据项目实际情况填写。这里填写示例为192.168.100.100

lf_port：立方停车系统对外端口号，固定填写：9988

lf_subPath：立方接口访问路径，固定填写：/Parking/Handheld/



点击Network下面的**自研节点边缘端口对**，如下图所示



协议：选择TCP

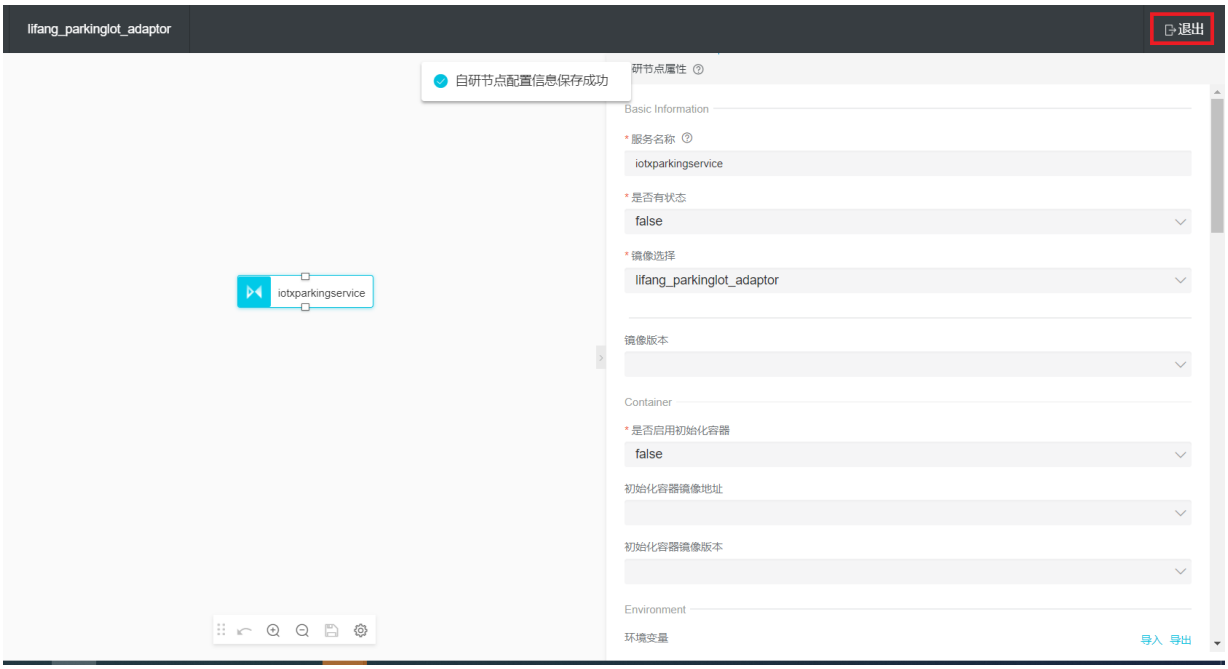
服务端口：固定填写 10060

应用自身端口：固定填写 10060

勾选固定主机端口，如上图所示。**主机端口**：固定填写 10060

服务类型：从下拉列表选择NodePort

其他直接使用默认参数。点击右上角**退出**：



选择保存并退出，如下图所示



2.2.3 模型与权限

可视化编排完成之后，在该应用**版本管理**里选择**模型与权限**，如下图所示：



2.2.3.1 添加数据模型

点击**数据模型**，**添加数据模型**，如下图所示：



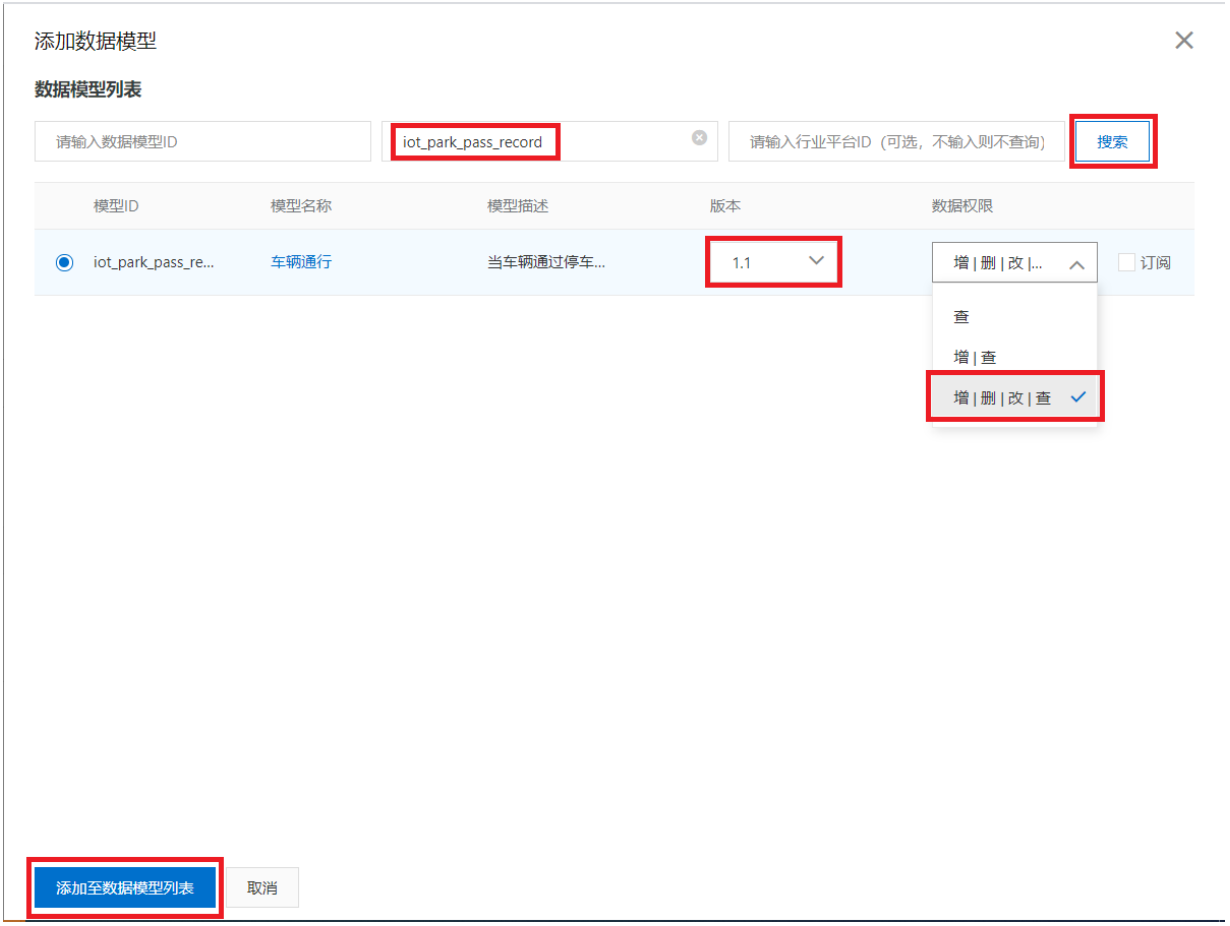
立方停车系统的适配器目前支持下面三种数据模型，需分别添加：

1 **车辆通行**：iot_park_pass_record

2 **收费事件**：iot_park_fee

3 黑名单车辆识别：iot_park_blacklist_detect

添加**车辆通行**数据模型：在数据模型名称搜索框里输入iot_park_pass_record，点击搜索



在搜索出对应数据模型，版本选择最新版本，数据权限选择增删改查，最后点击添加至数据模型列表，如上图。

然后依次添加**收费事件/黑名单车辆识别**数据模型，操作与上面类似。最后结果如下：

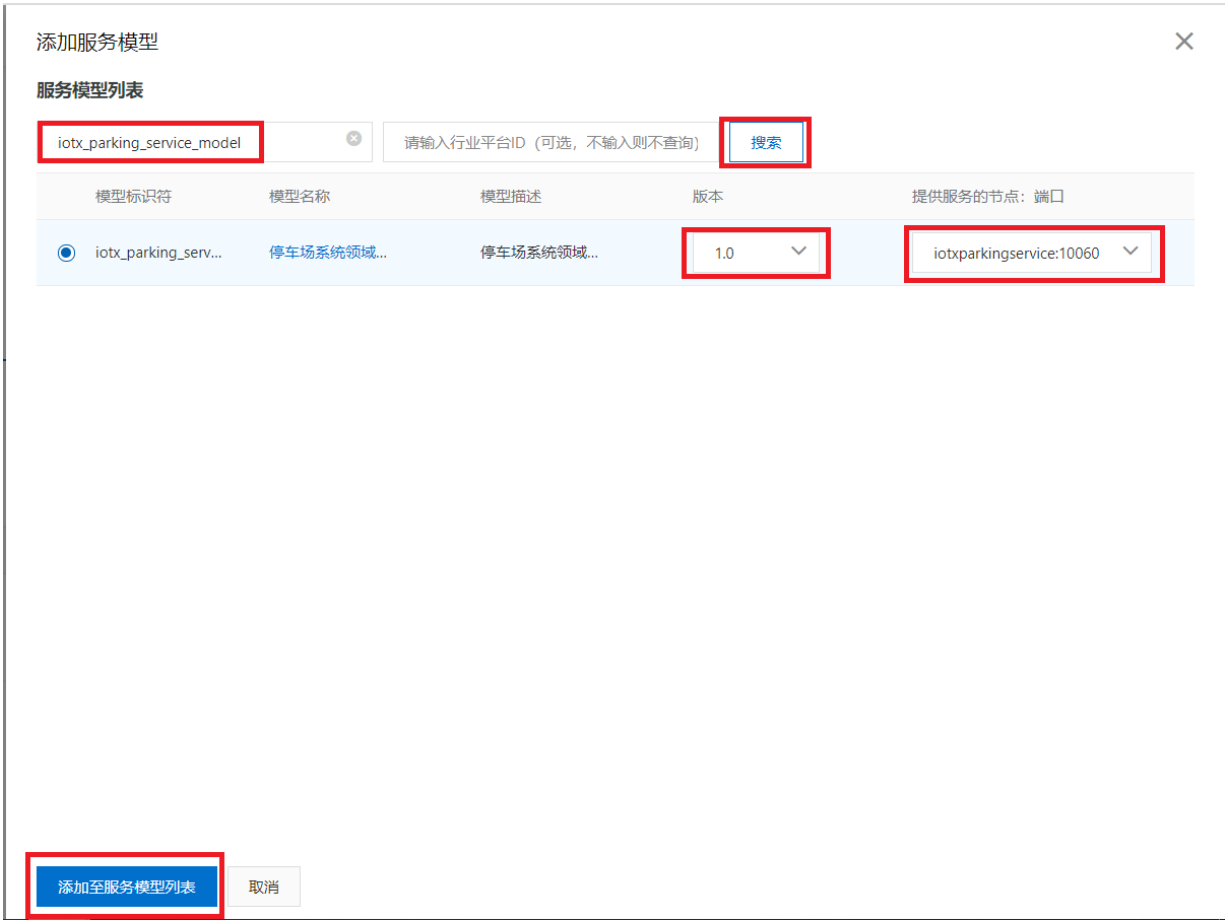


2.2.3.2 添加服务模型

点击**服务提供**，**添加服务模型**，如下图所示



在服务模型标识符搜索框里输入iotx_parking_service_model，这个是立方停车适配器支持的停车服务模型标识符，点击搜索，如下图所示



搜索出停车系统领域服务，版本选择最新的，“提供服务的节点：端口”选择10060，最后点击左下角添加至**服务模型列表**。添加完成后如下图所示：



服务模型授权完成以后，云端可以通过服务模型调用应用的接口，从而应用向云端提供服务。

2.2.4 应用版本发布

点击**版本管理**，**更多**下点击**发布版本**，如下图所示

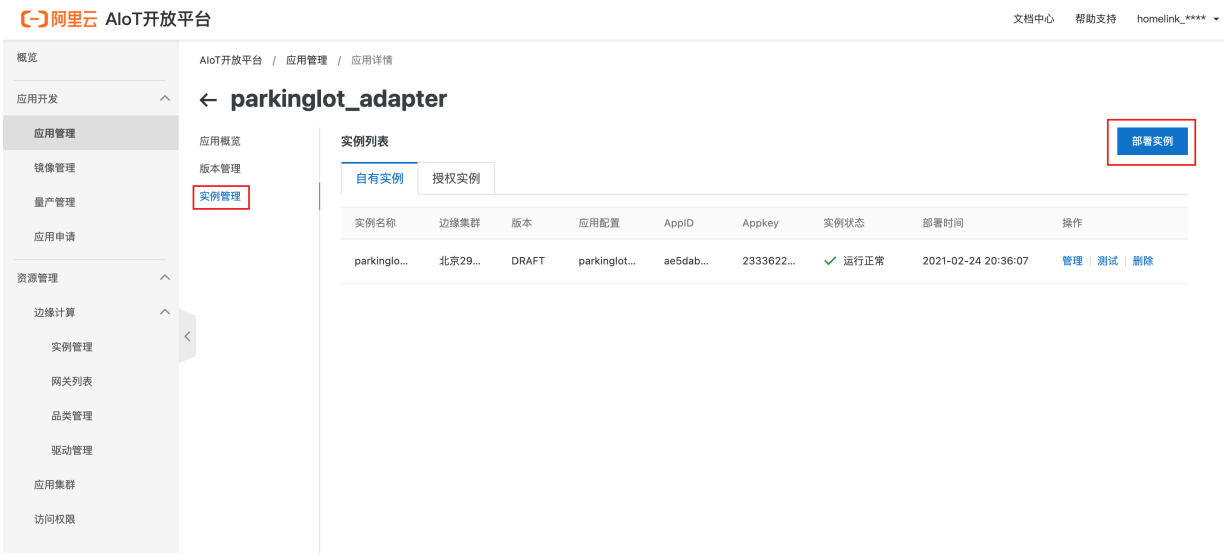


2.3 部署实例

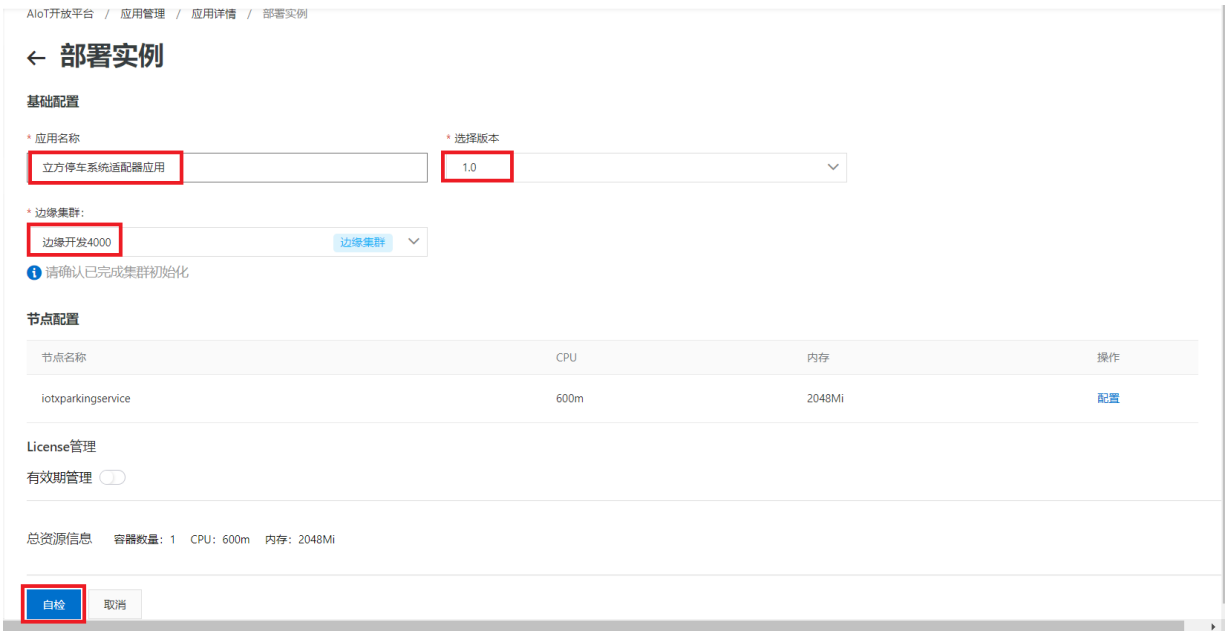
完成了准备工作，那么就开始正式部署。在**应用管理**中，选择**实例管理**，选择**部署实例**，即可进行实例部署。

2.3.1 部署实例

点击**实例管理**，选择**部署实例**，正式开始部署实例。



弹出下面页面：

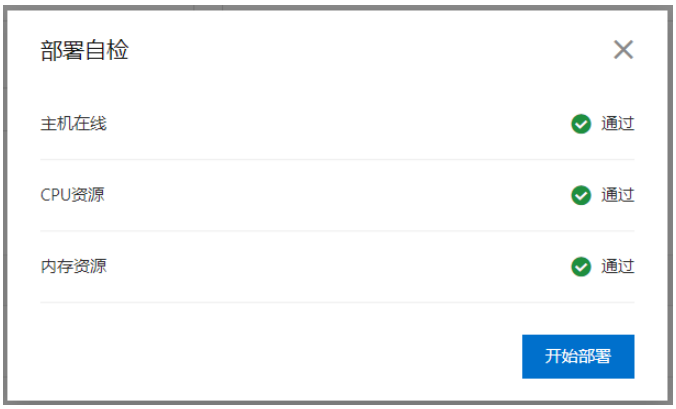


应用名称：可以输入：立方停车系统适配器应用

选择版本：1.0

边缘集群：请与项目获取该应用将推送到边缘集群名称

点击**自检**



自检通过后，点击**开始部署**

3 适配器应用升级

3.1 重新上传镜像

同名的镜像 需要改版本

旧的： sudo docker build -f Dockerfile -t [镜像名称]:1.0 .

新的： sudo docker build -f Dockerfile -t [镜像名称]:1.1 .

不同的名称的镜像可以使用相同版本号

之后在把最新的镜像上传到仓库即可

3.2 可视化编排

简介：版本管理分为已发布的版本管理和未发布的版本管理，已发布的版本管理不能再次进行可视化编排选择最新的镜像了，只有通过下面已发布的流程进行操作。

已发布

流程操作：版本管理 - 更多（应用升级） -重新选择上传的镜像

更多应该是有个应用升级的按钮的，图片只是示例点击哪个位置。



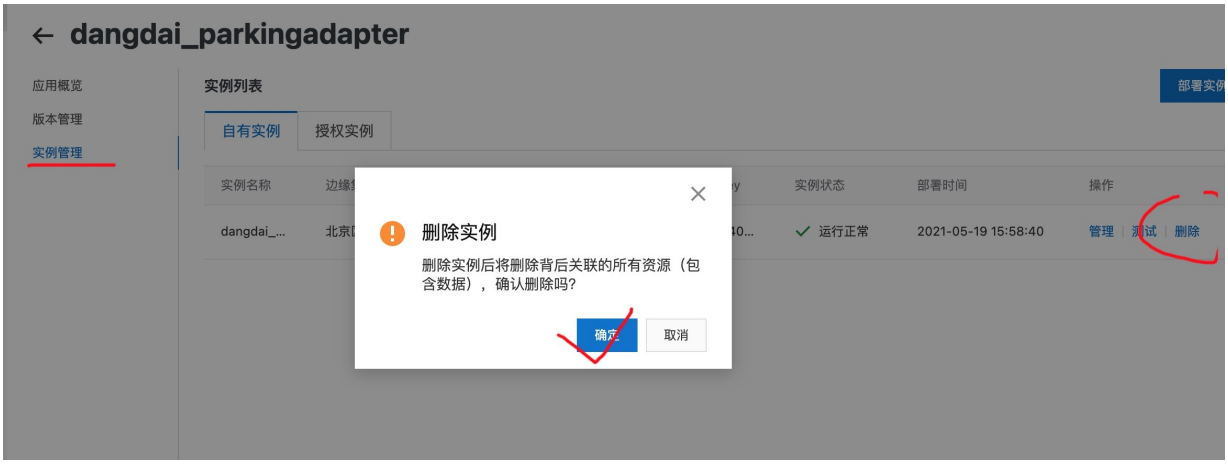
未发布

编排重新选择镜像即可



3.3 删除实例重新部署

删除操作



重新部署：见2.3.1

4 服务模型调试和数据模型数据查看

4.1 服务模型调试

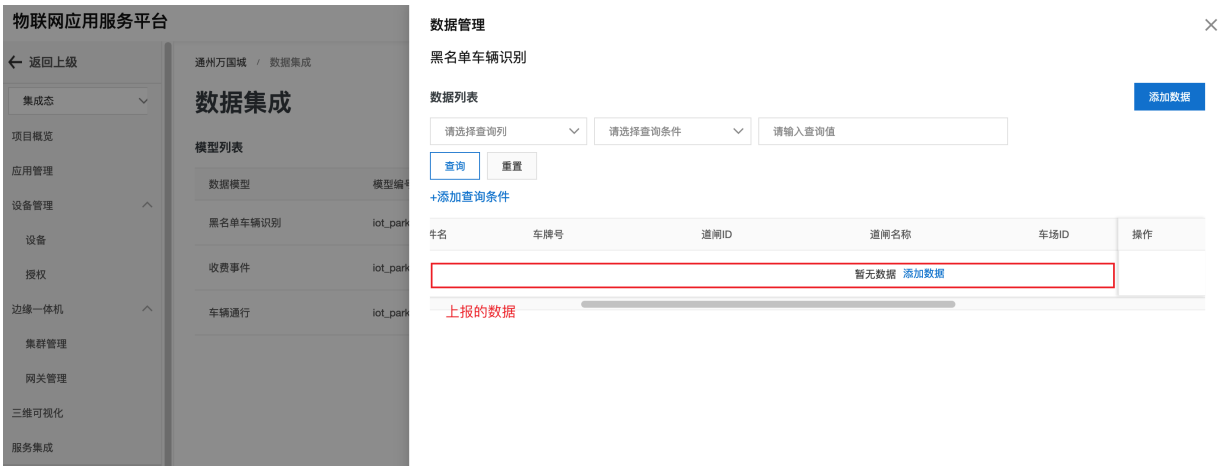
应用创建好了以后，再**模型和权限**中，点击**服务提供**，将服务模型授权给应用，此时适配器应用就和服务模型关联起来了，接下来就可以进行服务的调式。



4.2 数据模型数据查看

同理，再**权限和模型**中，点击**数据模型**，将已经发布的数据模型授权给应用，这个时候数据模型就和应用关联到一起了，可以在再**项目**下查看应用上报的数据；再SI平台点击**项目管理**，选择对应的**项目**，选择**集成态**，找到**数据集成**，就可以查看对应数据模型上报的数据了；





3.3.5.5. 安防设备接入

3.3.5.5.1. 安防设备接入

名词解释

下面对文档中使用的专业名词分别说明，便于使用手册的人更好的理解

名词	说明
物联网应用服务平台	阿里巴巴IoT智能化解决方案，能够集成的大量行业智能硬件和软件，快速打通OT与IT，并通过容器化技术，实现应用及解决方案的自动化部署和交付
IPC	IPC(IP Camera)即网络摄像头
NVR	NVR(Network Video Recorder)即网络硬盘录像机)，一般指项目上采购的海康、宇视等NVR设备
设备证书	设备证书包括ProductKey（PK）、DeviceName(DN)和DeviceSecret(DS)，为IoT平台识别设备的唯一标识

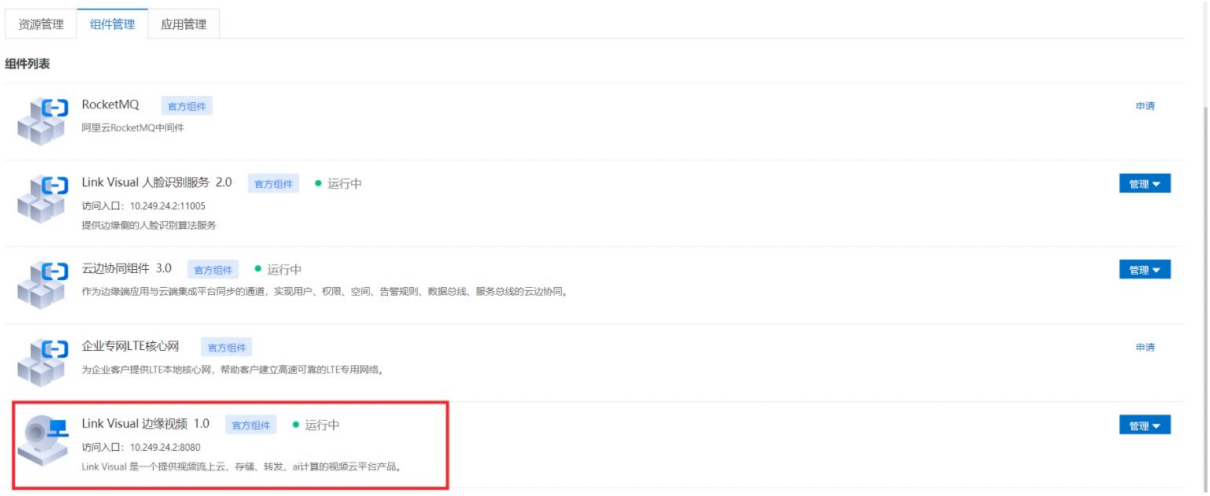
1 海康支持对接的型号说明

功能	硬盘录像机(NVR)	摄像机

对接型号支持	DS-8632N-E8-V2	DS-2CD222XD-BRCDS-2CD232XD-BRCDS-2CD252XP-BRCDS-2CD2626FD-BRCDS-2DE62THZS-BRCDS-2CD2726FWDA2-IZSDS-2CD7727FWD-LZDS-2XA8247F-LZS

2 应用服务平台创建产品和设备

在安装部署视频监控安防设备前，请确保项目的边缘集群已经部署了Link Visual 边缘视频组件。可以集群管理里查看，如下图所示：



本文后面统一将LinkVisual边缘视频组件简称为LV边缘视频

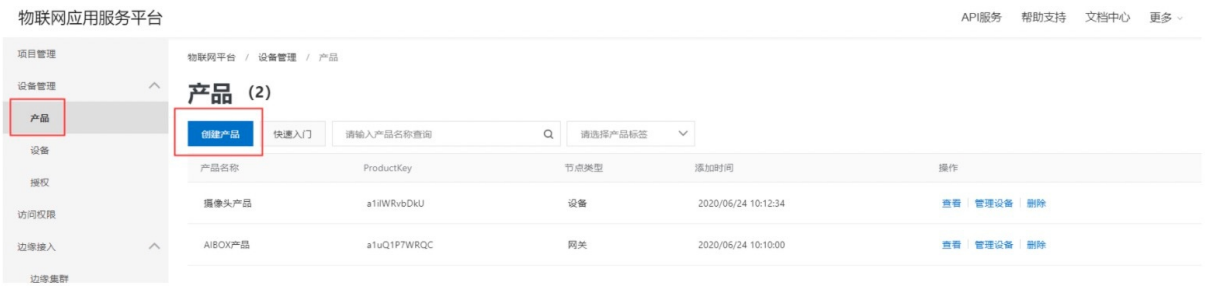
2.1 创建LV边缘视频网关产品和设备

LV边缘视频(AIBOX)，最多可最多接入256个安防摄像头设备。如果项目上接入的摄像头数目超过256，需要申请多个AIBOX

2.1.1 创建LV边缘视频网关产品

登录阿里云应用服务平台，“**设备管理 - 产品 - 创建产品**”，如下图所示。

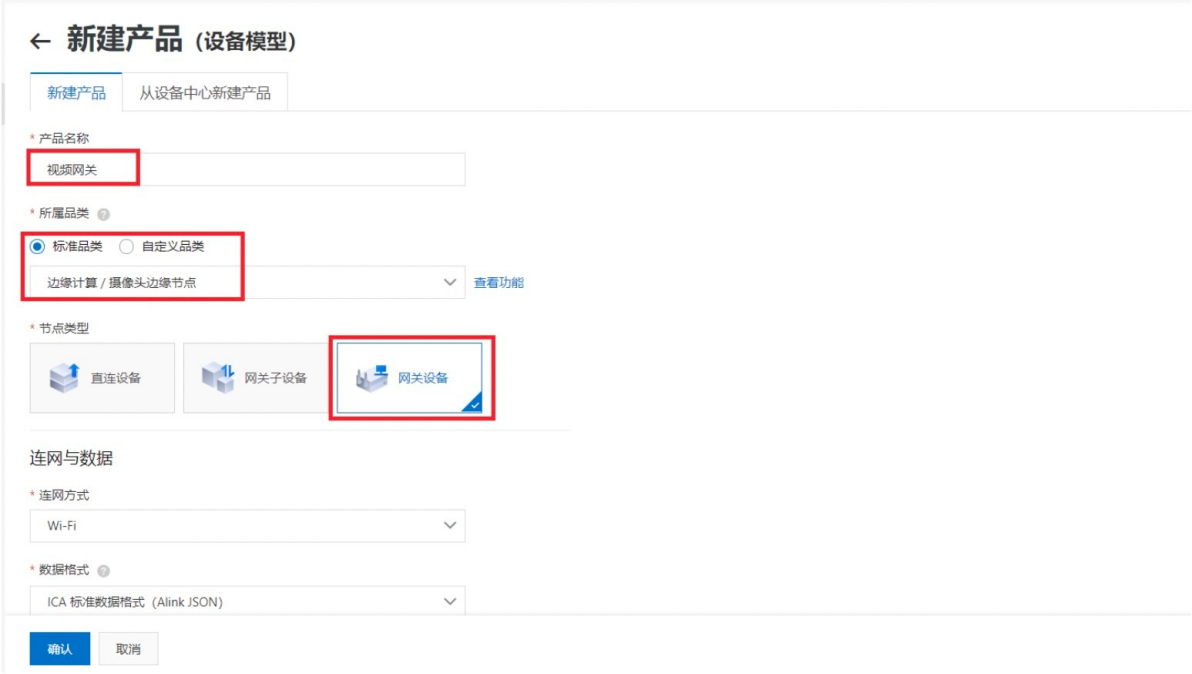
说明：如果一个账号下已经创建了LV边缘视频网关产品，请忽略此步骤。请不要在同一个账号下重复创建LV边缘视频网关产品。在阿里云主账号或者下面的一个RAM子账号创建的产品，该主账号和所有RAM子账号都可以看见该产品



产品名称：建议名称：视频网关

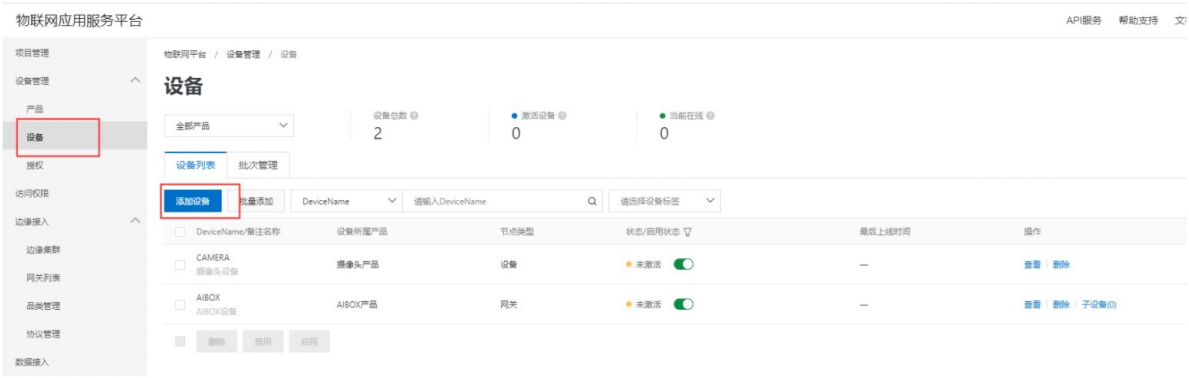
所属品类：请选择“标准品类：边缘计算/摄像头边缘节点”

节点类型：请选择“网关设备”，如下图所示



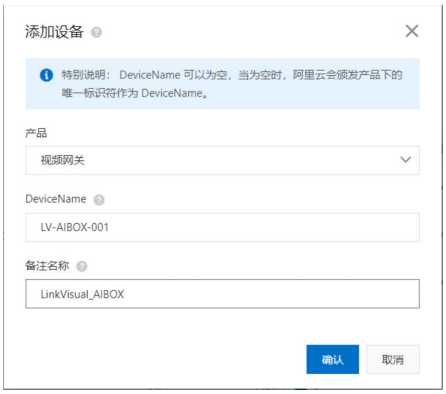
2.1.2 创建LV边缘视频网关设备

“**设备管理 - 设备 - 添加设备**”，如下图所示。



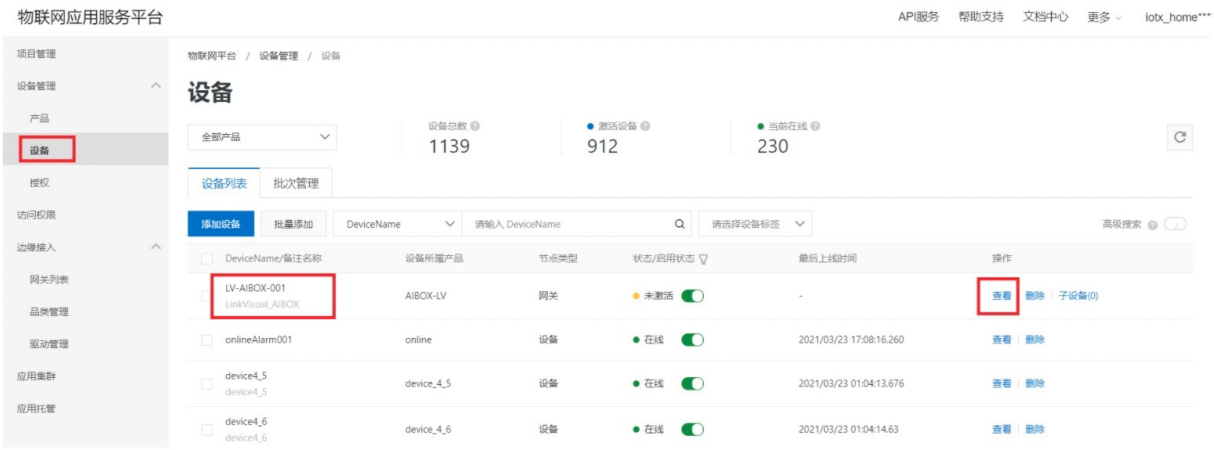
产品：选择之前在3.1.1流程中创建的LV边缘视频网关产品:视频网关，如下图所示。

DeviceName:根据甲方给出的规则定义产品名称。建议输入规则：品牌+功能+设备编号。例如：LV-AIBOX-001

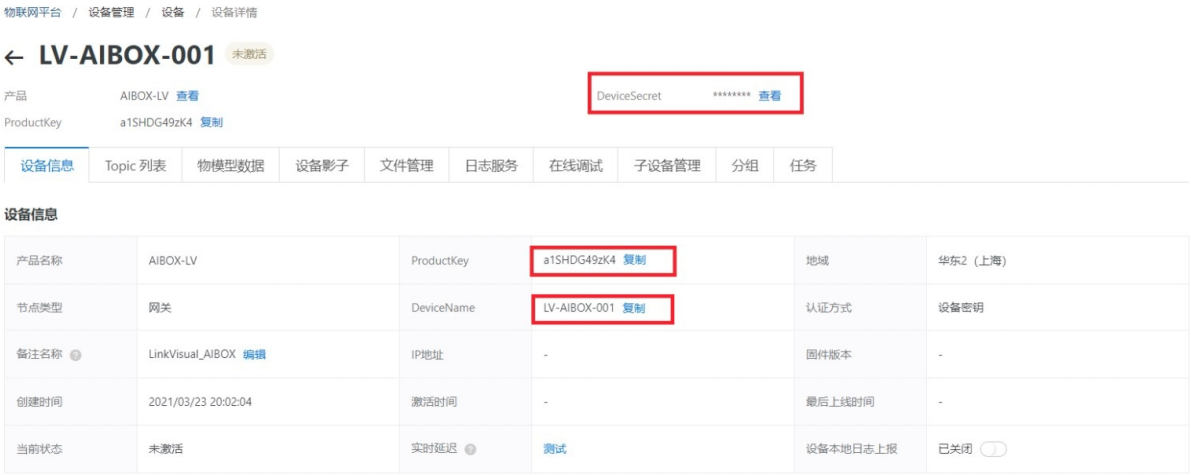


2.1.3 查看LV边缘视频网关设备三元组

查看LV边缘视频网关三元组方式：在“设备管理-设备”中，找到刚才创建的LV边缘视频网关设备，点击**查看**，如下图所示。



ProductKey、DeviceName、DeviceSecret三个数据即为设备唯一标识，本篇文档中统称为设备三元组，如下图所示。

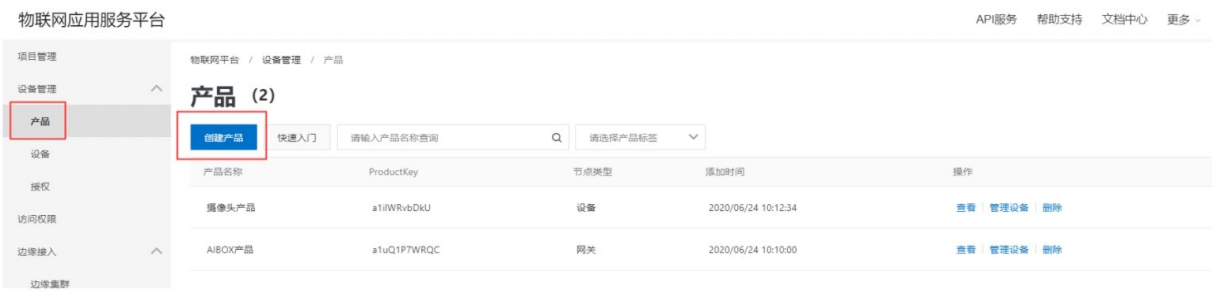


2.2 创建摄像头产品和设备

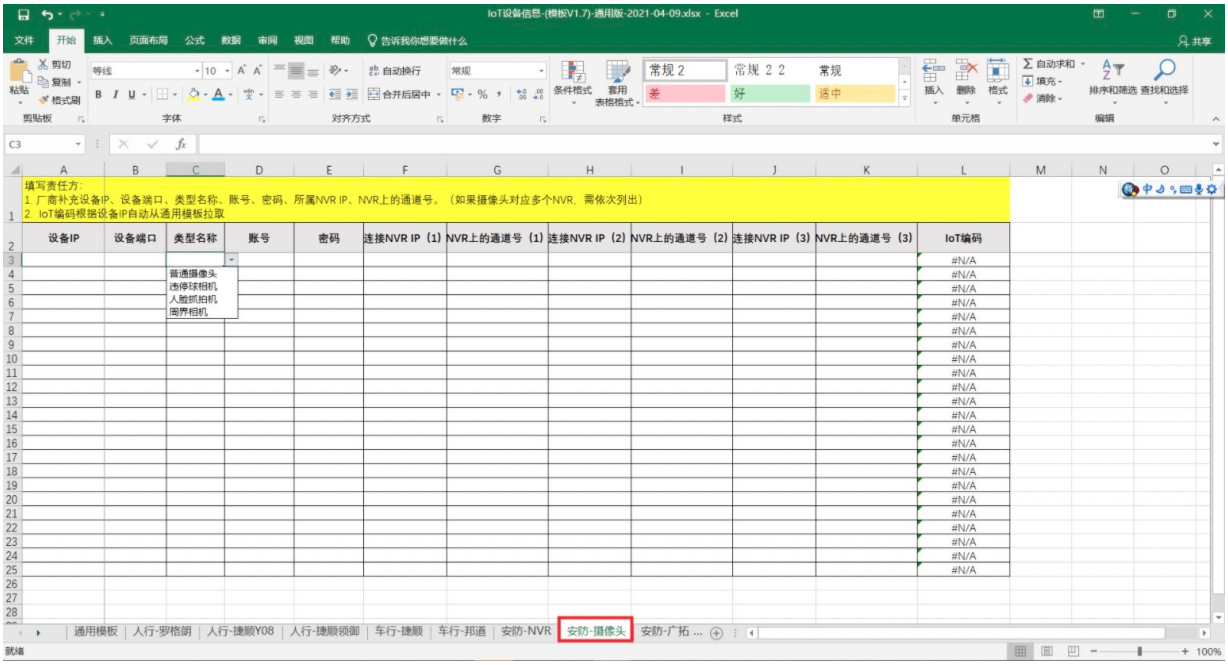
2.2.1 创建摄像头产品

说明：如果一个账号下已经创建了下面列出的摄像头产品，请忽略此步骤。请不要在同一个账号下重复创建下面列出摄像头产品。在阿里云主账号或者下面的一个RAM子账号创建的产品，该主账号和所有RAM子账号都可以看见该产品

点击“设备管理 - 产品 - 创建产品”，如下图所示。

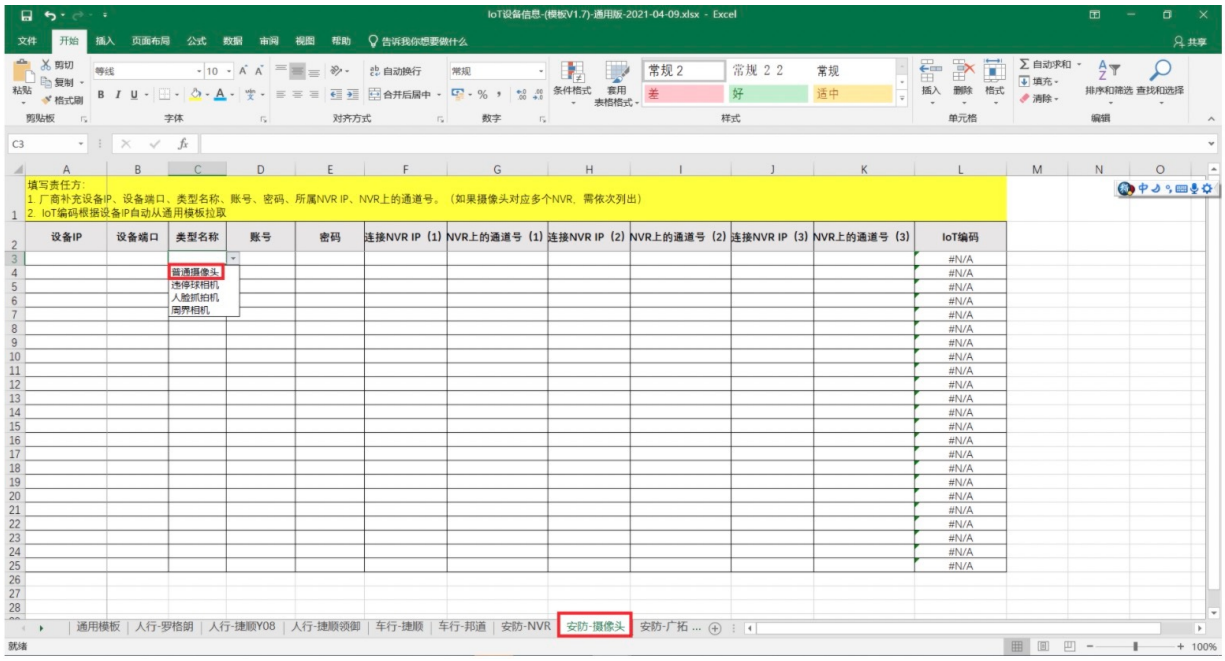


可以创建的摄像头产品类型共有四种：普通摄像头、违停球相机、人脸抓拍机、人脸抓拍机、周界相机。请参照甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页，根据设备支持的**类型名称里**支持的种类分别创建不同产品，如下图所示。如果同时有下面4种设备类型，则需要创建4种类型的产品，后面分别介绍



2.2.1.1 创建普通摄像头产品

产品名称：输入规则，请参照甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页，根据设备支持的**类型名称**填写。如果这个表格里设备的支持类型名称有**普通摄像头**类型，则产品名称输入**普通摄像头**，否则不需要创建**普通摄像头**产品，可以跳过此节。



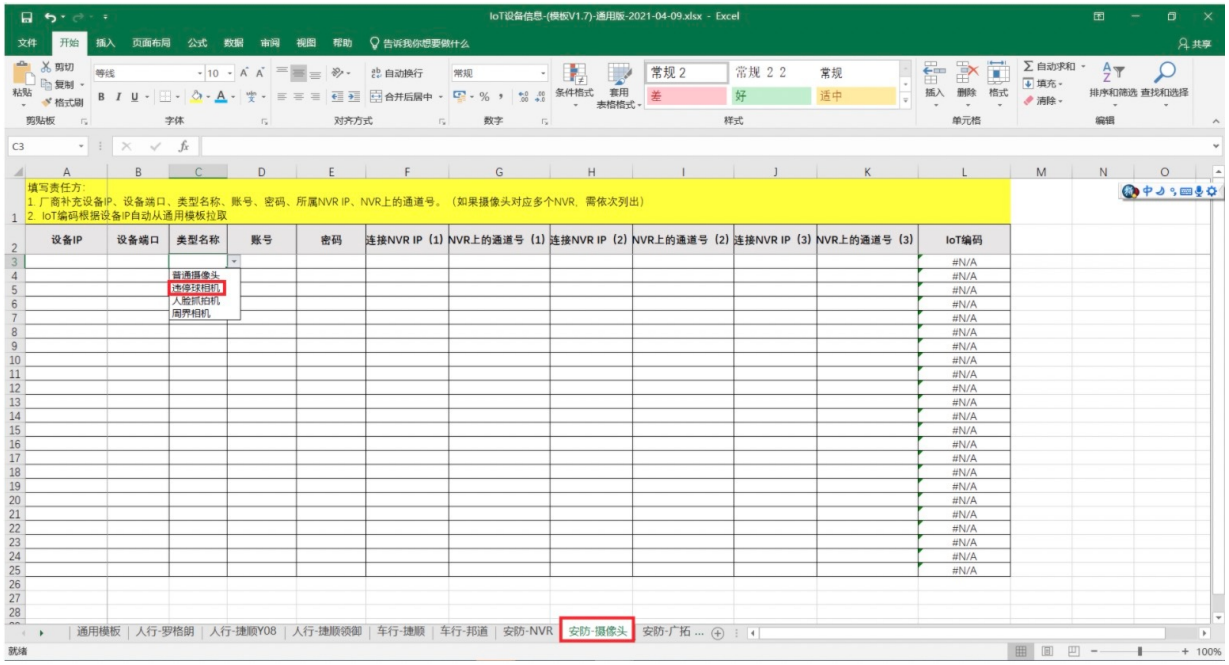
所属品类：请选择“标准品类：智能生活/家居安防/摄像头”

节点类型：请选择“直连设备”，如下图所示。



2.2.1.2 创建违停球相机产品

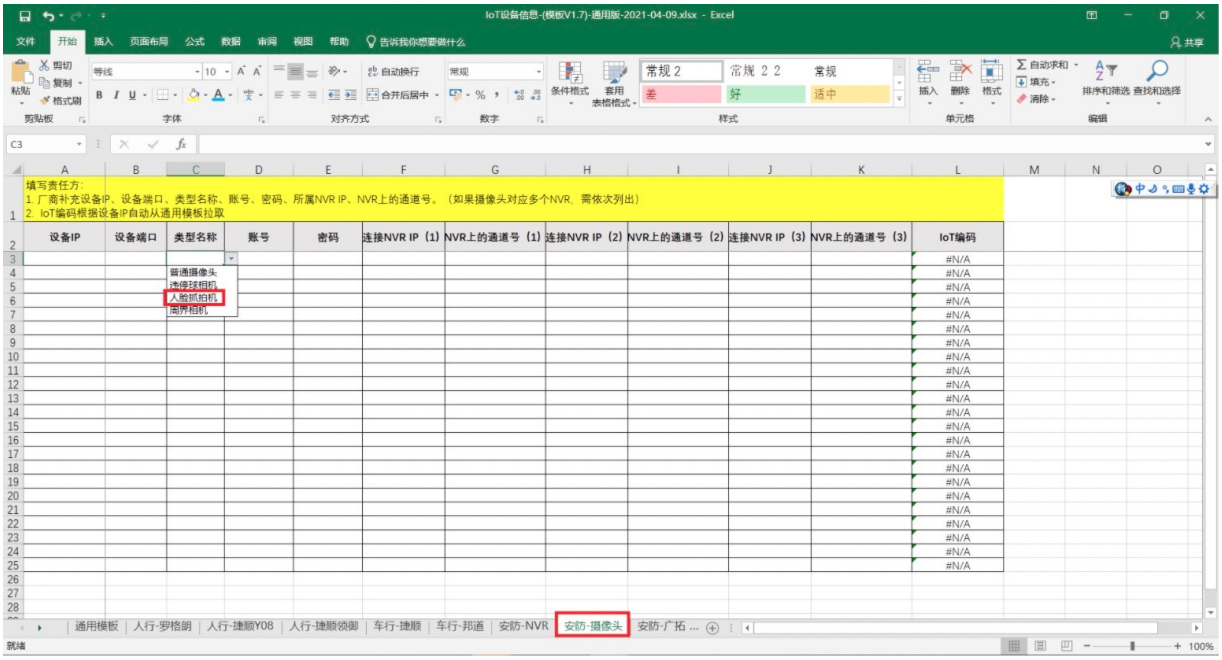
产品名称：输入规则，请参照甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页，根据设备支持的**类型名称**填写。如果这个表格里设备的支持类型名称有**违停球相机**类型，如下图所示，则产品名称输入**违停球相机**，否则不需要创建**违停球相机**产品，可以跳过此节。剩下操作与2.2.1.1完全相同





2.2.1.3 创建人脸抓拍机产品

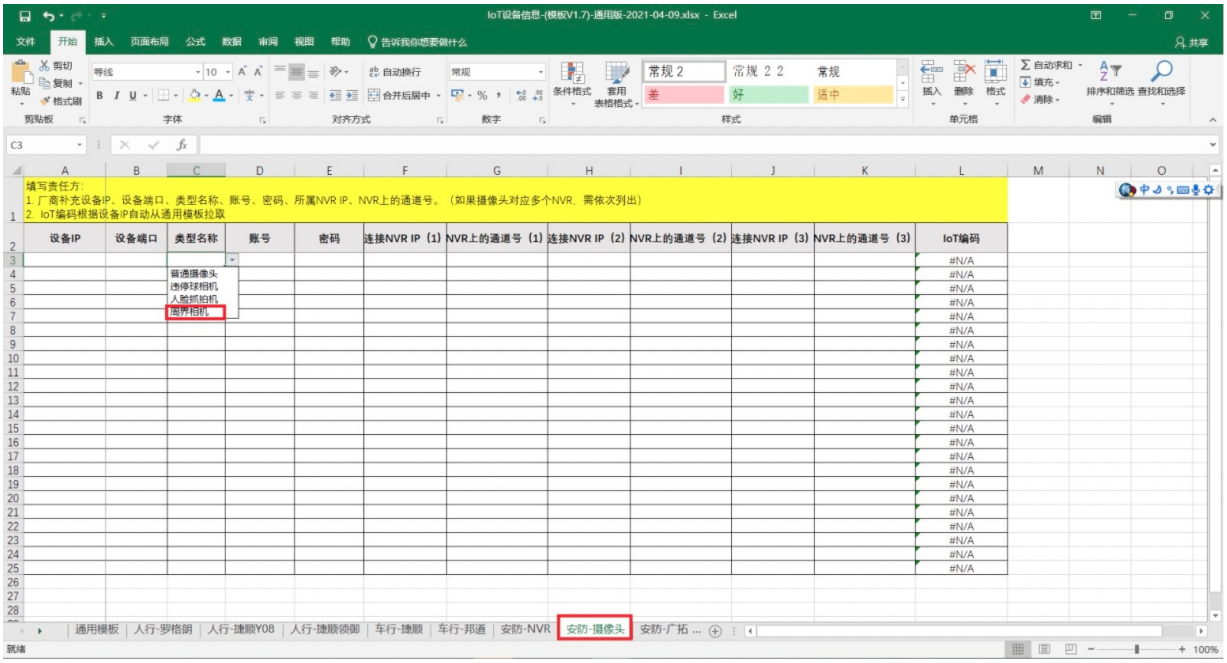
产品名称：输入规则，请参照甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页，根据设备支持的**类型名称**填写。如果这个表格里设备的支持类型名称有**人脸抓拍机**类型，则产品名称输入**人脸抓拍机**，否则不需要创建**人脸抓拍机**产品，可以跳过此节。剩下操作与2.2.1.1完全相同





2.2.1.4 创建周界相机产品

产品名称：输入规则，请参照甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页，根据设备支持的**类型名称**填写。如果这个表格里设备的支持类型名称有**周界相机**类型，则产品名称输入**周界相机**，否则不需要创建**周界相机**产品，可以跳过此节。剩下操作与2.2.1.1完全相同





2.2.2 无NVR接入摄像头和有NVR接入摄像头类型

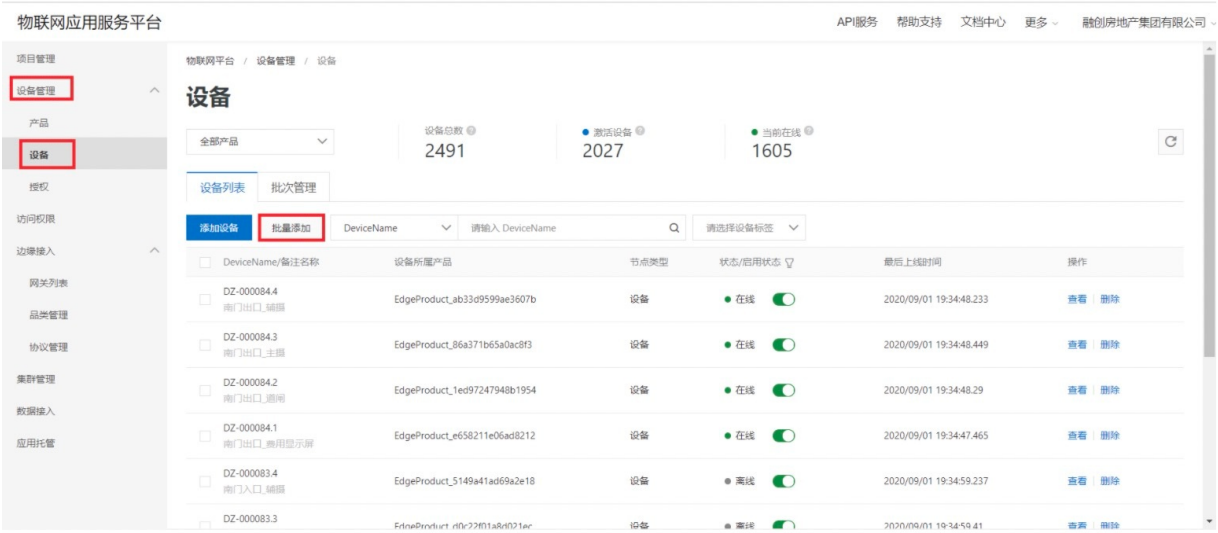
摄像头设备分两类，一类是直连LV边缘视频的摄像头设备，我们定义为**无NVR接入摄像头**。另外一类摄像头接入到NVR摄像头，然后NVR摄像头再连接LV边缘视频，我们定义为**有NVR接入摄像头**。

如何区分**有NVR接入摄像头**和**无NVR接入摄像头**：请参照甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页，如果下面红框里与NVR相关的列里有内容，则属于**有NVR接入摄像头**，否则属于**无NVR接入摄像头**

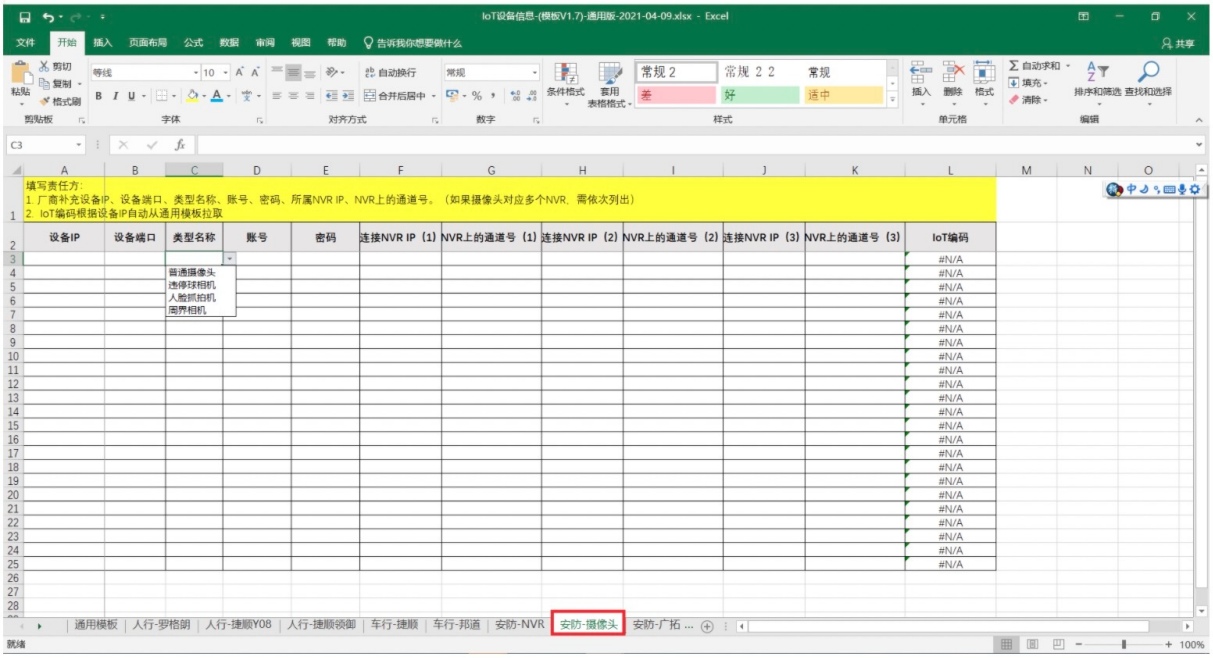


2.2.3 创建摄像头设备

点击“设备管理 - 设备 - 批量添加”，如下图所示。



在3.2.1里介绍过，可以创建的摄像头产品类型共有四种：普通摄像头、违停球相机、人脸抓拍机、人脸抓拍机、周界相机。下面将分别为这四类摄像头产品进行设备创建。请参照甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页，查找出某个摄像头类型下的所有设备，然后参照下面说明完成设备创建。如果同时有下面4种设备类型，则需要创建4种类型产品的设备，如下图所示

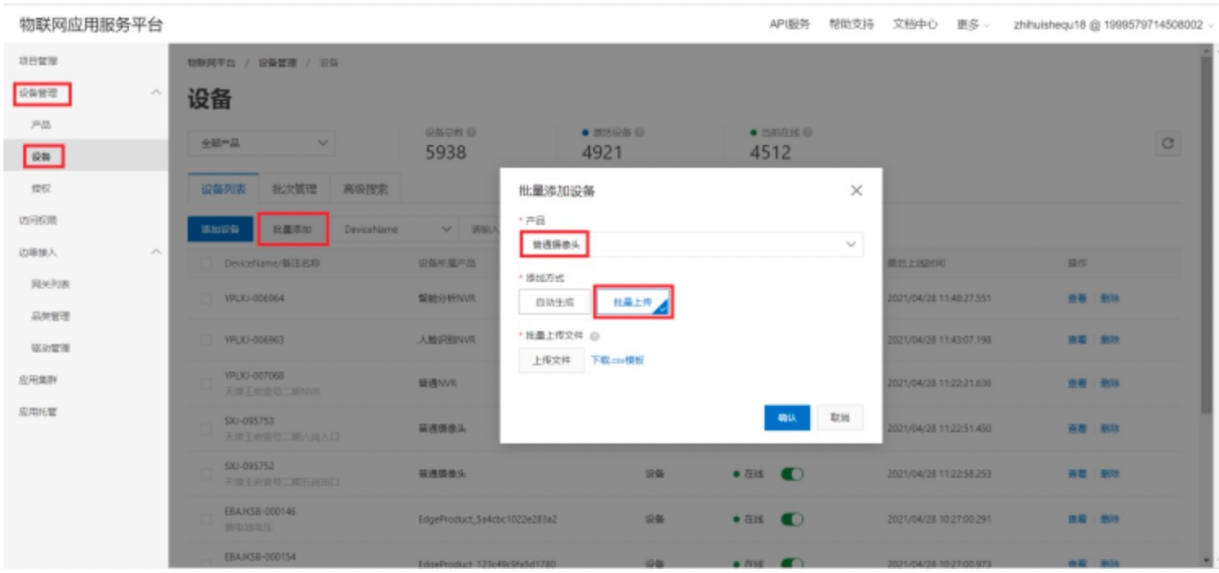


2.2.3.1 创建普通摄像头设备

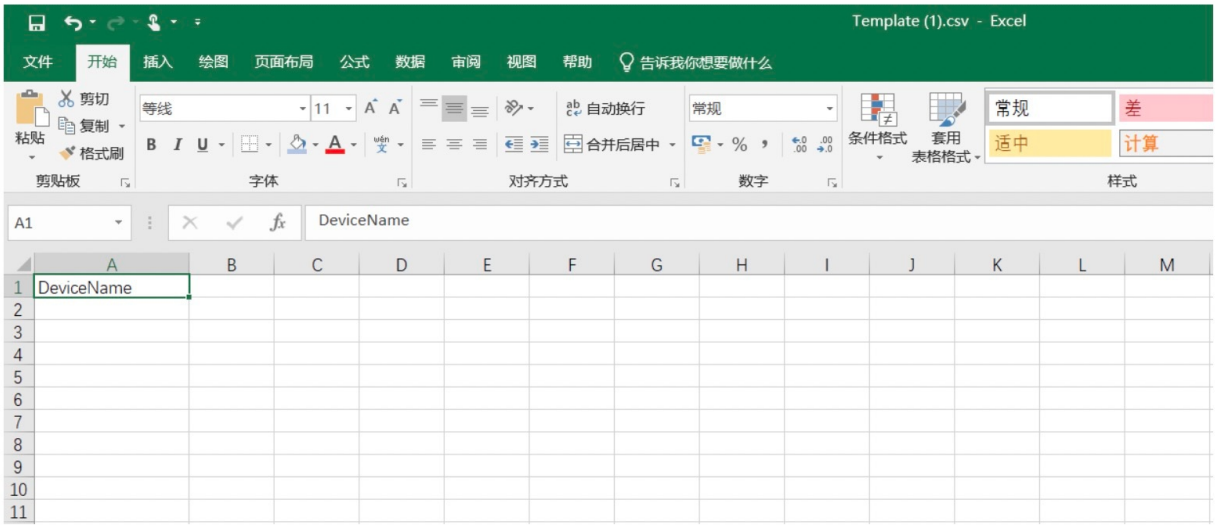
根据甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页，如果包含的设备类型有**普通摄像头**，则参照本节完成普通摄像头设备创建。

产品选择：选择之前在2.2.1.1 节中创建的**普通摄像头**产品。

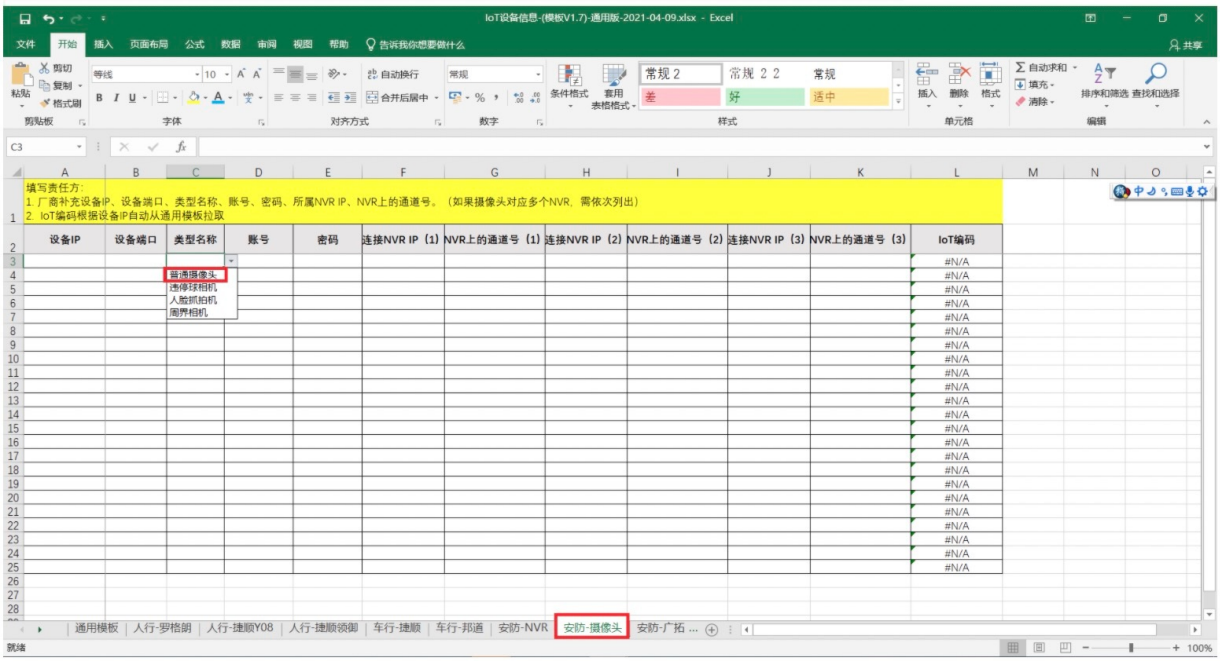
添加方式：选择**批量上传**，如下图所示



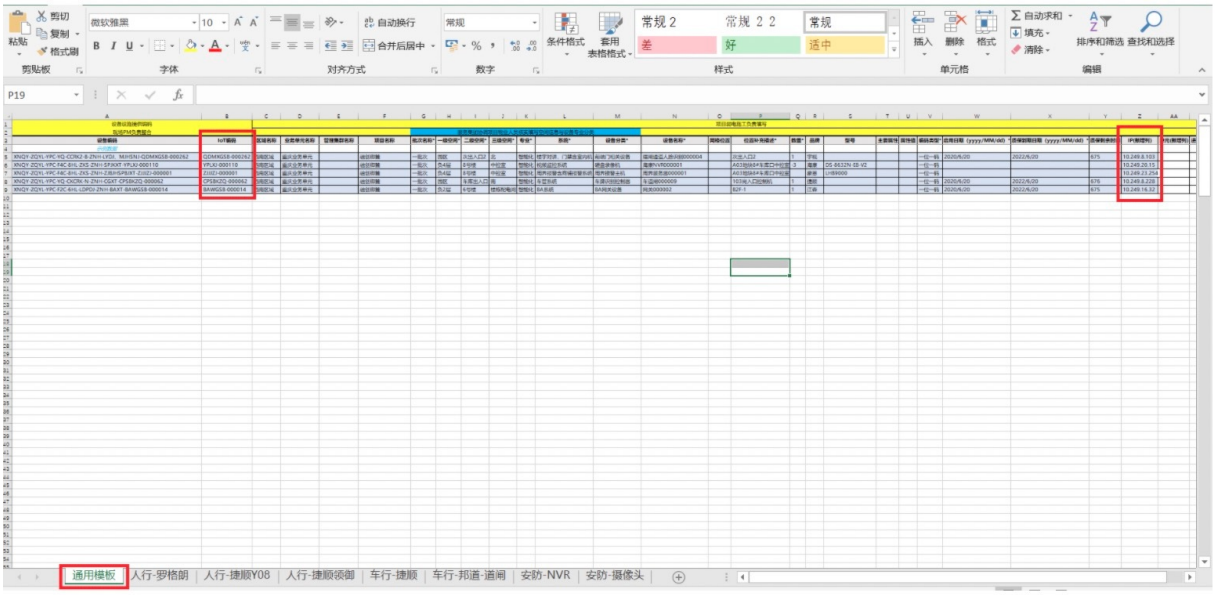
批量上传文件：首先点击"下载.csv模板"。模板内容如下图所示。这里的DeviceName, 需要填入类型为普通摄像头的设备的IoT编码。



无NVR接入摄像头的DeviceName：请参照甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页查看类型名称为普通摄像头的**设备IP**



然后在<IoT设备信息表>的**通用模板**TAB页，查找**设备IP**对应的**IoT编码**，然后将**IoT编码**填入上面的csv模板表格里DeviceName列里。



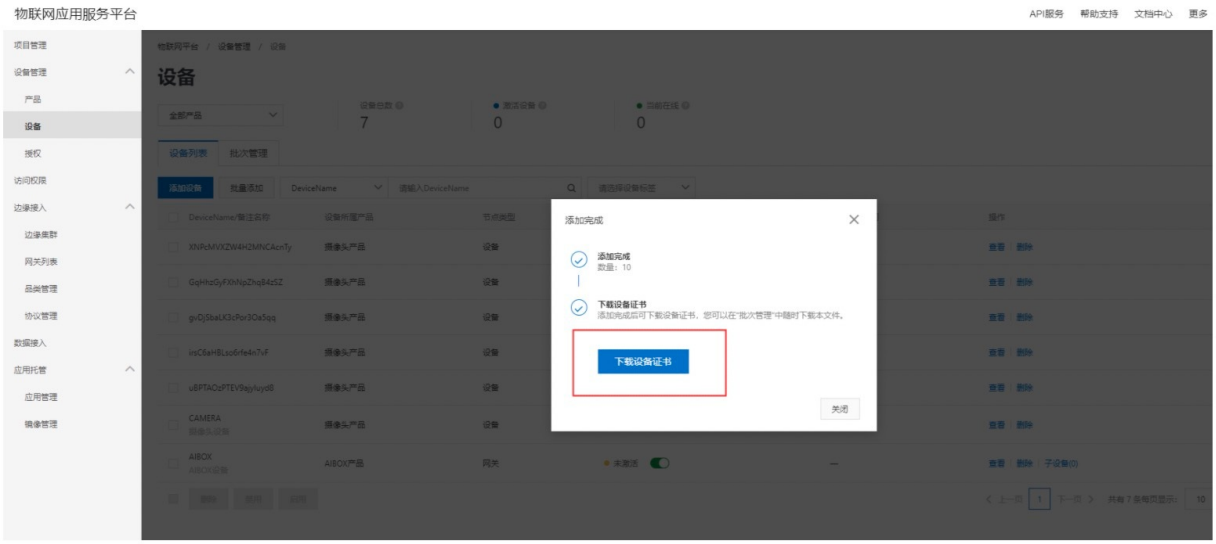
有NVR接入摄像头的DeviceName: 请参照甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页查看类型名称为普通摄像头的**设备IP**，然后在<IoT设备信息表>的**通用模板**TAB页，查找**设备IP**对应的**IoT编码**，例如CPSBKZQ-000062。由于一个**有NVR接入的摄像头**可能同时连接多个NVR设备，则需要在IoT平台上为该**有NVR接入摄像头**创建多个设备。接入第一个NVR对应的IoT编码是CPSBKZQ-000062.1，接入第二个NVR对应的IoT编码是CPSBKZQ-000062.2。后面以此类推。然后将所有的IoT编码填入上面的csv模板表格DeviceName列里。例如下面表格的三个连接NVR的列里都有数据，那需要创建三个IoT编码。



A	B	C	D	E	F	G	H	I	J	K	L
填写责任方: 厂商补充设备IP、设备端口、类型名称、账号、密码、所属NVR IP、NVR上的通道号。（如果摄像头对应多个NVR，需依次列出）											
设备IP	设备端口	类型名称	账号	密码	连接NVR IP（1）	NVR上的通道号（1）	连接NVR IP（2）	NVR上的通道号（2）	连接NVR IP（3）	NVR上的通道号（3）	

将所有**无NVR接入摄像头和有NVR接入摄像头**设备的**IoT编码**添入表格里后。然后点击"**上传文件**"并确认

添加完成后点击“下载设备证书”（下载好的设备证书 都以“.csv”结尾，例如“设备证书.csv”），如下图所示。



2.2.3.2 创建违停球相机设备

根据甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页，如果包含的设备类型有违停球相机，则需完成违停球相机设备创建。

产品选择：选择之前在2.2.1.2 节中创建的违停球相机产品。如下图所示：



后续操作与上节2.2.3.1类似。需要注意的是，批量创建操作时，在上传的csv模板里，输入的DeviceName, 注意一定是类型为违停球相机的所有设备的IoT编码。

2.2.3.3 创建人脸抓拍机设备

根据甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页，如果包含的设备类型有人脸抓拍机，则需完成人脸抓拍机设备创建。

产品选择：选择之前在2.2.1.3 节中创建的人脸抓拍机产品。如下图所示：



后续操作与2.2.3.1节类似。需要注意的是，批量创建操作时，在上传的csv模板里，输入的DeviceName, 注意一定是类型为人脸抓拍机的所有设备的IoT编码。

2.2.3.4 创建周界相机设备

根据甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页，如果包含的设备类型有周界相机，则需完成周界相机设备创建。

产品选择：选择之前在2.2.1.4 节中创建的周界相机产品，如下图所示



后续操作与2.2.3.1类似。需要注意的是，批量创建操作时，在上传的csv模板里，输入的DeviceName, 注意一定是类型为周界相机的所有设备的IoT编码

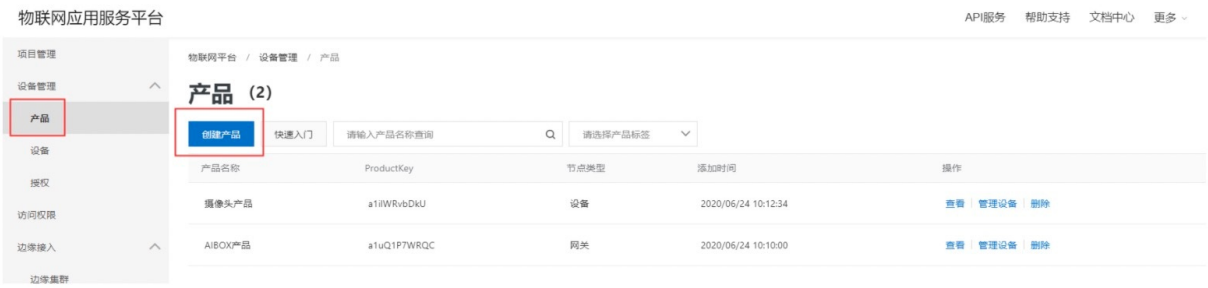
2.3 创建NVR类型的产品和设备

当项目中存在摄像头设备接入到NVR设备中实现上云的情况，需要添加相关的NVR设备。操作示例如下：

2.3.1 创建NVR产品

说明：如果一个账号下已经创建了下面列出的NVR产品，请忽略此步骤。请不要在同一个账号下重复创建下面列出的NVR产品。在阿里云主账号或者下面的一个RAM子账号创建的产品，该主账号和所有RAM子账号都可以看见该产品

点击“设备管理 - 产品 - 创建产品”，如下图所示。

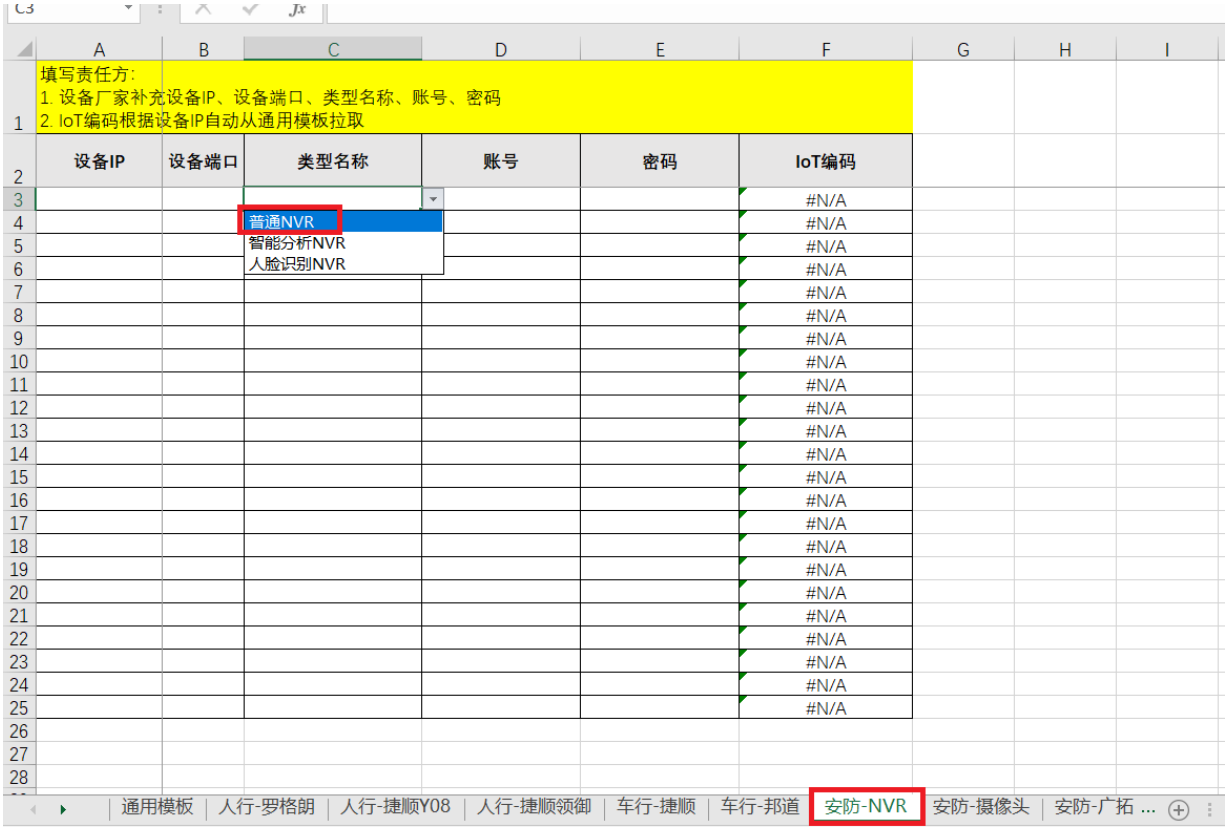


可以创建的NVR产品类型共有三种：普通NVR、智能分析NVR，人脸识别NVR。请参照甲方提供的<IoT设备信息表>，在**安防-NVR**TAB页，根据NVR设备的**类型名称里**支持的种类分别创建不同产品，如下图所示。如果同时有下面3种NVR设备类型，则需要创建3种类型的产品，后面分别介绍



2.3.1.1 创建普通NVR产品

产品名称：输入规则，请参照甲方提供的<IoT设备信息表>，在**安防-NVR**TAB页，根据设备支持的**类型名称**填写。如果这个表格里设备的支持类型名称有**普通NVR**类型，则产品名称输入**普通NVR**，否则不需要创建**普通NVR**产品，可以跳过此节。



所属品类：请选择“标准品类：智能生活/家居安防/网络硬盘录像机”

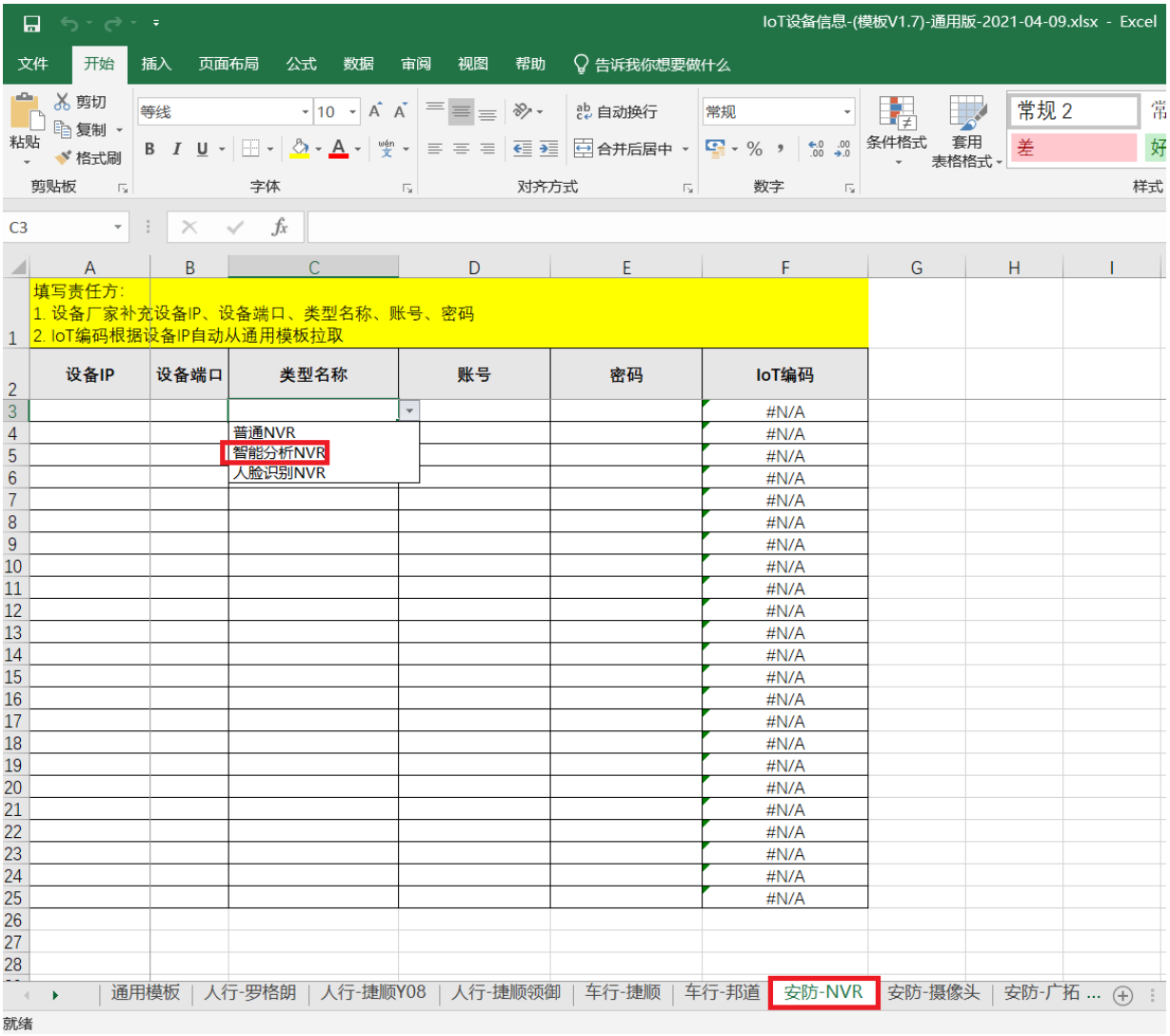
节点类型：请选择“直连设备”，如下图所示。



点击**确认**保存

2.3.1.2 创建智能分析NVR产品

产品名称：输入规则，请参照甲方提供的<IoT设备信息表>，在**安防-NVR**TAB页，根据设备支持的**类型名称**填写。如果这个表格里设备的支持类型名称有**智能分析NVR**类型，则产品名称输入**智能分析NVR**，否则不需要创建**智能分析NVR**产品，可以跳过此节。



所属品类：请选择“标准品类：智能生活/家居安防/网络硬盘录像机”

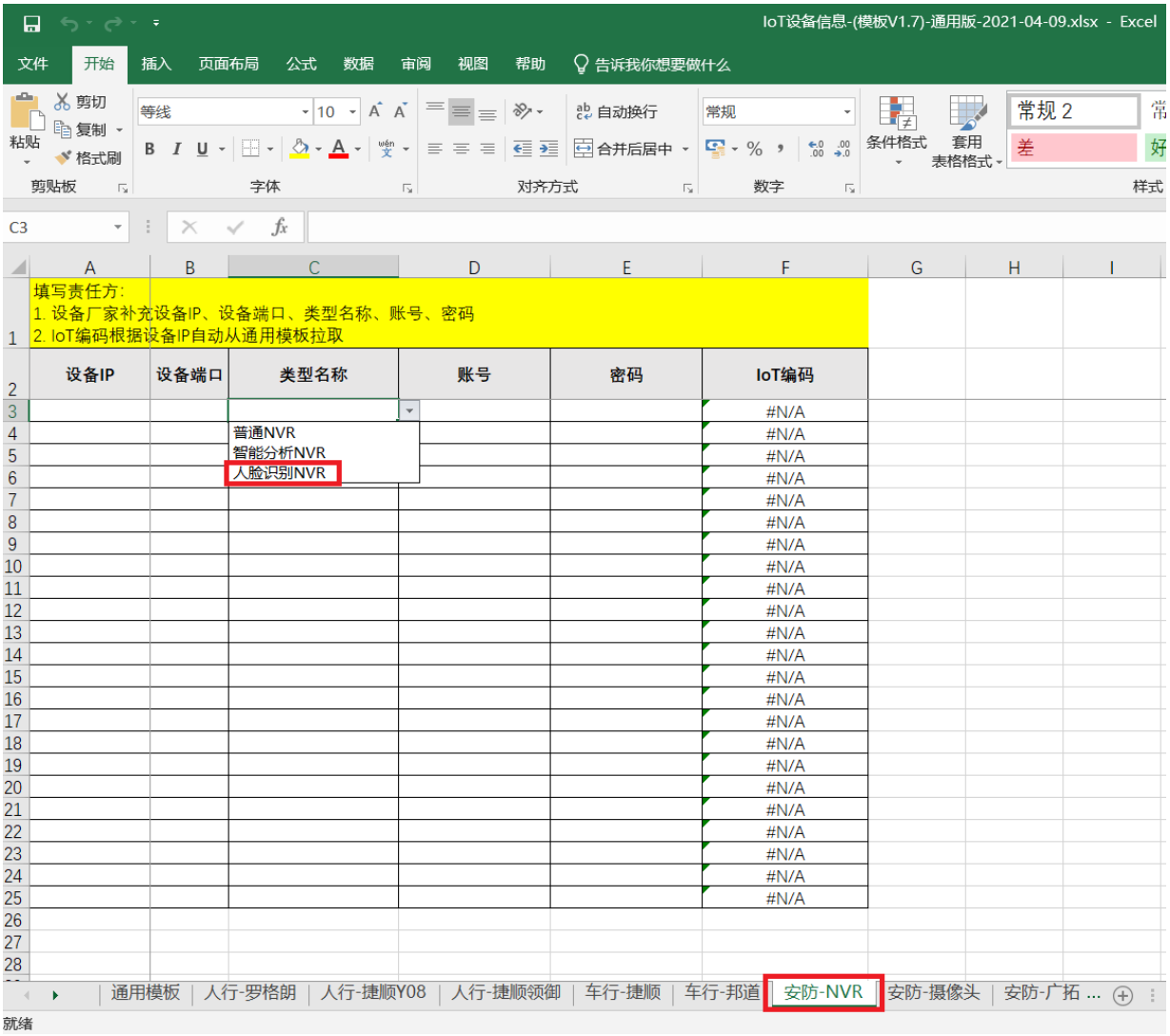
节点类型：请选择“直连设备”，如下图所示。



点击**确认**保存

2.3.1.3 创建人脸识别NVR产品

产品名称：输入规则，请参照甲方提供的<IoT设备信息表>，在**安防-NVR**TAB页，根据设备支持的**类型名称**填写。如果这个表格里设备的支持类型名称有**人脸识别NVR**类型，则产品名称输入**人脸识别NVR**，否则不需要创建**人脸识别NVR**产品，可以跳过此节。



所属品类：请选择“标准品类：智能生活/家居安防/网络硬盘录像机”

节点类型：请选择“直连设备”，如下图所示。



点击**确认**保存

2.3.2 创建NVR类型设备

当项目中存在摄像头设备接入到NVR设备然后实现上云的情况，需要添加相关的NVR设备。

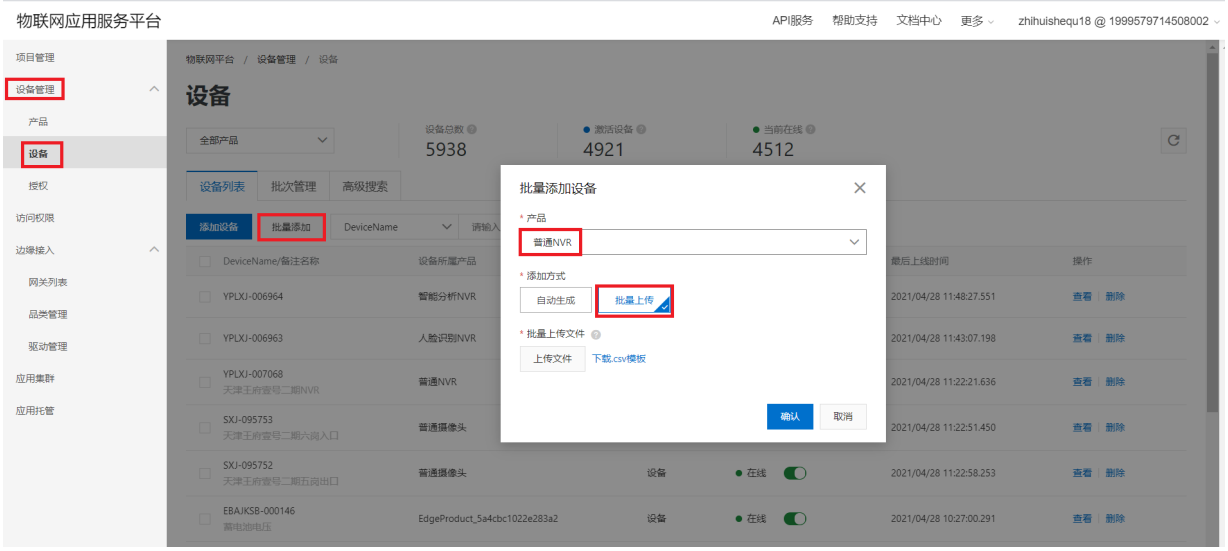
NVR设备分普通NVR, 智能分析NVR, 人脸识别NVR。请参照甲方提供的<IoT设备信息表>，在**安防-NVR**TAB页，可以区分出NVR设备类型，如下图所示：



需要分别为这三类NVR设备进行设备创建，请参照下面章节操作：

2.3.2.1 创建普通NVR设备

选择设备管理 - 设备，点击批量添加，如下图所示：



产品选择：选择**普通NVR**产品。

添加方式：选择批量上传

批量上传文件：首先点击"下载.csv模板"。模板内容与3.2.2节一样，需要填写NVR设备的DeviceName。请参照甲方提供的<IoT设备信息表>，在**安防-NVR**TAB页查看**设备IP**，然后在<IoT设备信息表>的**通用模板**TAB页，查找所有设备类型为普通NVR的**设备IP**对应的**IoT编码**，并直接填入上面的csv模板表格里。

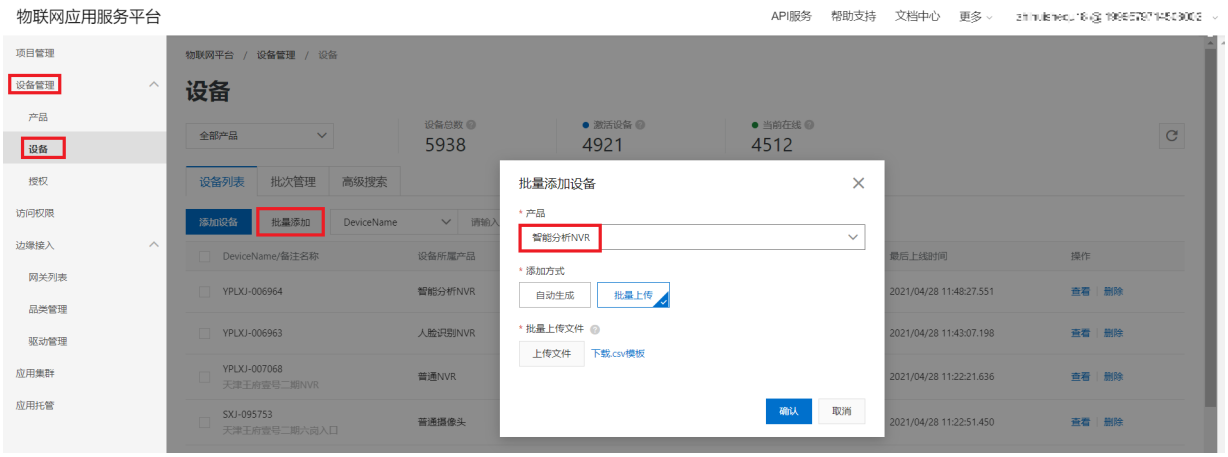


当所有普通NVR设备的**IoT编码**添入csv表格里后，点击"**上传文件**"并确认。

添加完成后点击“下载设备证书”并保存（下载好的设备证书 都以“.csv”结尾，例如“设备证书.csv”），后续操作将使用。

2.3.2.2 创建智能分析NVR设备

选择设备管理 - 设备，点击批量添加，如下图所示：

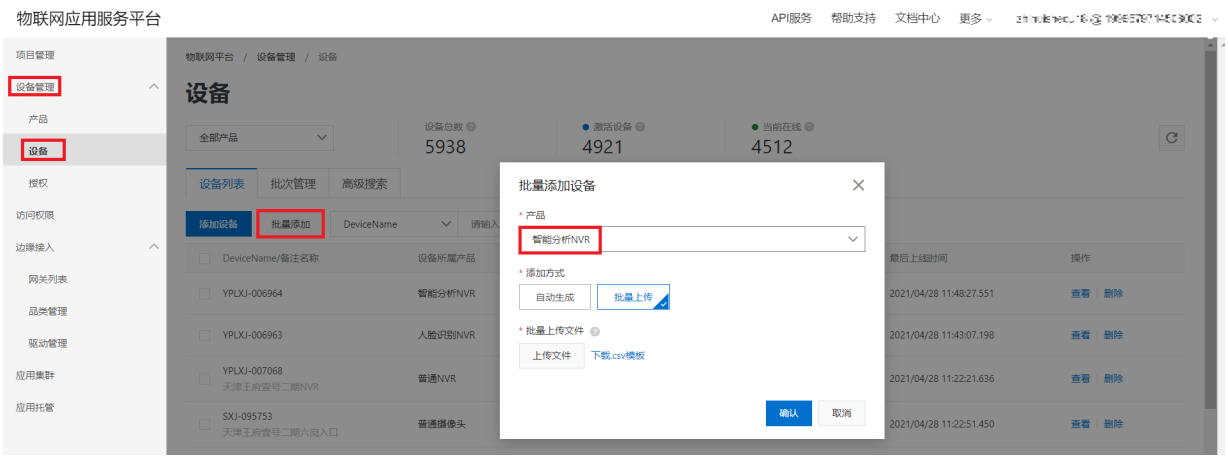


产品选择：选择**智能分析NVR**产品。

后续操作，与2.3.2.1类似

2.3.2.3 创建人脸识别NVR设备

选择设备管理 - 设备，点击批量添加，如下图所示：



产品选择：选择**人脸识别NVR产品**。

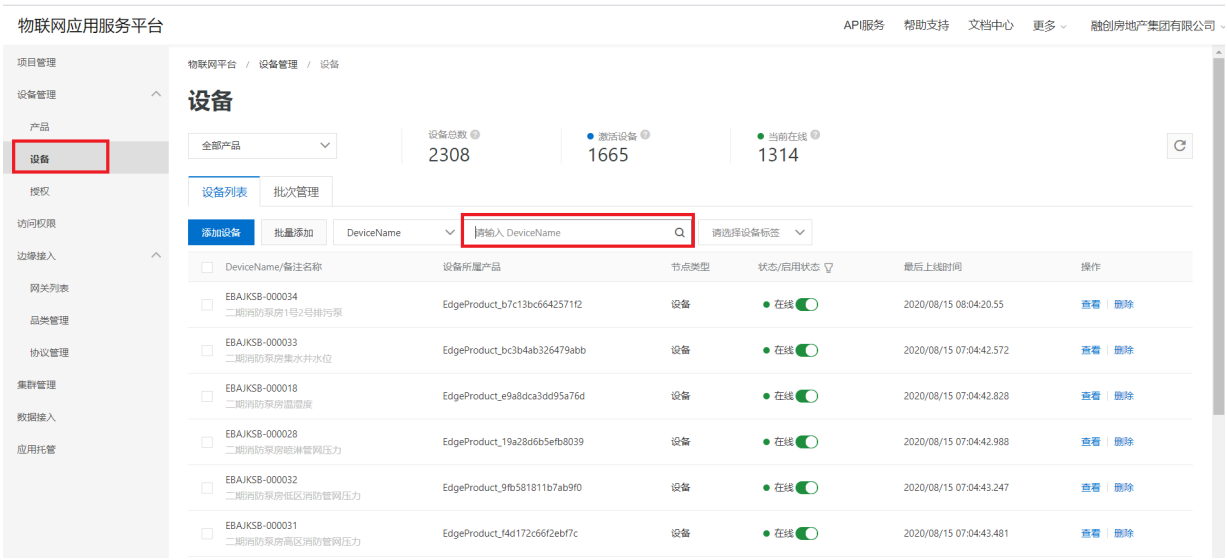
后续操作，与2.3.2.1类似

2.4 添加设备备注信息

安防设备，需要添加设备位置的备注信息，具体方式如下

2.4.1 查找设备

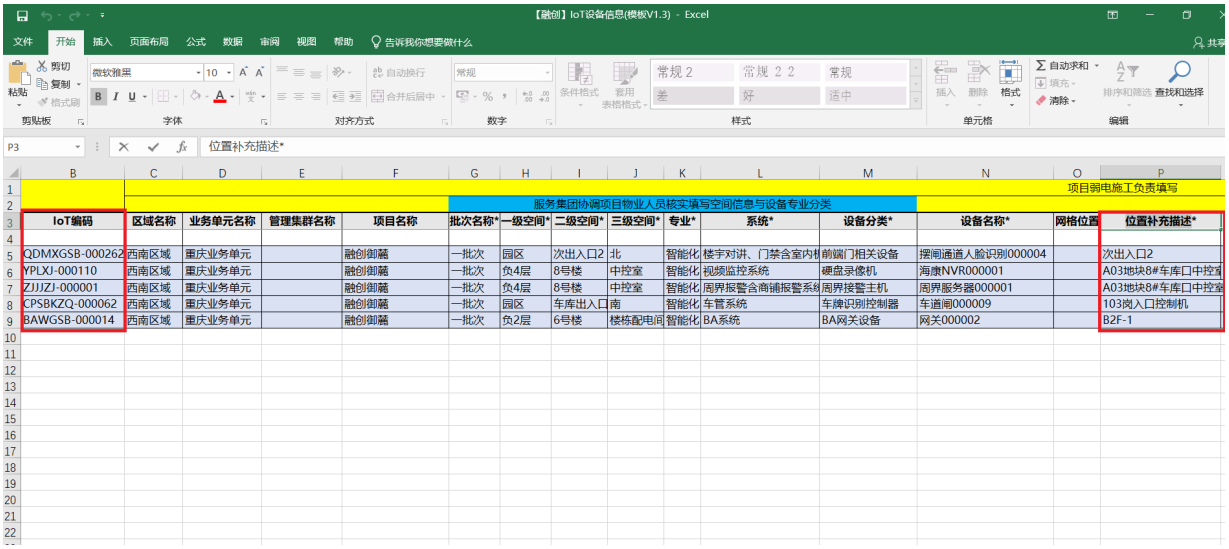
打开物联网应用服务平台。注意不要选择项目，点击设备，如下图所示：



然后将需要填写设备备注信息的设备的IoT编码，添加到上面的DeviceName搜索框里，点击搜索。

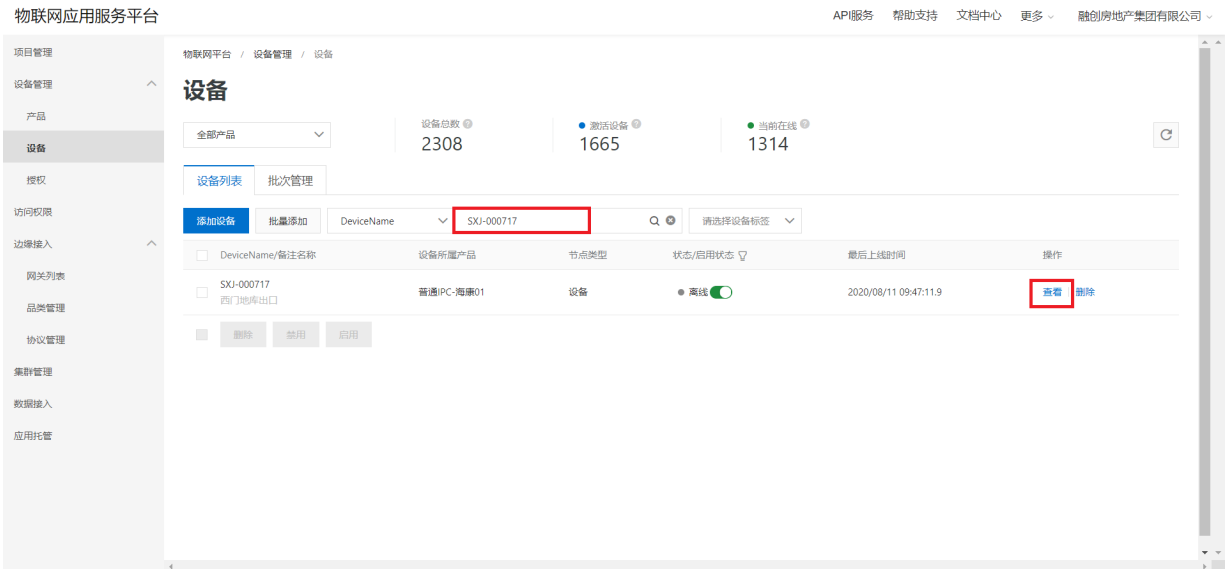
设备的IoT编码查找方法：

参照甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页查看某个**设备IP**，然后在<IoT设备信息表>的**通用模板**TAB页，查找**设备IP**对应的**IoT编码 和位置补充描述**。如下图所示：



IoT编码	区域名称	业务单元名称	管理集群名称	项目名称	批次名称*	一级空间*	二级空间*	三级空间*	专业*	系统*	设备分类*	设备名称*	网格位置	位置补充描述*
QDMXGSB-000262	西南区域	重庆业务单元		融创御藸	一批次	园区	次出入口2	北	智能化	楼宇对讲、门禁含室内机	前端门相关设备	擦闸通道人脸识别000004		次出入口2
YPLXJ-000110	西南区域	重庆业务单元		融创御藸	一批次	负4层	8号楼	中控室	智能化	视频监控系统	硬盘录像机	海康NVR000001		A03地块8#车库口中控室
ZJJJZJ-000001	西南区域	重庆业务单元		融创御藸	一批次	负4层	8号楼	中控室	智能化	周界报警含商铺报警系统	周界接警主机	周界服务器000001		A03地块8#车库口中控室
CPSBKZQ-000062	西南区域	重庆业务单元		融创御藸	一批次	园区	车库出入口	南	智能化	车管系统	车牌识别控制器	车道闸000009		103岗入口控制机
BAWGSB-000014	西南区域	重庆业务单元		融创御藸	一批次	负2层	6号楼	楼栋配电间	智能化	BA系统	BA网关设备	网关000002		B2F-1

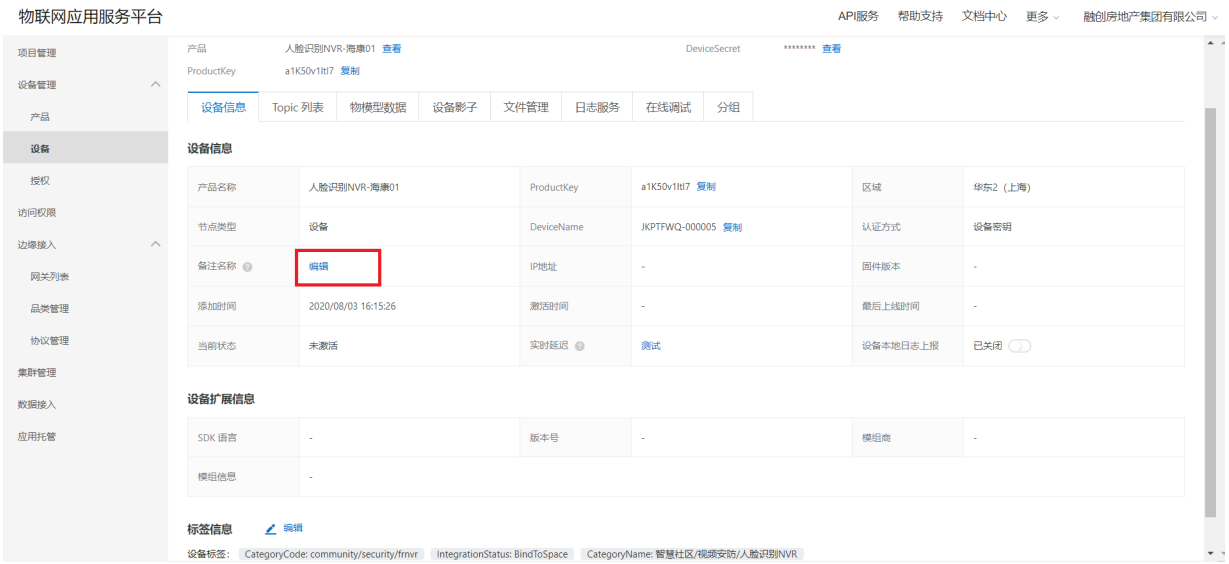
将上面找到的设备的IoT编码，输入到上面打开的物联网应用服务平台的DeviceName搜索框里，点击搜索。结果如如下图所示：



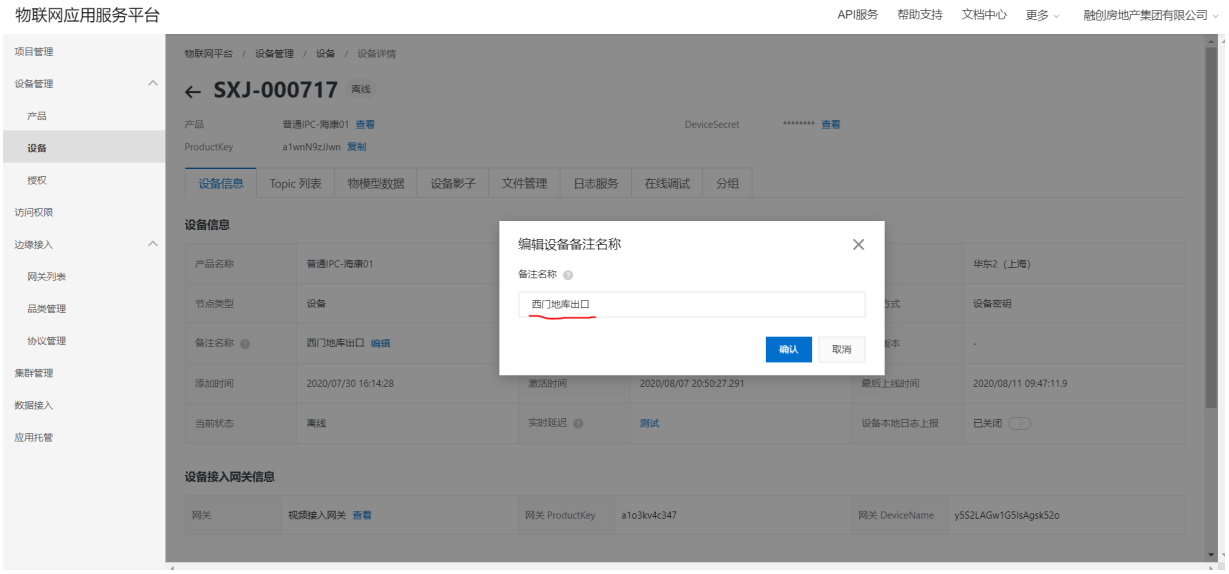
找到搜索到的设备，点击右侧**查看**

2.4.2 添加备注信息

点击**编辑**：



将上一节里找到的**位置补充描述**信息，添加到弹出的编辑框里，点击**确认**。如下图所示



重复上面操作，将所有安防设备的备注名完成添加。

3 添加设备到项目中

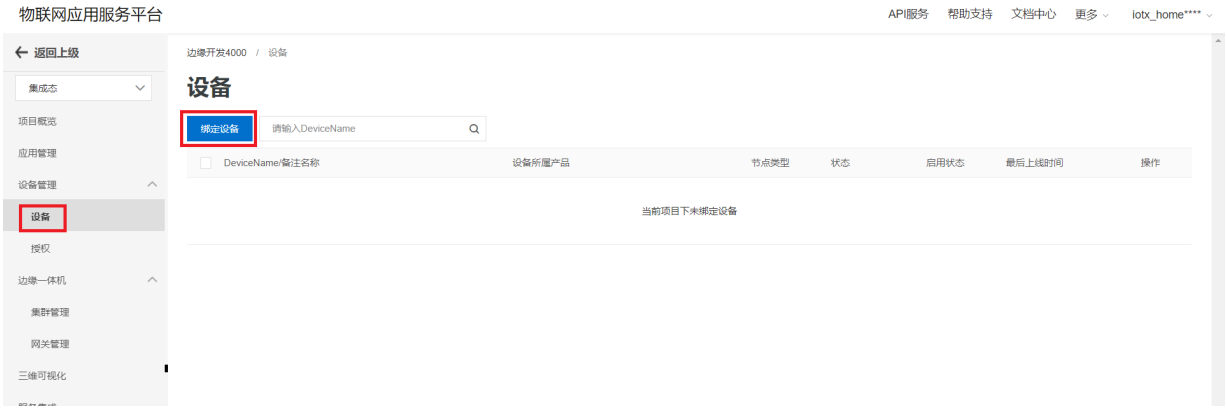
由于LV边缘视频的网关设备和摄像头设备都是在SI工作台直接创建的，没有依赖集群中LinkIoTEdge生产的网关。所以需要将设备添加到项目中。

说明：请特别注意，为指定项目创建的设备，必须绑定到指定项目下。

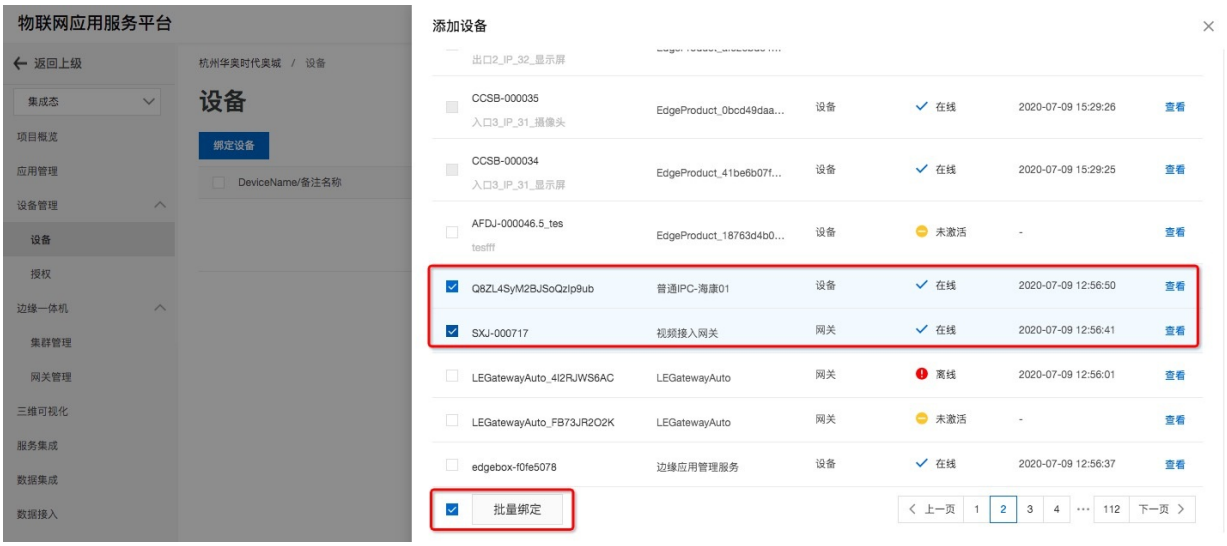
登录物联网应用服务平台，找到对应的项目，点击进入项目详情中。如下图所示。



点击**设备-绑定设备**。



选择刚才创建的LV边缘视频网关设备和摄像头设备。实现设备绑定到项目中。如下图所示。



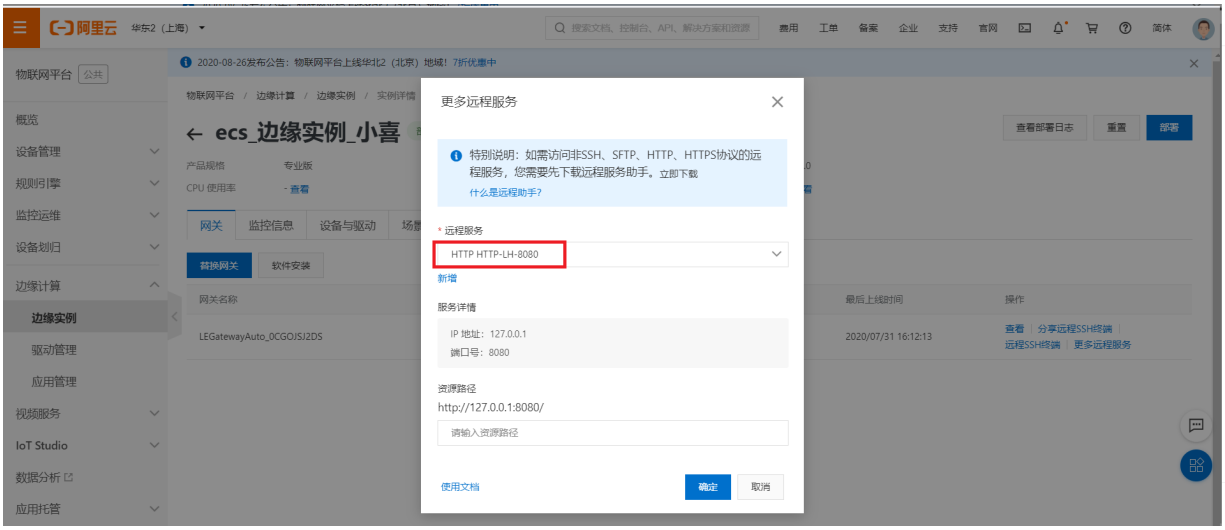
绑定成功页面如下图所示。



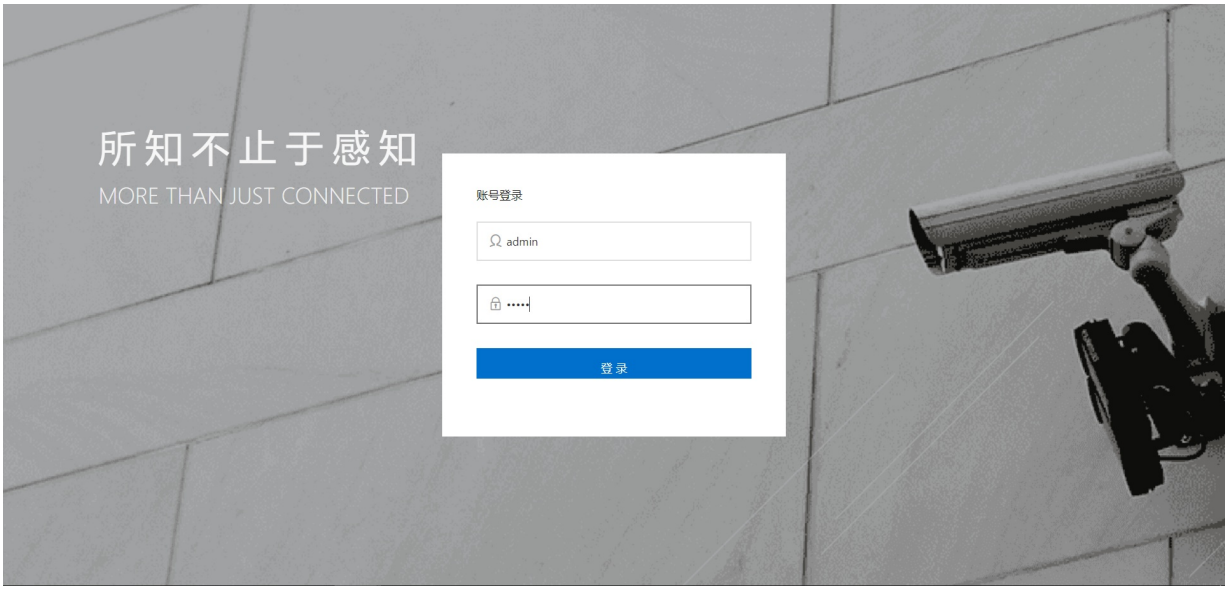
4.LV边缘视频本地控制台设备操作

4.1 登录LV边缘实例本地控制台

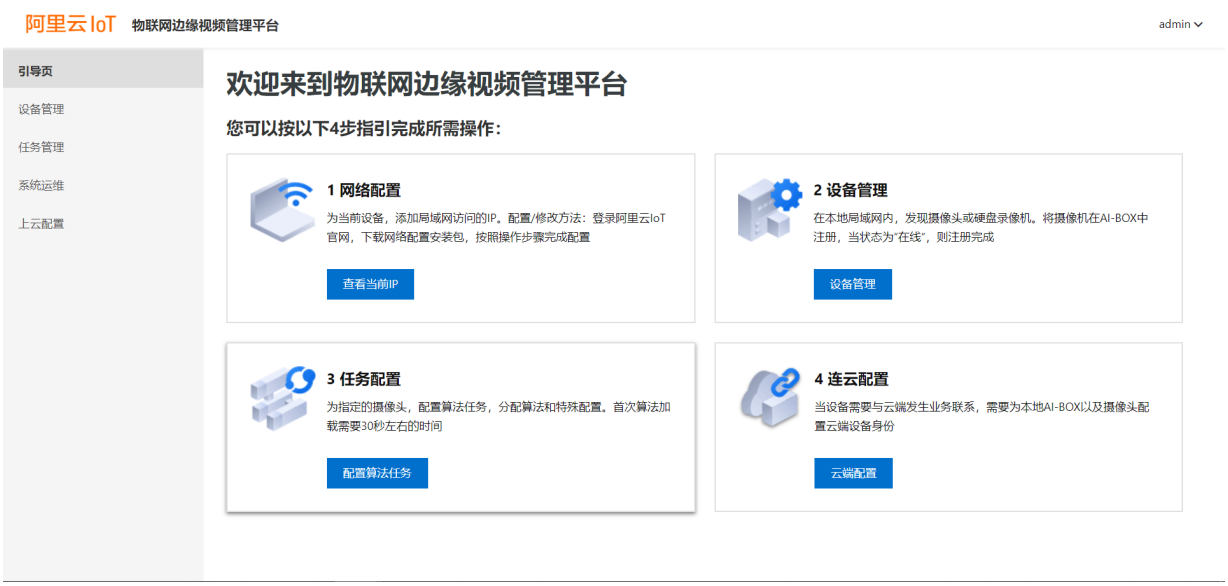
点击**更多远程服务**，远程服务选择“**HTTPHTTP-LH-8080**”，点击确定，如图所示



弹出的页面，输入账号密码，登录LV边缘视频的本地控制台。登录账号：admin；密码 admin 。



登录后可进行设备的相关配置操作，如下图所示。

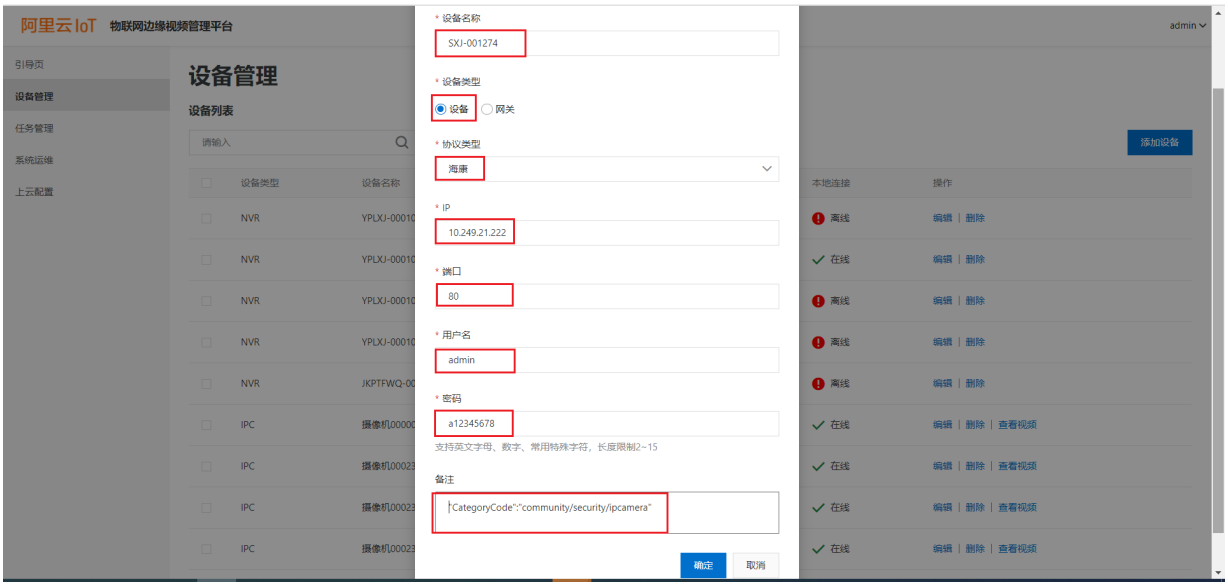


4.2 添加设备

4.2.1 添加无NVR接入摄像头

请特别注意，本地控制台只需要添加无NVR接入摄像头设备和NVR设备。通过NVR接入的摄像头设备不需要添加。当NVR设备添加成功后，通过NVR接入的摄像头设备将会自动拉取出来。

需要将IPC设备加入到LV边缘视频中，点击“**设备管理-添加设备**”进行设备信息的相关配置，如下图所示。



设备名称:填写无NVR接入设备的**IoT编码**。与3.2.3节类似，请参照甲方提供的<IoT设备信息表>，在**安防-摄像头**TAB页先查找到无NVR接入摄像头设备，查找方法参见前面的3.2.2节介绍。 找到无NVR接入的**设备IP**，然后在<IoT设备信息表>的**通用模板**TAB页，查找**设备IP**对应的**IoT编码**。输入示例为SXJ-001274

设备类型:直接接入IPC设备则选择“**设备**“即可。

协议类型：如果摄像头设备是海康的，协议选择海康；如果摄像头设备是大华的，协议选择大华；如果摄像头设备是宇视的，协议选择onvif；

IP：在上面填写**设备名称**时，已经查找到<IoT设备信息表>**安防-摄像头**TAB页 的**设备IP**。输入示例为10.249.20.23

端口，用户名，密码：参照甲方提供的<IoT设备信息表>，**安防-摄像头**TAB页的**设备端口**，**账号，密码，**如下图所示



备注信息：参照甲方提供的<IoT设备信息表>，根据**安防-摄像头**TAB页的**类型名称**填写：

普通摄像头：类型编码是**ipcamera**，备注信息内容：**"CategoryCode":"community/security/ipcamera"**

违停球相机：类型编码是**tptzcamera**，备注信息内容：**"CategoryCode":"community/security/tptzcamera"**

人脸抓拍机：类型编码是**frcamera**，备注信息内容：**"CategoryCode":"community/security/frcamera"**

周界相机：类型编码是**pmcamera**，备注信息内容：**"CategoryCode":"community/security/pmcamera"**



4.2.2 添加NVR设备

在LV边缘视频本地控制台，当添加NVR设备之前，需要确认两点：

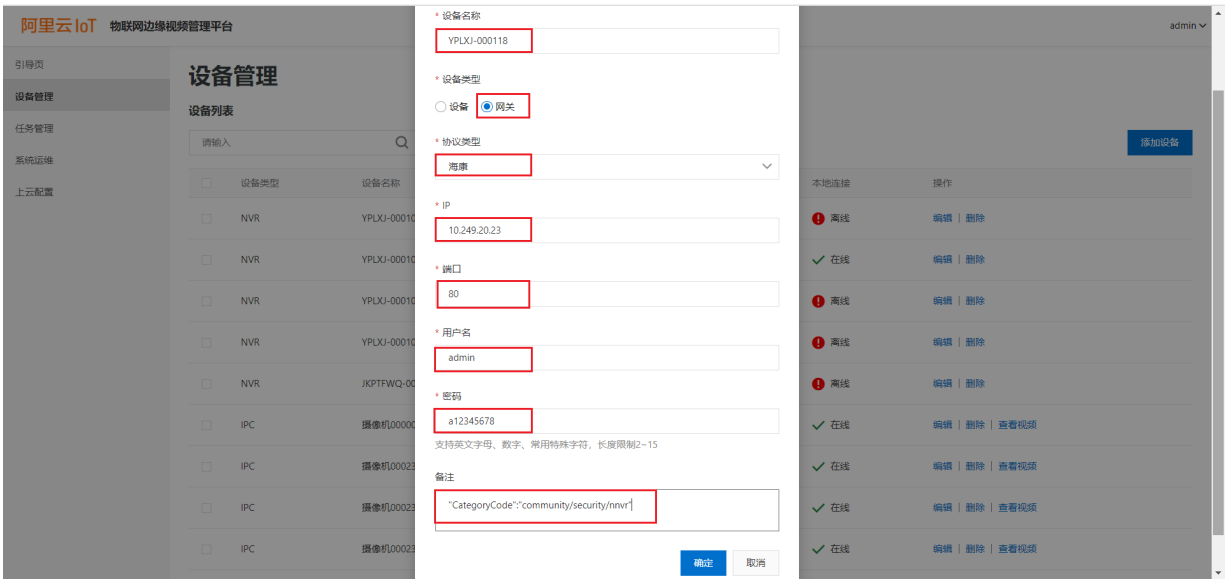
a: 要确保NVR设备内网正常。因为在添加NVR摄像头时，NVR设备下面连接的摄像头，将自动拉取添加。如果NVR设备不在线情况下完成添加，当NVR摄像头上线后，不会自动拉取。

b:要确保所有IPC正确配置到NVR。不然即使配上了NVR，以后也要删除NVR重新配置

如果发现NVR设备下面连接的摄像头没有自动拉取出来，需要删除NVR设备后再重新添加。详细请参考6.2节："NVR下的IPC摄像头无法自动拉取处理方法"

如果某个NVR以后再新添加了几个摄像头，也需要删除整个NVR，再全量添加

将NVR设备加入到LV边缘视频方法：点击“设备管理 - 添加设备”进行设备信息的相关配置，如下图所示。



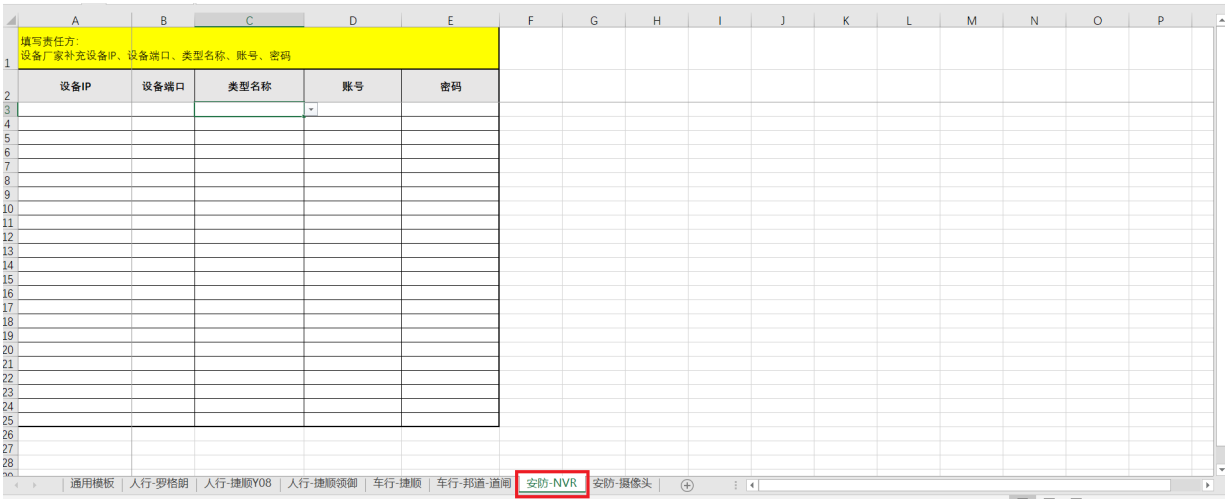
设备名称:填写设备的**IoT编码**。与3.2.3节类似，请参照甲方提供的<IoT设备信息表>，如下图。在**安防-NVR**TAB页查看**设备IP**,然后在<IoT设备信息表>的**通用模板**TAB页，查找**设备IP**对应的**IoT编码**。输入示例为YPLXJ-000118

设备类型:NVR类型的设备在添加的时候，设备类型需要选择**网关**。

协议类型：如果NVR设备是海康的，协议选择海康；如果NVR设备是大华的，协议选择大华；如果NVR设备是宇视的，协议选择GB;

IP：参照甲方提供的<设备信息表>中的**摄像头IP**。输入示例为10.249.20.23

端口，用户名，密码：参照甲方提供的<设备信息表>中的**账号密码和端口**

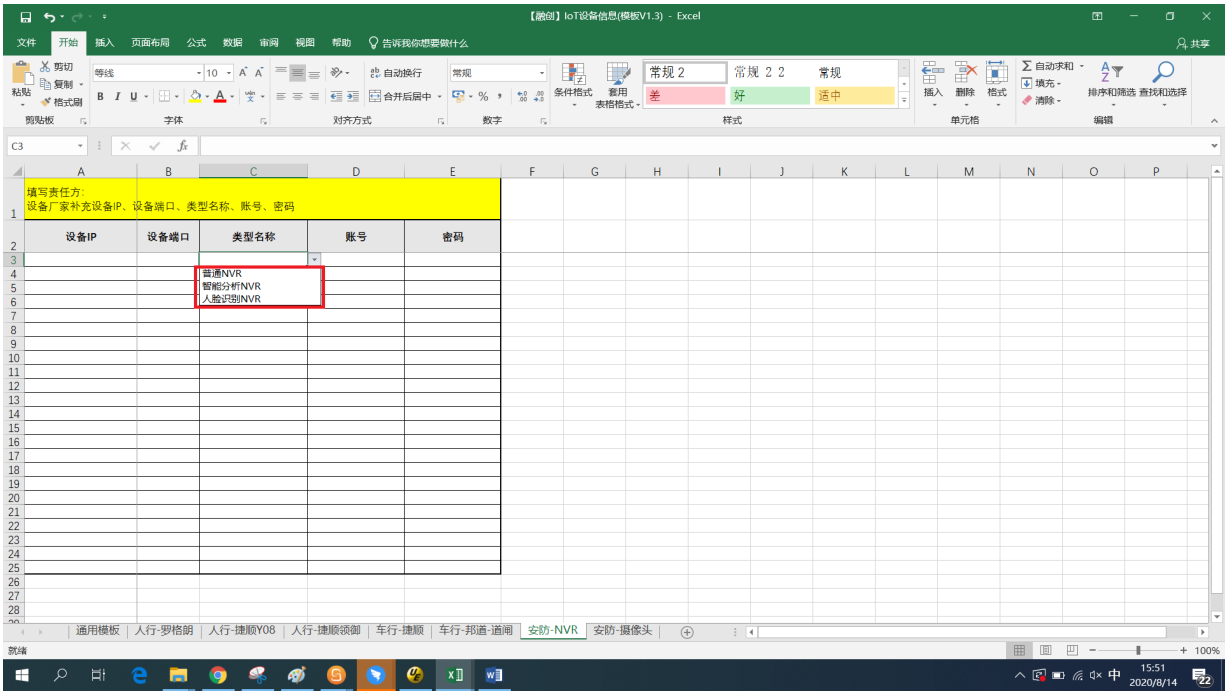


备注信息：参照甲方提供的<IoT设备信息表>，根据**安防-NVR**TAB页的**类型名称**填写：

普通NVR：类型编码是**nnvr**,备注信息内容：**"CategoryCode":"community/security/nnvr"**

智能分析NVR：类型编码是**invr**,备注信息内容：**"CategoryCode":"community/security/invr"**

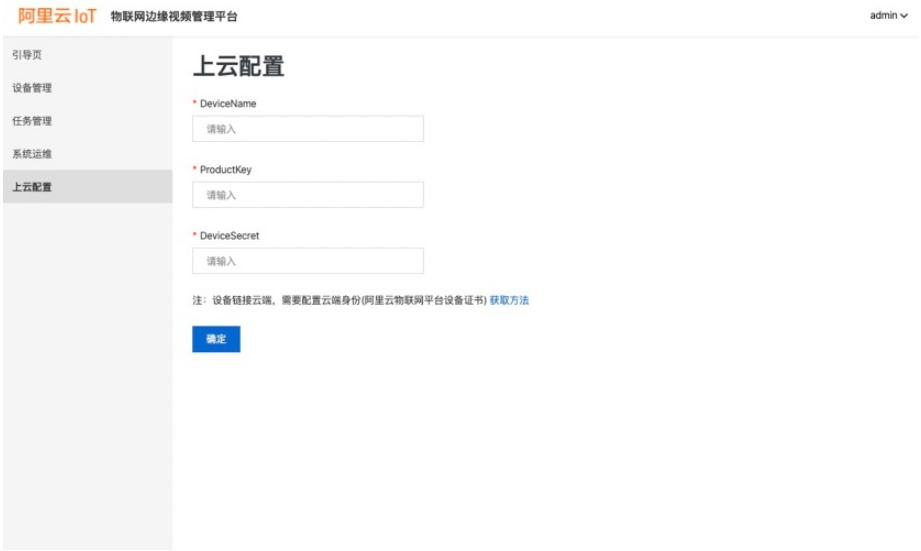
人脸识别NVR：类型编码是**frnvr**,备注信息内容：**"CategoryCode":"community/security/frnvr**



4.3 设备上云

4.3.1 LV边缘视频的网关设备上云

点击**上云配置**，将在3.1.3 节获取的LV边缘视频三元组信息输入到控制台，即可使LV边缘视频的网关设备在云端上线。特别注意，对于三个不同的实例，需要分别使用在3.1节创建的三个不同的LV边缘视频网关设备。



4.3.2 摄像头设备上云

在“上云配置 - 设置三元组”栏目中，点击“上传三元组 - 文件上传”，可以将从云端获取到的摄像头三元组到LV边缘视频中（“文件上传”中的文件选择我们在之前3.2.3步骤中下载好的“设备证书.csv”即可），如下图所示。注意：nvr自动拉取摄像头设备的三元组信息也需要上传设备证书，这些设备在3.2.3都创建完成



4.4 摄像头设备和设备三元组映射

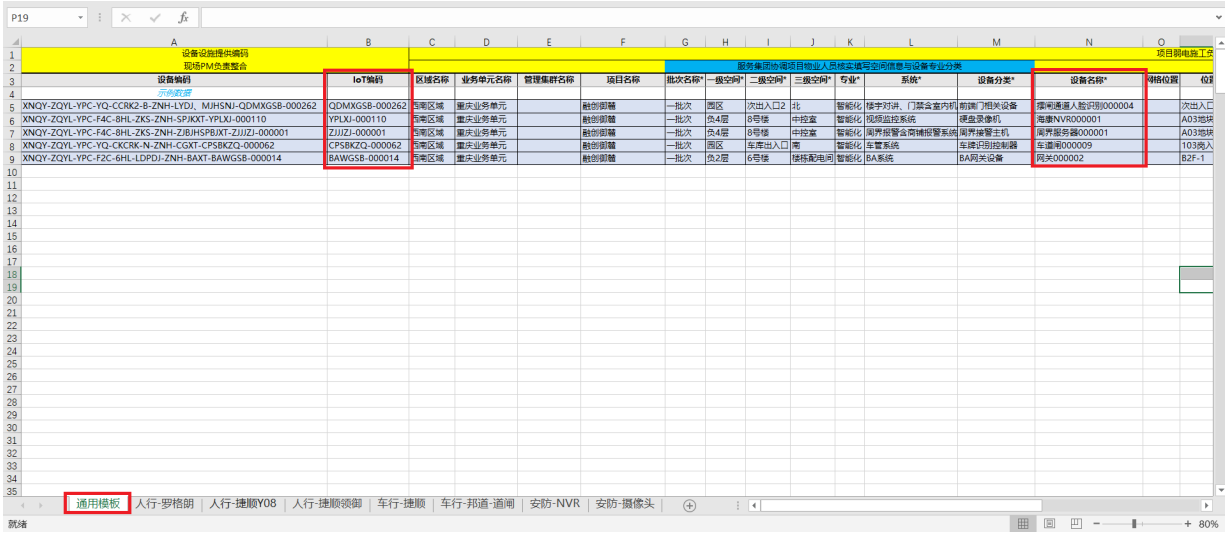
4.4.1 无NVR接入摄像头

在4.2.1节添加普通摄像头时，如果设备名称是**IOT编码**，则LV边缘视频本地控制台系统会自动进行三元组匹配。如果不是，则参考下面流程进行操作：

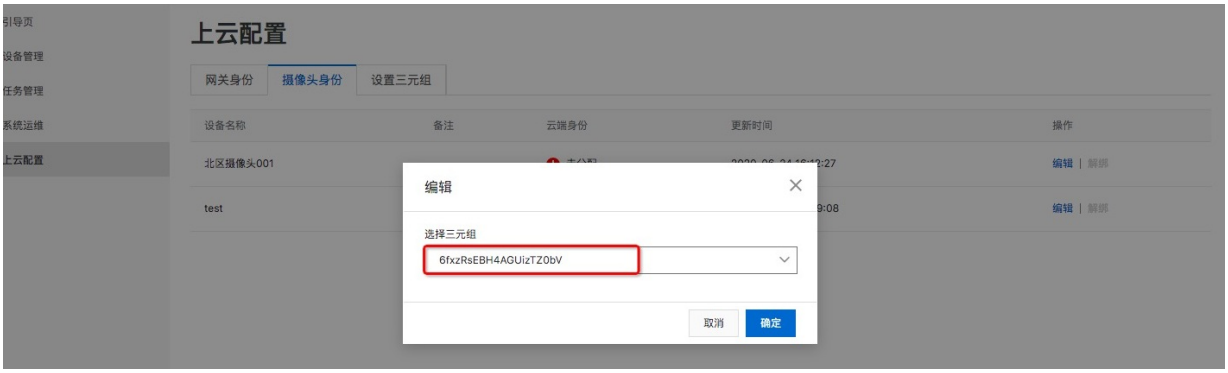
选择“上云配置 - 摄像头身份”，对于**云端身份**未分配的设备，点击**编辑**，进行本地设备与云端设备身份映射，如下图所示。



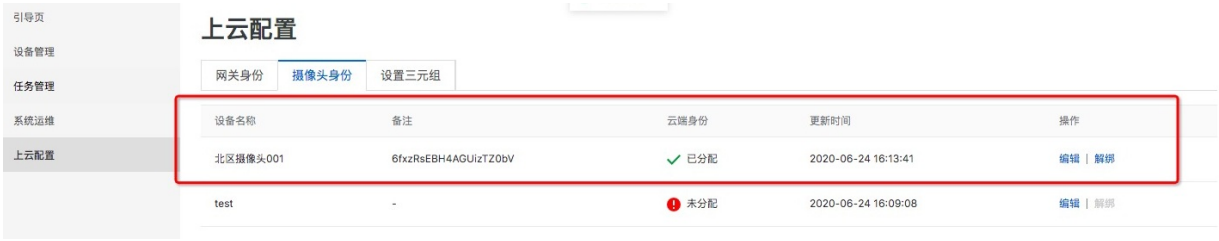
参照甲方提供的<IoT设备信息表>，如下图。在**通用模板 TAB页**，根据设备名称，找到对应的**IOT编码**



然后在下拉列表里找到对应的IOT编码，点击确定进行匹配，如下图所示。



匹配成功后，云端身份显示“已分配”，如下图所示。



4.4.2 有NVR接入摄像头

在4.2.2节添加完成某个NVR设备后，边缘控制台上会自动拉取的通过该NVR设备连接的所有IPC摄像头设备，即**有NVR接入摄像头**设备，如下图所示。



上面显示的就是所有通过某个NVR设备连接的所有IPC摄像头设备。**有NVR接入摄像头**的设备名称由三部分组成：

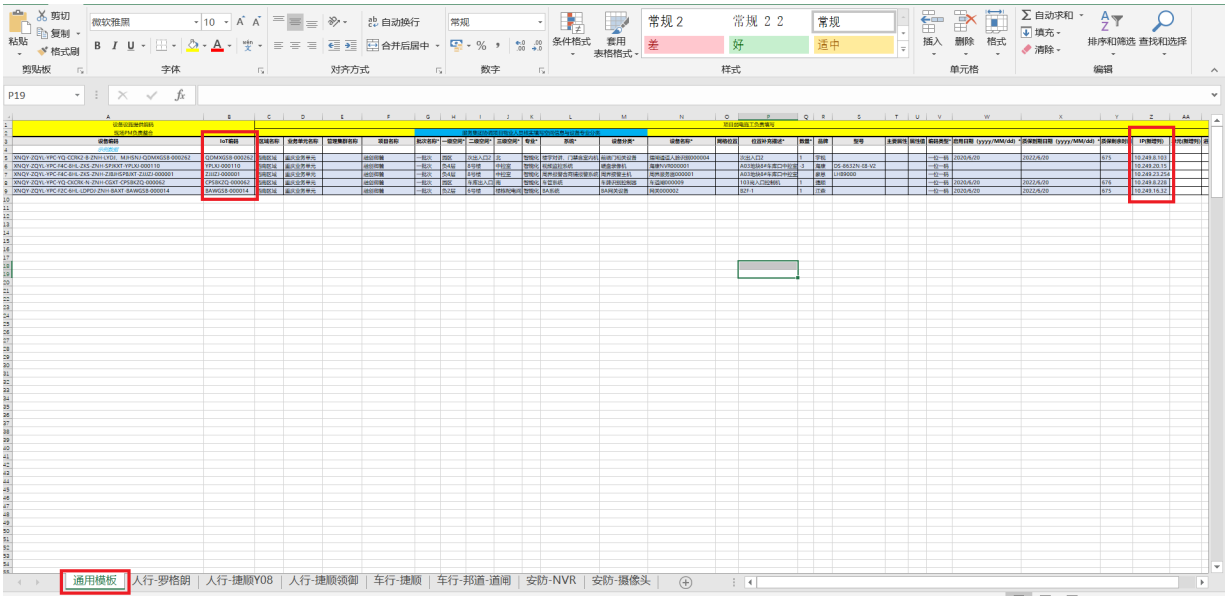
NVR设备的IoT编码+“有NVR接入摄像头”设备的通道号+“有NVR接入摄像头”设备的空间信息。

例如：**YPLXJ-000105_2_4#3单元西电梯：**其中NVR设备的IoT编码是**YPLXJ-000105，**“有NVR接入摄像头”设备的通道号是**2，**“有NVR接入摄像头”设备的空间信息是**4#3单元西电梯**

下面举例说明如何对这个设备名称为**YPLXJ-000105_2_4#3单元西电梯**的设备如何配置三元组

(1) 查找这个**有NVR接入摄像头**的设备连接的NVR IP地址

通过该设备连接的NVR的IOT编码是**YPLXJ-000105，**通过甲方提供的<设备信息表>的**通用模板**TAB页，可以找到对应的IP地址。如下图所示：



(2) 查找该**有NVR接入摄像头**设备的IP地址

参照甲方提供的<设备信息表>，在**安防-摄像头**TAB页，根据NVR设备的IP地址和设备名称里的通道号信息，找到该**有NVR接入摄像头**设备的IP地址。如图所示



填写责任方： 厂商补充设备IP、设备端口、类型名称、账号、密码、所属NVR IP、NVR上的通道号。（如果摄像头对应多个NVR，需依次列出）											
设备IP	设备端口	类型名称	账号	密码	连接NVR IP（1）	NVR上的通道号（1）	连接NVR IP（2）	NVR上的通道号（2）	连接NVR IP（3）	NVR上的通道号（3）	

关于通道号的补充说明：

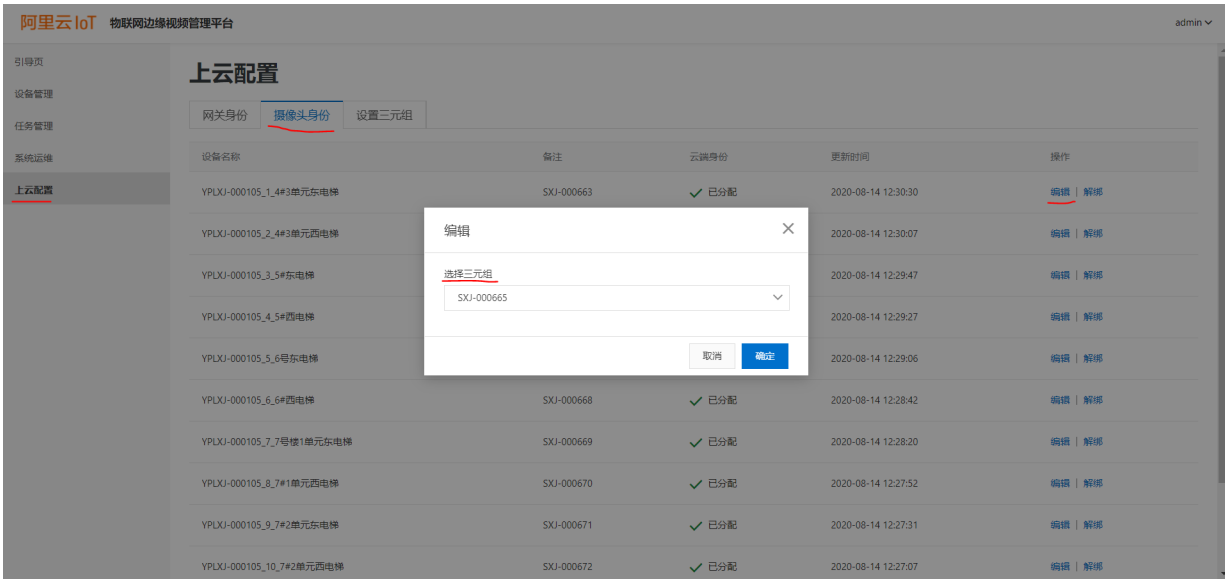
由于某些NVR设备，由于1~32的通道保留处理，对外提供的通道号是从33开始。这时根据自动拉取的有NVR接入摄像头的名称查找通道号时，需要将通道号减32。例如：YPLXJ-000105_33_14#东电梯，该设备的通道号可能是1，也可能是33. 具体可以查看上面**安防-摄像头**TAB页，如果对应NVR的通道信息显示是从1开始，而且最大通道号小于33，则实际通道号很可能是1。

(3)查找该**有NVR接入摄像头**设备的IoT编码

通过甲方提供的<设备信息表>的**通用模板**TAB页，可以通过IP地址，找到该设备对应的IoT编码

(4) 对该**有NVR接入摄像头**设备配置三元组

回到本小结开始LV边缘视频本地控制台，点击该**有NVR接入摄像头**设备右侧的**编辑，**在弹出的对话框里，查找选择（3）里找到的**IoT编码，**然后点击**确定**



匹配成功后，该设备的云端身份显示“已分配”。

4.4.3 NVR设备

NVR设备必须手动分配三元组，无论NVR设备名称与三元组的DeviceName是否相同

对NVR设备分配三元组操作，与5.2.4.1节无NVR接入摄像头类似：选择“上云配置 - 摄像头身份”，对于**云端身份**未分配的NVR设备，点击**编辑。**然后在下拉列表里选择NVR设备对应的IOT编码。NVR设备的IOT编码查找方法：在**通用模板 TAB页，**根据NVR设备的IP地址，找到对应的**IOT编码。**

4.5 设备联通性验证

在进行设备联通性验证前，需要首先查验厂商的设备状态

1、厂商设备的软件版本已经升级到完成与阿里物业一体机对接过的版本。

2、厂商的设备配置正确，例如配置了物业一体机(LV边缘视频)的IP地址和端口号。

3、厂商的设备IP地址设置正确，物业一体机可以正常ping通厂商设备

所有设备按照上面描述操作完成后，可以在本地控制台，点击设备管理，逐个查看上面添加的所有设备状态，验证设备的联通性。如果**云端连接**和**本地连接**状态都是**在线**，则说明设备成功与LV边缘视频的网关连接成功，如下图所示；如果有任何一个显示离线，说明设备与LV边缘视频的网关没有连接成功，这时可以排查的点：

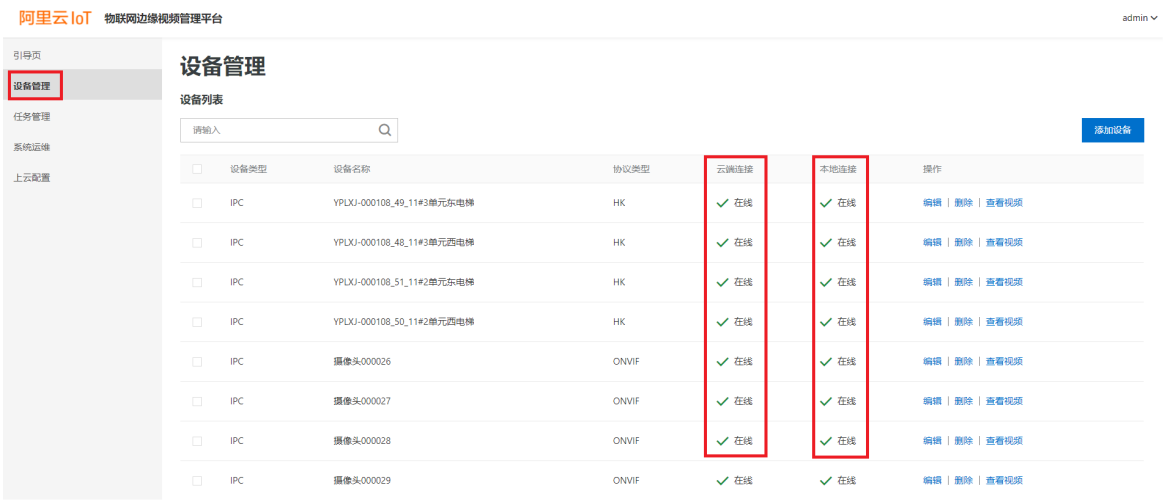
a 排查NVR设备内网是否通的。可以登录网关，去ping NVR设备

b 排查NVR设备配置是否正确：需要配置 LV边缘视频的网关IP地址，端口号，以及其下面接入的IPC摄像头信息

c 排查LV边缘视频本地控制台，创建NVR设备时配置是否有问题。例如接入方式应该是网关，而不是设备

d. 排查NVR设备分配的三元组，是否是有效的

当设备联通性验证失败时，也可以参考<**FAQ：常见问题排查手册**>进行问题排查



5 异常情况处理

5.1 更换设备DeviceName流程

如果因为某些原因，需要更换LV边缘视频本地控制台里IPC或NVR设备的DeviceName，处理流程如下：

1、在LP平台批量创建三元组

2、在LV边缘视频本地控制台，将IPC设备的三元组解绑

3、重新绑定三元组

4、重启LV边缘视频相应实例的容器(重新绑定的三元组，必须重启容器才能生效)

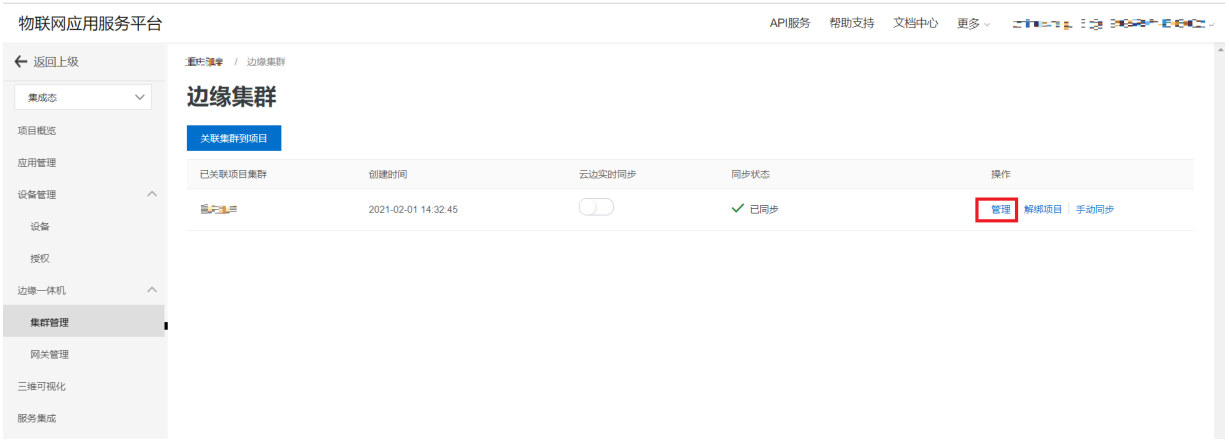
请参照5.1.1重启相应实例的容器

5、请阿里小二对IPC/NVR设备打标签

5.1.1 重启LV边缘实例对应的容器

首先需要登录LV边缘实例组件所在的物业一体机：

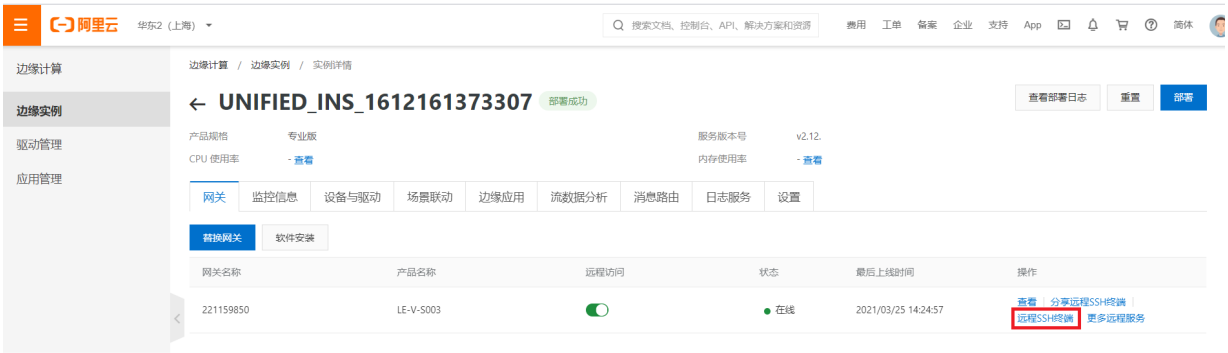
登录物联网应用服务平台，选择对应项目，进入边缘一体机-集群管理，找到对应集群名称，点击右侧**管理**：



点击组件管理，找到Link Edge边缘设备接入，点击**实例**，如下图所示。如果是融合实例，点击**前往实例**



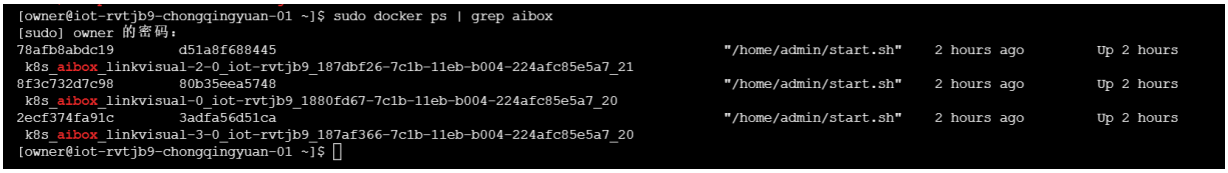
在打开的边缘实例页面，点击对应实例右侧的远程SSH终端



在打开的远程连接控制台，输入用户名和密码。融创项目固定为：owner,sunac123456

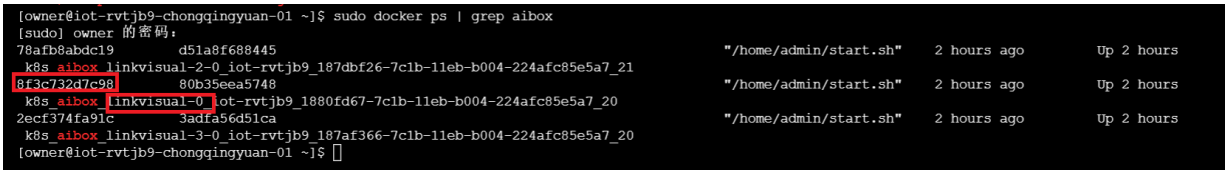


命令行输入：sudo docker ps | grep aibox，如下图所示：



注意：只重启更换了设备三元组所在的LV边缘实例对应容器，不要操作其他容器

查找边缘实例的容器ID: 找到有"linkvisual-0"所在行，最前面的一串字符为对应容器ID，如下图所示为：8f3c732d7c98。注意：不同的项目，对应的容器ID不同



命令行输入：docker restart 8f3c732d7c98，如下图所示：

```
[owner@iot-rvtjb9-chongqingyuan-01 ~]$ sudo docker ps | grep aibox
[sudo] owner 的密码：
78afb8abdc19        d51a8f688445                                                  "/home/admin/start.sh"   2 hours ago         Up 2 hours
 k8s_aibox_linkvisual-2-0_iot-rvtjb9_187dbf26-7c1b-11eb-b004-224afc85e5a7_21
8f3c732d7c98        80b35eea5748                                                  "/home/admin/start.sh"   2 hours ago         Up 2 hours
 k8s_aibox_linkvisual-0_iot-rvtjb9_1880fd67-7c1b-11eb-b004-224afc85e5a7_20
2ecf374fa91c        3adfa56d51ca                                                  "/home/admin/start.sh"   2 hours ago         Up 2 hours
 k8s_aibox_linkvisual-3-0_iot-rvtjb9_187af366-7c1b-11eb-b004-224afc85e5a7_20
[owner@iot-rvtjb9-chongqingyuan-01 ~]$ docker restart 8f3c732d7c98
```

5.2 NVR下的IPC摄像头无法自动拉取处理方法

如果NVR设备内网不通，这时在LV边缘视频的本地控制台添加NVR设备，无法自动拉取NVR下面挂载的IPC摄像头，即使NVR设备内网联通上线后也不行。

出现该问题后的处理方法：在LV边缘视频的本地控制台，按照下面顺序处理：

a. 在上云配置里先解绑NVR，

b. 在设备管理里删除NVR设备，

c. 在设备管理里添加NVR设备，

d. 在上云配置里再分配三元组。

3.3.5.5.2. 广拓周界报警主机设备接入

1 广拓支持对接的型号说明

对接型号支持	A7全系列报警主机
系统版本	N/A

2 周界报警设备接入操作

2.1 登录应用服务平台

2.1.1 选择项目

登录阿里云应用服务平台，找到需要配置的项目，该项目已由项目方创建，如果未创建，请联系项目方。



2.1.2 选择对应网关的设备集成

进入项目内，找到**边缘一体机-网关管理**，如下图所示。点击网关对应的**设备集成**



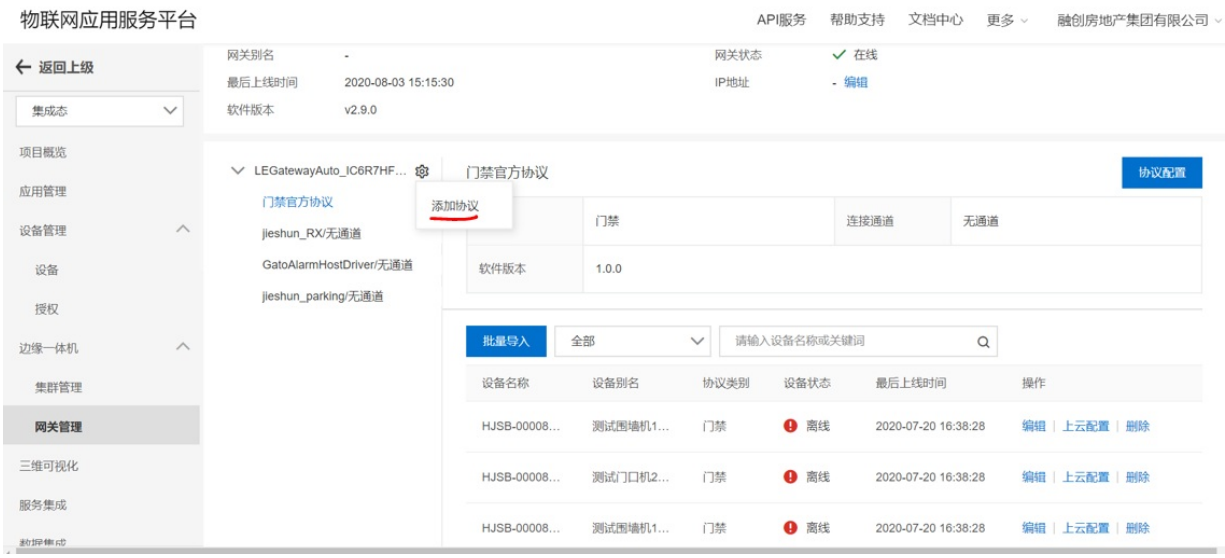
2.2广拓周界报警主机设备集成

2.2.1 添加广拓周界报警协议

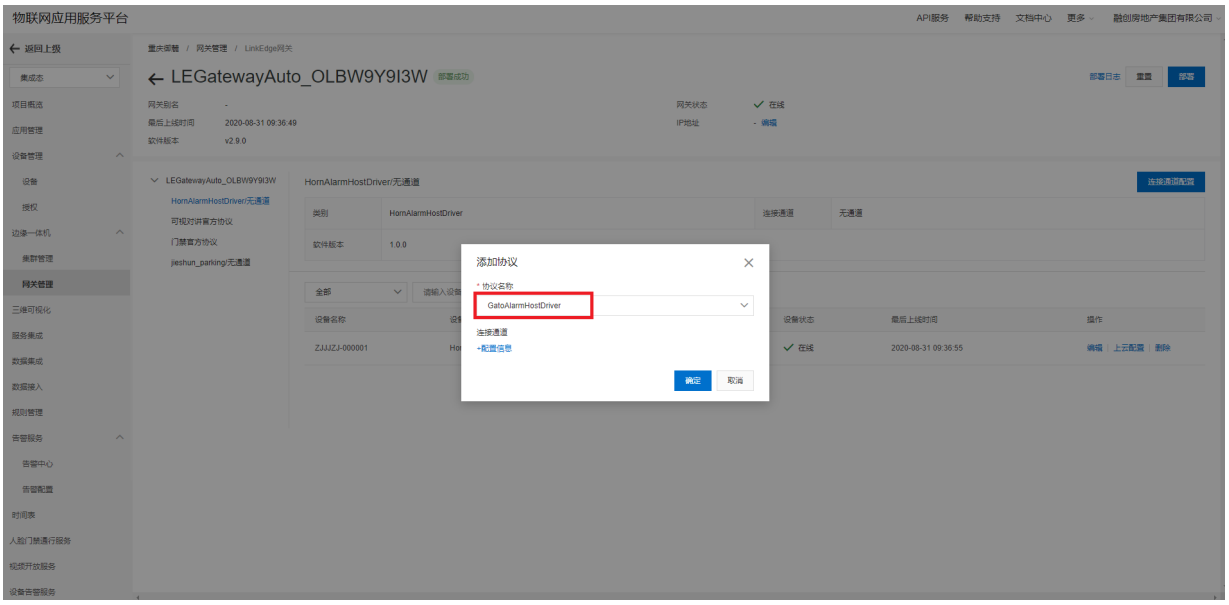
首次添加，选择新建协议



网关里已添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



协议名称选择GatoAlarmHostDriver, 如下图所示：

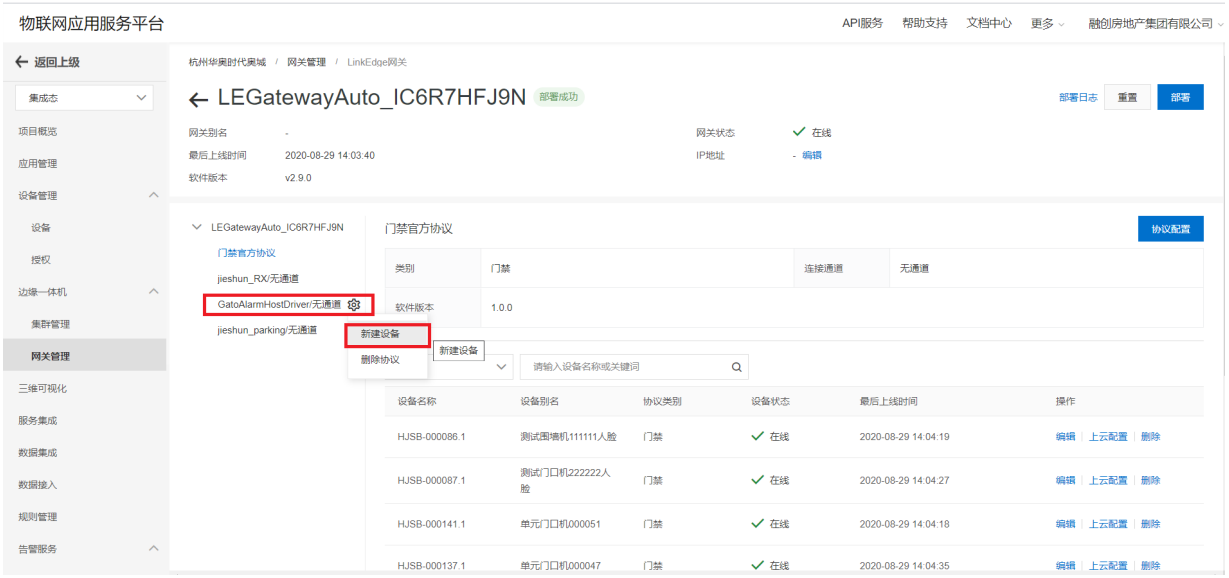


选择完协议，点击确定。

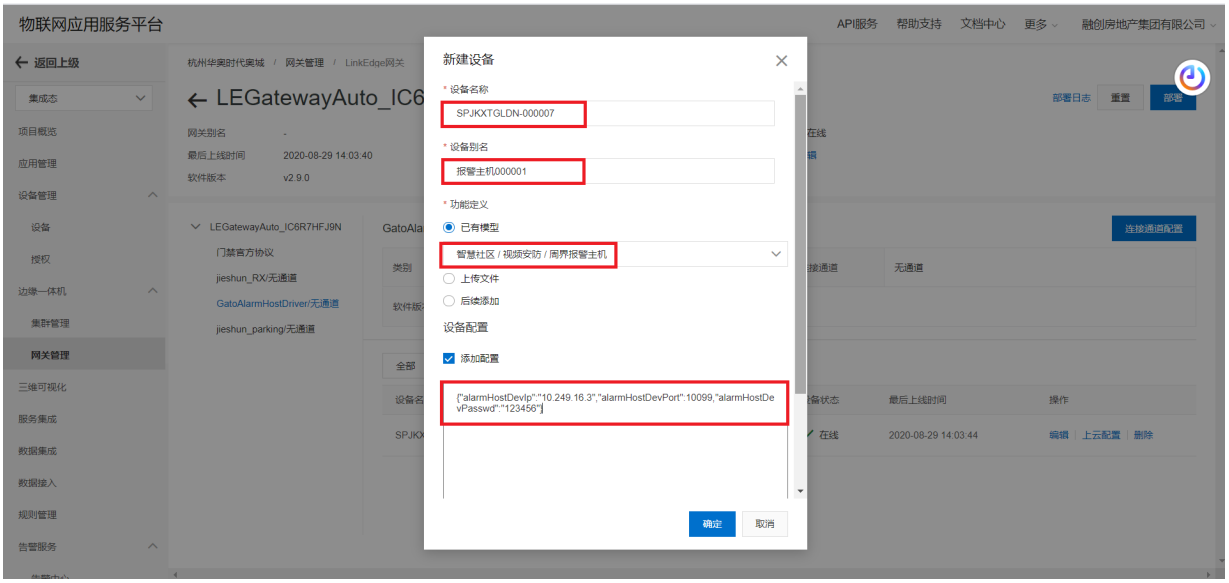
说明：选择完协议，不需要点击**+配置信息**对驱动进行配置。边缘网关添加的驱动，只能在LP平台进行配置。

2.2.2 添加广拓周界报警设备

如下图所示，鼠标移动到到刚添加的**GatoAlarmHostDriver**驱动，点击**新建设备**，进行设备的添加。



如下图所示，在**GatoAlarmHostDriver**协议下新增周界报警设备。



其中：

设备名称：参照甲方提供的<IoT设备信息表>，在**安防-广拓-周界报警主机**TAB页查看设备**IP**，例如10.249.16.3。

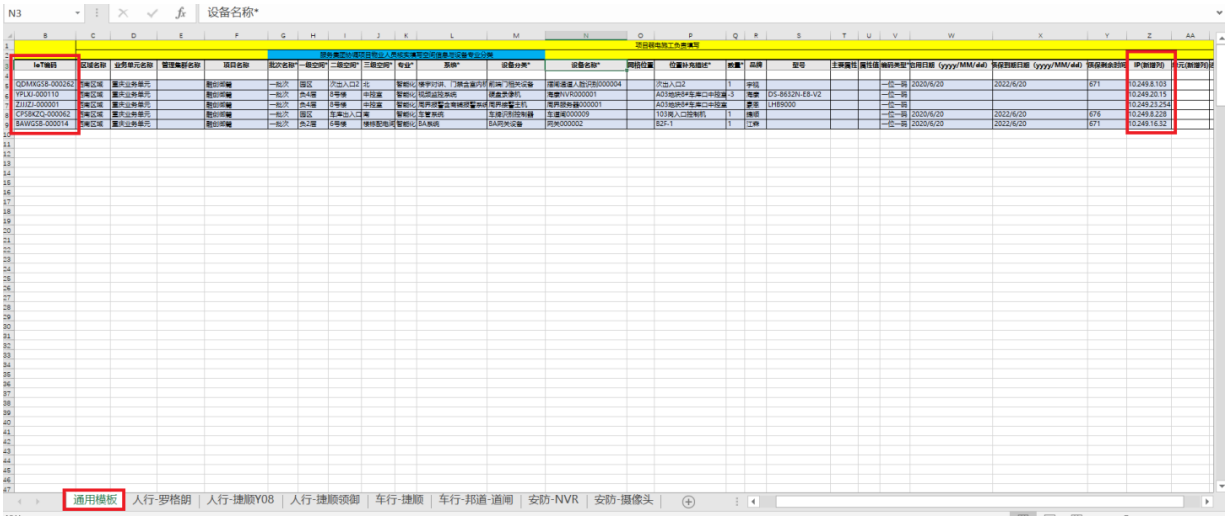


填写责任方:
IP: 项目方提供设备IP地址
设备配置：设备厂商广拓提供alarmHostDevPort端口和alarmHostDevPasswd

IP	设备配置
10.249.16.3	{"alarmHostDevIp":"10.249.16.3", "alarmHostDevPort":10099, "alarmHostDevPasswd":"123456" }

... | 人行-捷顺Y08 | 人行-捷顺领御 | 车行-捷顺 | 车行-邦道-道闸 | 安防-NVR | 安防-摄像头 | 安防-广拓-周界报警主机

然后在<IoT设备信息表>的**通用模板**TAB页，根据IP地址查找对应的**IoT编码**，例如SPJKXTGLDN-000007，如下图所示。



设备别名：参照甲方提供的<设备信息表>**通用模板**TAB页中**设备IP**对应的**设备名称**。配置示例为：报警主机000001

设备配置：参照甲方提供的<IoT设备信息表>，在**安防-广拓-周界报警主机**TAB页查看**设备配置**

配置示例：{"alarmHostDevIp":"10.249.16.3","alarmHostDevPort":10099,"alarmHostDevPasswd":"123456"}

2.3 周界报警主机设备与空间绑定

周界报警设备如果需要与空间绑定，可根据设备的IP地址，参照甲方提供的<IoT设备信息表>**通用模板**TAB页的**位置补充描述**，可以得到周界报警主机设备的空间信息。具体绑定方法，请参照<**物业管理一体机实施手册**>的第4章：设备绑定空间。

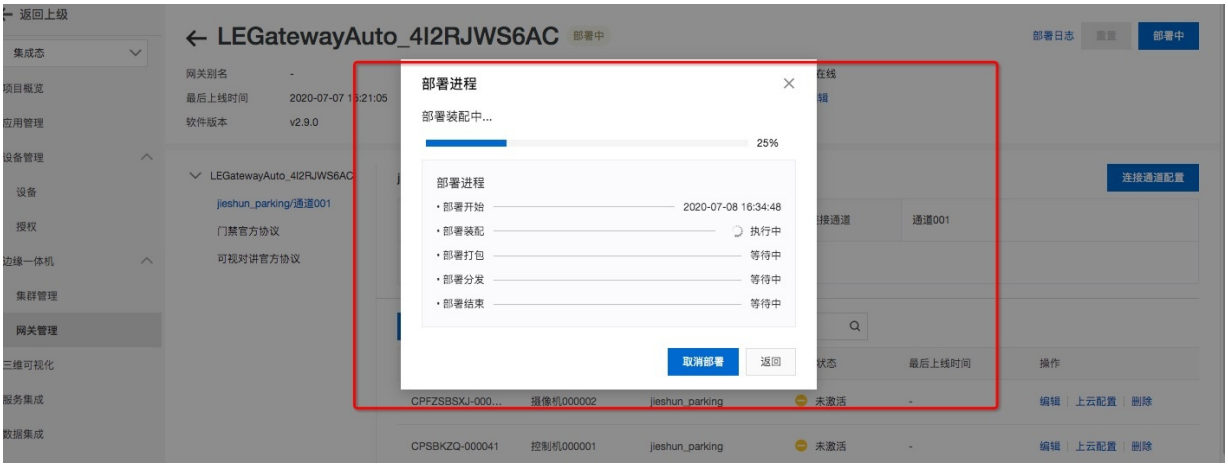
2.4 部署设备配置

在完成设备的配置后，点击部署，进行设备配置的协议下发到边缘服务器。如下图所示。

特别提醒：驱动部署时，网关侧对应驱动将重启，该驱动下的所有设备将出现离线后在线。这期间设备不可用，请部署人员提前与甲方沟通部署时间



点击部署后可以查看设备配置下发的状态和部署进度。部署大概需要5分钟。



说明：如果某个设备有新更新的配置信息，重新点击部署后，只会下发有改动的设备配置协议，其余未修改的配置信息不会再次下发到边缘服务器，部署的时间根据修改的设备数量有关。

3.3.5.5.3. 豪恩周界报警主机设备接入

1 豪恩支持对接的型号说明

对接型号支持	LHB9000
系统版本	N/A

2 周界报警设备接入操作

2.1 登录应用服务平台

登录阿里云应用服务平台，找到需要配置的项目，该项目已由项目方创建，如果未创建，请联系项目方。



2.1.1 选择项目

登录阿里云应用服务平台，找到需要配置的项目，该项目已由项目方创建，如果未创建，请联系项目方。



2.1.2 选择对应网关的设备集成

进入项目内，找到**边缘一体机-网关管理**，如下图所示。点击网关对应的**设备集成**



2.2 豪恩周界报警主机设备集成

2.2.1 添加豪恩周界报警协议

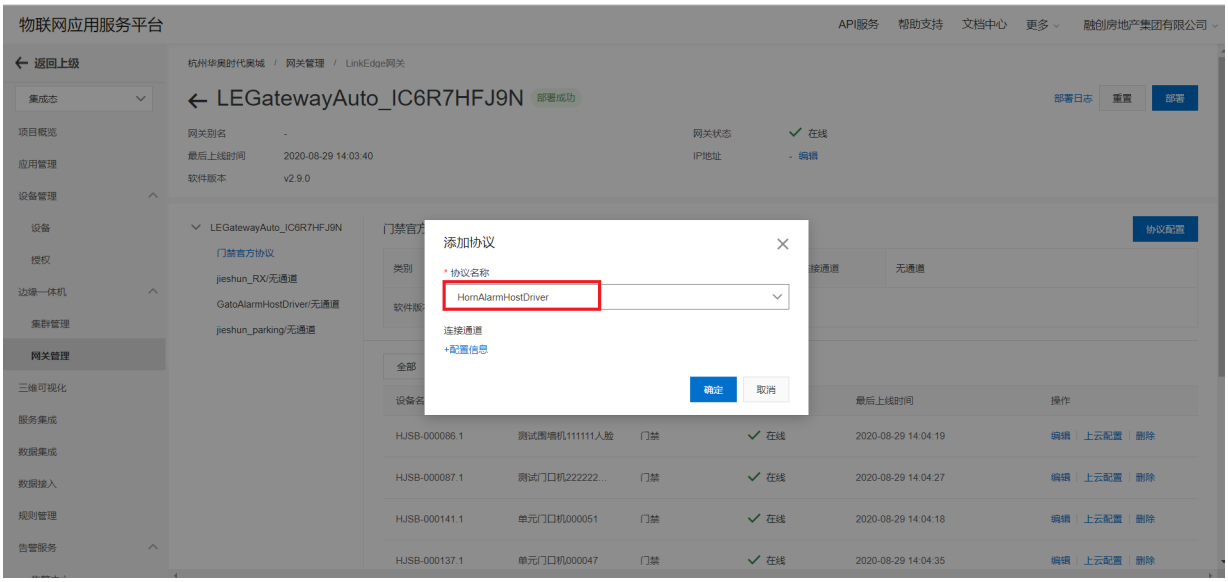
首次添加，选择新建协议



网关里已添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



协议名称选择**HornAlarmHost Driver**,如下图所示：

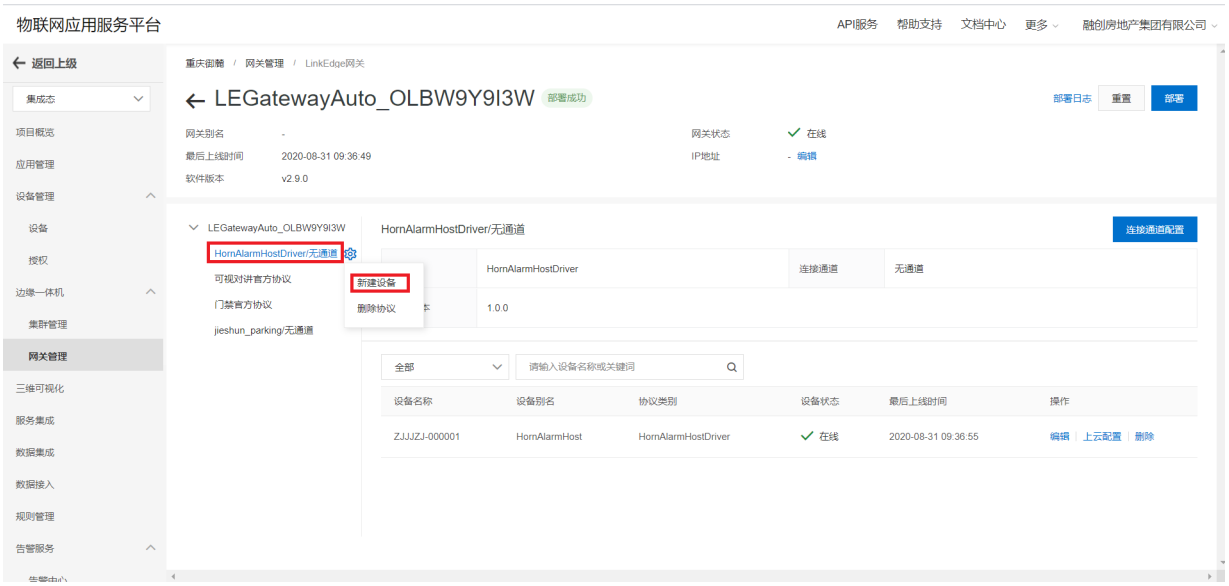


选择完协议，点击确定。

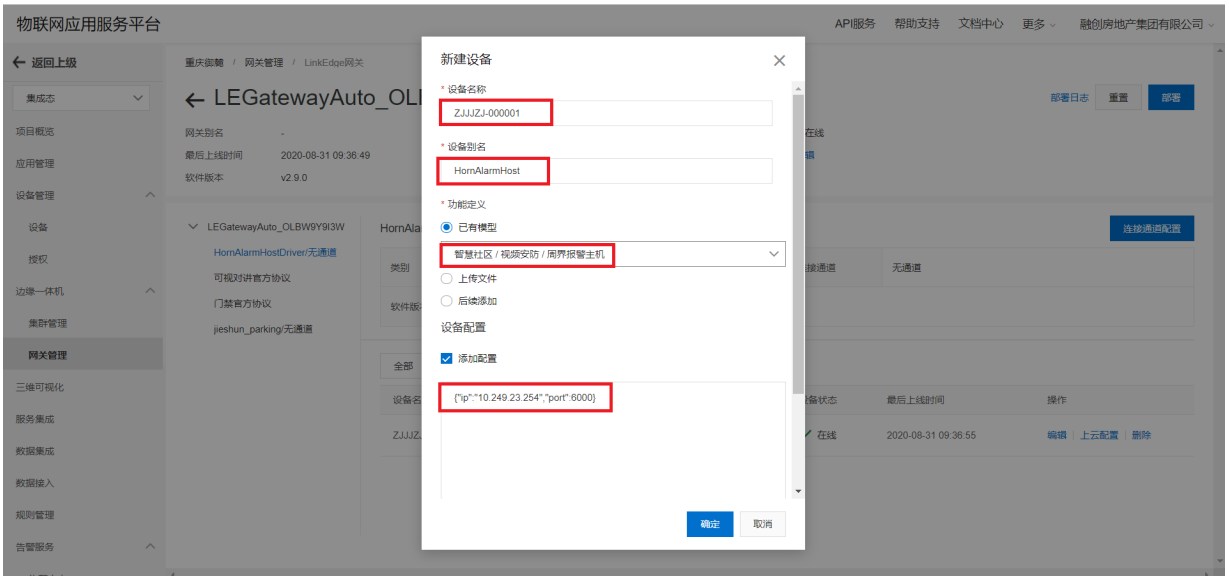
说明：选择完协议，不需要点击**+配置信息**对驱动进行配置。边缘网关添加的驱动，只能在LP平台进行配置。

2.2.2 添加豪恩周界报警设备

如下图所示，鼠标移动到到刚添加的**HornAlarmHost Driver**驱动，点击**新建设备**，进行设备的添加。



如下图所示，在**HornAlarmHostDriver**协议下新增周界报警设备。



其中：

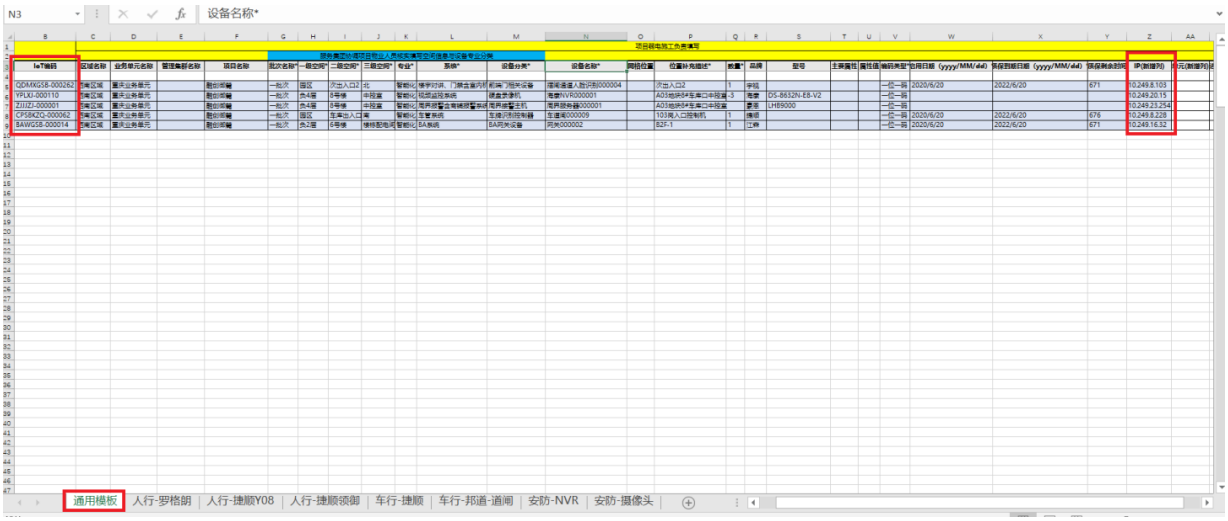
设备名称：参照甲方提供的<IoT设备信息表>，在**安防-豪恩-周界报警主机**TAB页查看设备**IP**，例如10.249.23.254。如下图所示：



填写责任方: IP: 项目方提供设备IP地址 端口： 设备厂商豪恩提供	
IP	端口
10.249.23.254	6000

... | 车行-捷顺 | 车行-邦道-道闸 | 安防-NVR | 安防-摄像头 | 安防-广拓-周界报警主机 | 安防-豪恩-周界报警主机

然后在<IoT设备信息表>的**通用模板**TAB页，根据IP地址查找对应的**IoT编码**，例如ZJJZJ-000001，如下图所示。



通用模板 | 人行-罗格朗 | 人行-捷顺Y08 | 人行-捷顺领御 | 车行-捷顺 | 车行-邦道-道闸 | 安防-NVR | 安防-摄像头

设备别名：参照甲方提供的<设备信息表>**通用模板**TAB页中**设备IP**对应的**设备名称**。配置示例为：HornAlarmHost

设备配置：参照甲方提供的<IoT设备信息表>，在**安防-豪恩-周界报警主机**TAB页，将IP和端口号填入配置里。

配置示例：{"ip":"10.249.23.254","port":6000}。特别注意：豪恩的周界报警主机端口号默认是6000. 如果设备信息表提供的端口不是6000，需要与设备厂商或者鸿雁(驱动提供商)确认。

2.3 周界报警主机设备与空间绑定

周界报警设备如果需要与空间绑定，可根据设备的IP地址，参照甲方提供的<IoT设备信息表>**通用模板**TAB页的**位置补充描述**，可以得到周界报警主机设备的空间信息。具体绑定方法，请参照<**物业管理一体机实施手册**>的第4章：设备绑定空间。

2.4 部署设备配置

在完成设备的配置后，点击部署，进行设备配置的协议下发到边缘服务器。如下图所示。

特别提醒：驱动部署时，网关侧对应驱动将重启，该驱动下的所有设备将出现离线后在线。这期间设备不可用，请部署人员提前与甲方沟通部署时间



点击部署后可以查看设备配置下发的状态和部署进度。部署大概需要5分钟。



说明：如果某个设备有新更新的配置信息，重新点击部署后，只会下发有改动的设备配置协议，其余未修改的配置信息不会再次下发到边缘服务器，部署的时间根据修改的设备数量有关。

3.3.5.5.4. 精华隆周界报警主机设备接入

1精华隆支持对接的型号说明

对接型号支持	EP3164-IP
系统版本	N/A

2周界报警设备接入操作

2.1 登录应用服务平台

2.1.1 选择项目

登录阿里云应用服务平台，找到需要配置的项目，该项目已由项目方创建，如果未创建，请联系项目方。



2.1.2 选择对应网关的设备集成

进入项目内，找到**边缘一体机-网关管理**，如下图所示。点击网关对应的**设备集成**



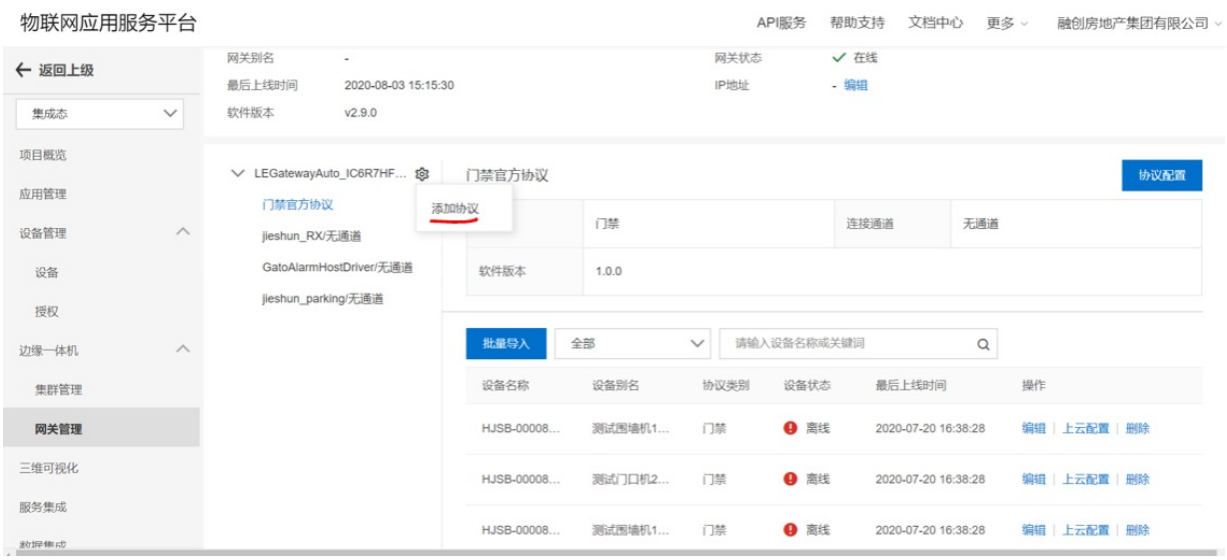
2.2精华隆周界报警主机设备集成

2.2.1 添加精华隆报警协议

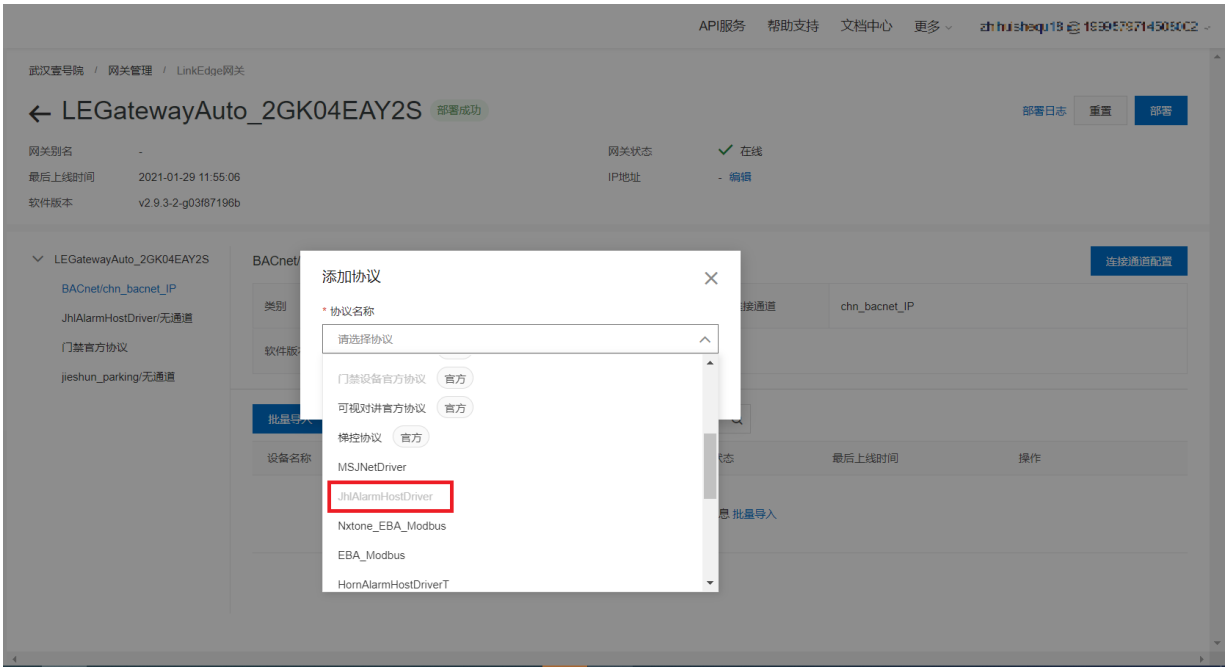
首次添加，选择新建协议



网关里已添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



协议名称选择**JhlAlarmHost Driver**,如下图所示：



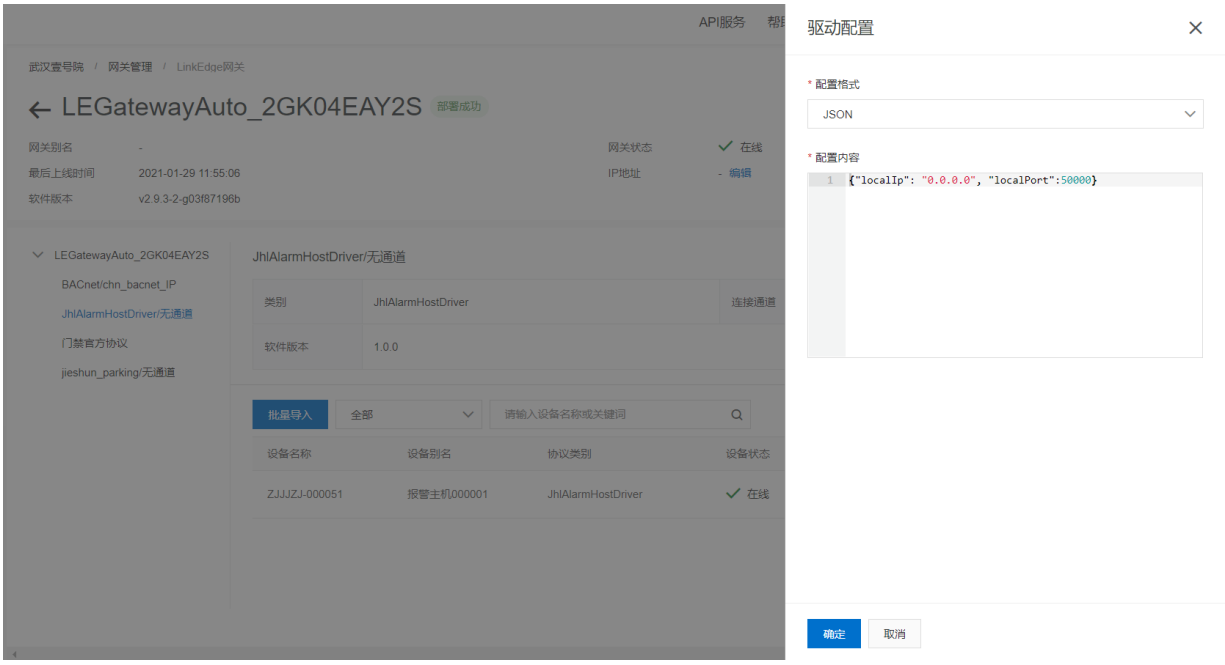
选择完协议，点击确定。

2.2.2 对驱动进行配置

添加完协议后，点击**驱动配置**，对精华隆周界报警主机的协议进行配置，如下图所示：

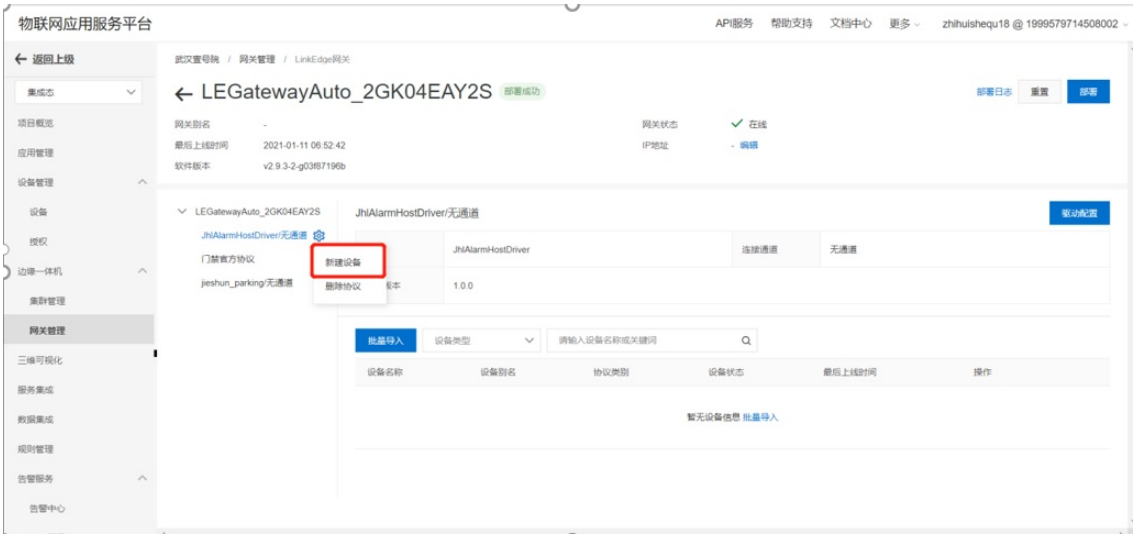


精华隆报警主机驱动默认监听端口50000，请将下面配置内容输入：{"localIp":"0.0.0.0","localPort":50000}

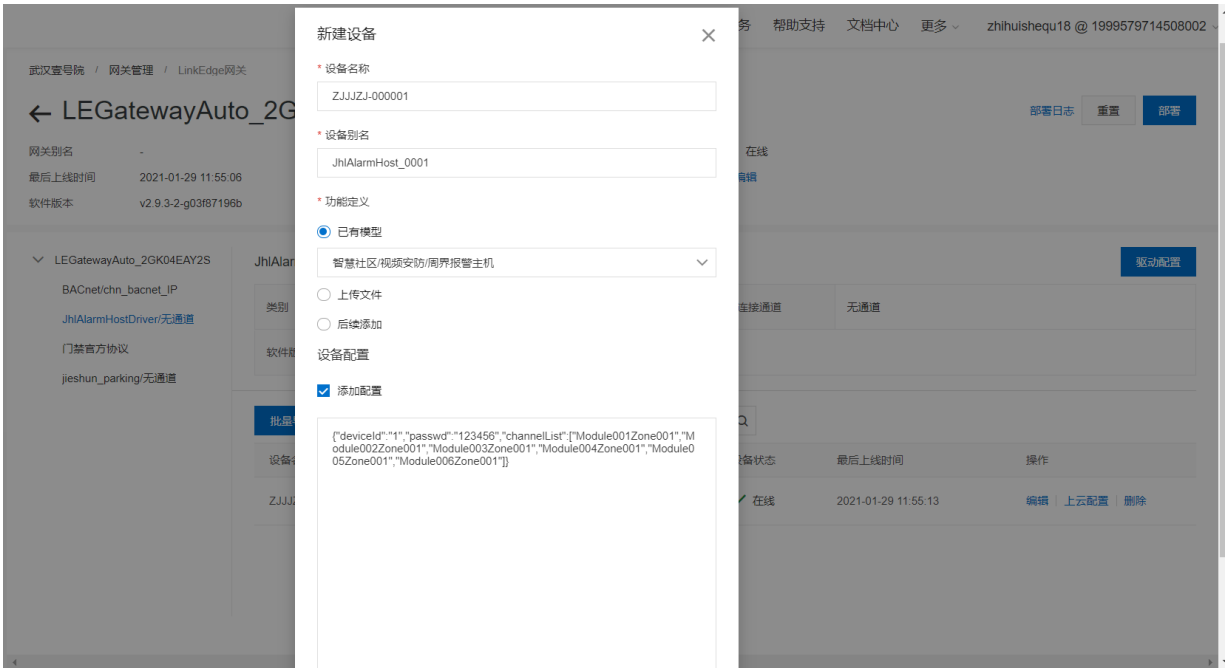


2.2.2 添加精华隆报警设备

如下图所示，鼠标移动到刚添加的**JhlAlarmHostDriver**驱动，点击**新建设备**，进行设备的添加。



如下图所示，在**JhlAlarmHost Driver**协议下新增报警设备。



其中：

设备名称：参照甲方提供的<IoT设备信息表>， 例如ZJJJZJ-000001。

设备别名：参照甲方提供的<设备信息表>，配置示例为：JhlAlarmHost_0001

功能定义：选择已有模型：智慧社区-视频安防-周界报警主机

设备配置：参照甲方提供的设备配置信息表。

配置示例如下：

{"deviceId":"1","passwd":"123456","channelList":
["Module001Zone001","Module002Zone001","Module003Zone001","Module004Zone001","Module005Zone001","Module006Zone001"]}

特别注意：精华隆的报警主机的deviceId(设备ID) passwd(设备操作密码) channelList(防区列表)需要与设备厂家联系。

2.3 周界报警主机设备与空间绑定

周界报警设备如果需要与空间绑定，可根据设备的IP地址，参照甲方提供的<IoT设备信息表>**通用模板**TAB页的**位置补充描述**，可以得到周界报警主机设备的空间信息。具体绑定方法，请参照<**物业管理一体机实施手册**>的第4章：设备绑定空间。

2.4 部署设备配置

在完成设备的配置后，点击部署，进行设备配置的协议下发到边缘服务器。如下图所示。

特别提醒：驱动部署时，网关侧对应驱动将重启，该驱动下的所有设备将出现离线后在线。这期间设备不可用，请部署人员提前与甲方沟通部署时间



点击部署后可以查看设备配置下发的状态和部署进度。部署大概需要5分钟。



说明：如果某个设备有新更新的配置信息，重新点击部署后，只会下发有改动的设备配置协议，其余未修改的配置信息不会再次下发到边缘服务器，部署的时间根据修改的设备数量有关。

3.3.5.5.5. 安防设备接入-三实例版

名词解释

下面对文档中使用的专业名词分别说明，便于使用手册的人更好的理解

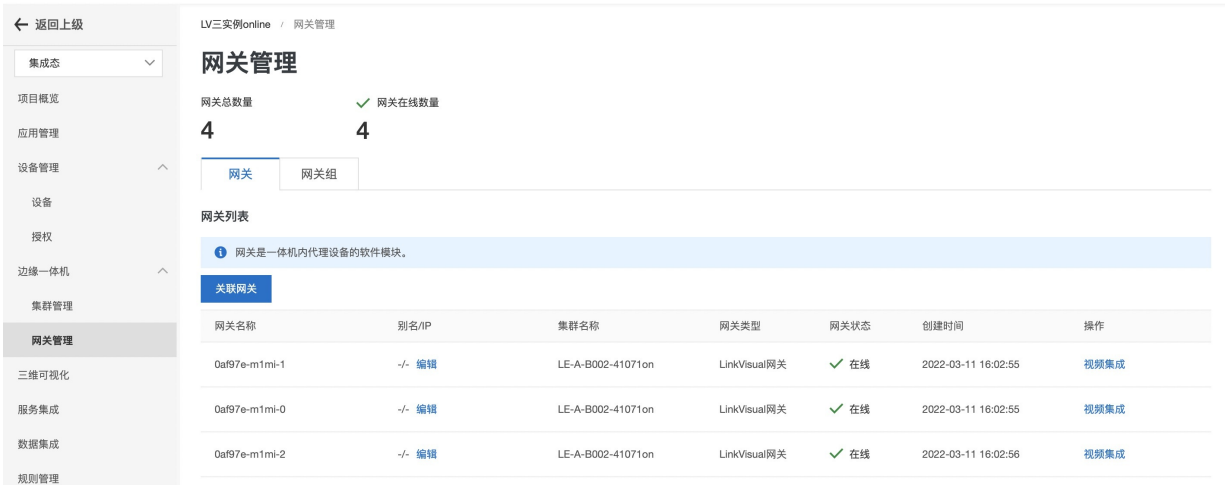
名词	说明
物联网应用服务平台	阿里巴巴IoT智能化解决方案，能够集成的大量行业智能硬件和软件，快速打通OT与IT，并通过容器化技术，实现应用及解决方案的自动化部署和交付
IPC	IPC(IP Camera)即网络摄像头
NVR	NVR(Network Video Recorder)即网络硬盘录像机)，一般指项目上采购的海康、宇视等NVR设备

1 应用服务平台安防设备集成

在安装部署视频监控安防设备前，请确保项目的边缘集群已经部署了Link Visual 三实例边缘视频组件。可以集群管理里查看，如下图所示：



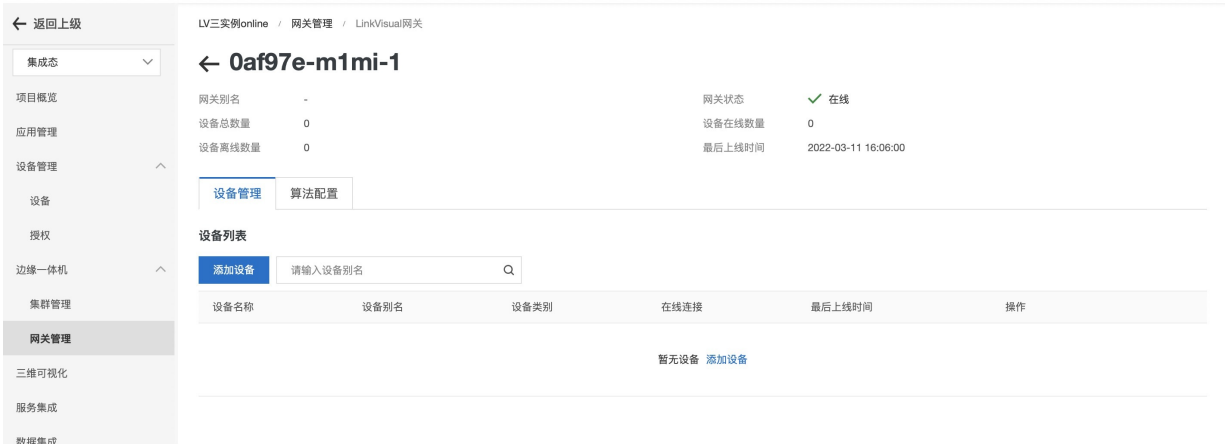
安装完成后在，在应用服务平台**边缘一体机 - 网关管理** 页面便会露出三个LV边缘视频(AIBOX)网关实例，三个网关实例前缀相同，以0，1，2序号进行区分，入下图所示：



目前每个实例可最多接入256个设备包括安防摄像头设备和NVR设备。建议在项目部署实施时，先在第一个实例上按照最大支持的256个摄像头完成添加，再对第二个实例按照最大支持的256个摄像头完成添加，最后对第三个实例进行添加。注意同一个NVR设备下面连接的所有摄像头，需要安装部署到同一个实例里。

1.1 添加设备到第一个LV边缘网关实例

选择网关名称尾部序号为0，点击右侧的**视频集成**，进入第一个LV边缘网关实例的视频集成页面，如下图所示：

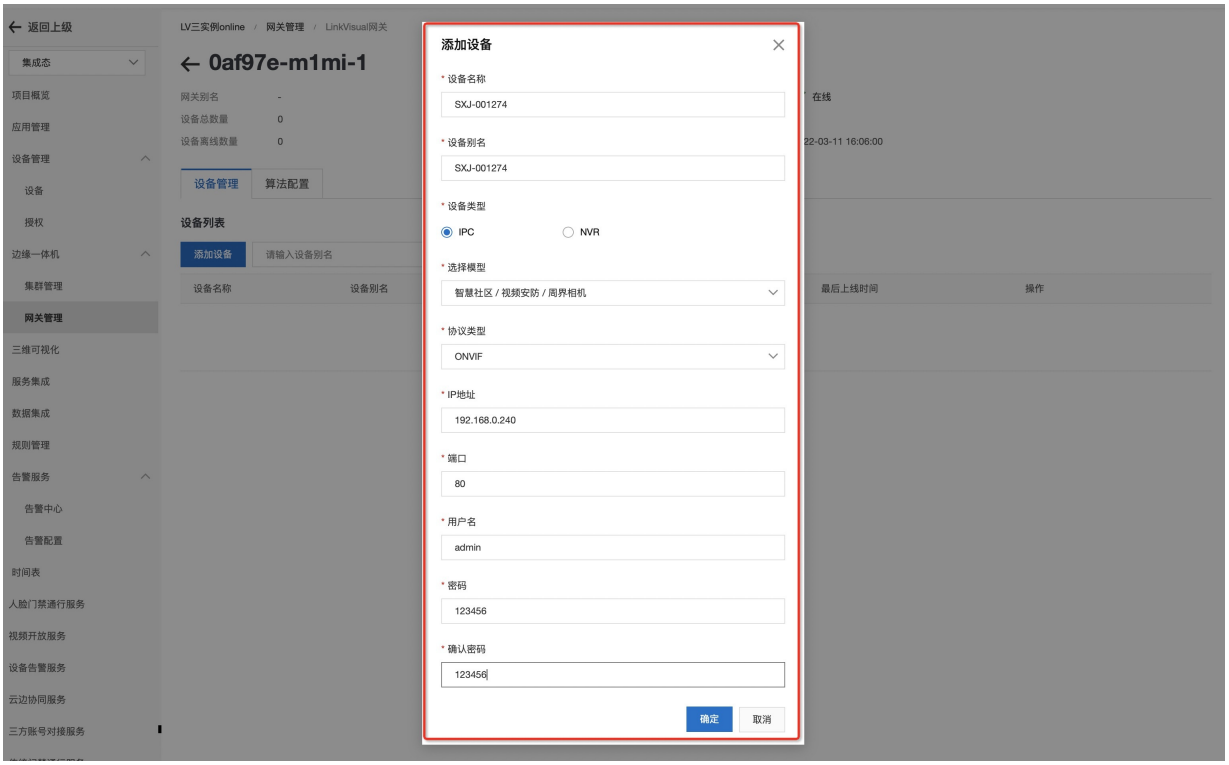


1.1.1 无NVR接入摄像头和有NVR接入摄像头类型

摄像头设备分两类，一类是直连LV边缘视频的摄像头设备，我们定义为**无NVR接入摄像头**。另外一类摄像头先接入到NVR设备，然后NVR设备再连接LV边缘视频，我们定义为**有NVR接入摄像头**。

1.1.2. 添加无NVR接入摄像头

请特别注意，只需要添加无NVR接入摄像头设备和NVR设备。通过NVR接入的摄像头设备不需要添加。当NVR设备添加成功后，通过NVR接入的摄像头设备将会自动拉取出来。需要将IPC设备加入到LV边缘视频中，点击“**设备管理 - 添加设备**”进行设备信息的相关配置，如下图所示。



设备名称：填写无NVR接入设备的**IoT编码**，请遵循**IoT编码**规范，并确保IoT编码的唯一性。

设备别名：填写无NVR接入设备的**设备别名**，请遵循IoT**设备别名**规范，并确保设备别名的唯一性。

设备类型：直接接入IPC设备则选择“**IPC**“即可。

选择模型：点击选择模型，选择智慧社区 - 视频安防后进行选择， 可以选择的摄像头产品类型共有四种：普通摄像头、违停球相机、人脸抓拍机、周界相机。如下图所示：



协议类型：请基于设备所支持的协议选择协议类型。目前我们支持海康，大华，onvif及国标

IP：请填写设备的IP地址。输入示例为192.168.0.240

端口：请填写设备的端口

用户名：请填写设备的用户名

密码：请填写设备的秘密

1.1.3. 添加NVR设备

在LV边缘视频云端控制台，当添加NVR设备之前，需要确认两点：

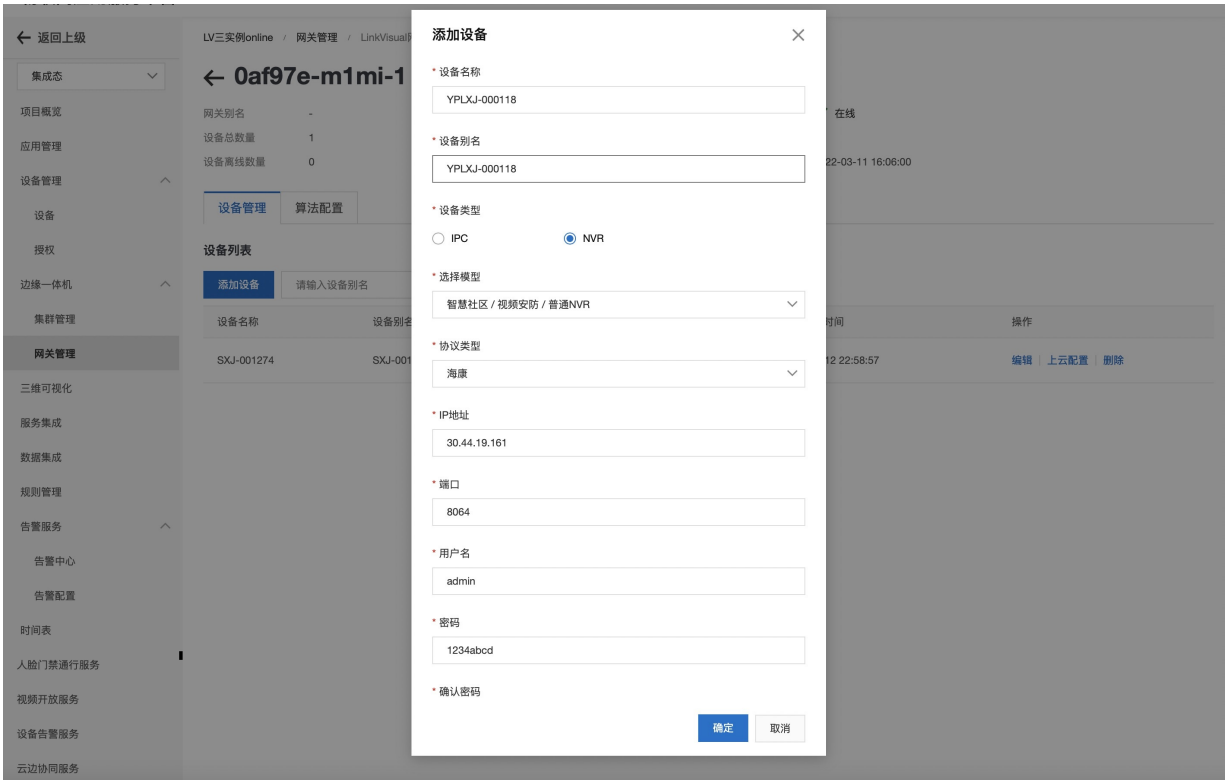
a: 要确保NVR设备内网正常且在线。因为在添加NVR摄像头时，NVR设备下面连接的摄像头，将自动拉取添加。如果NVR设备不在线情况下完成添加，当NVR摄像头上线后，不会自动拉取。

b: 要确保所有IPC正确配置到NVR。不然即使配上了NVR，以后也要删除NVR重新配置

如果发现NVR设备下面连接的摄像头没有自动拉取出来，需要删除NVR设备后再重新添加。详细请参考3.2节："NVR下的IPC摄像头无法自动拉取处理方法"

如果某个NVR部署完成后，需要再新添加几个摄像头，也需要删除整个NVR，再全量添加

将NVR设备加入到LV边缘视频方法：点击“设备管理 - 添加设备”进行设备信息的相关配置，如下图所示。



设备名称：填写设备的**IoT编码**，请遵循**IoT编码**规范，并确保IoT编码的唯一性。

设备别名：填写无NVR接入设备的**设备别名，**请遵循IoT**设备别名**规范，并确保设备别名的唯一性。

设备类型：NVR类型的设备在添加的时候，设备类型需要选择**NVR**。

选择模型：选择设备的**模型**。与1.1.2节类似，点击选择模型，选择智慧社区 - 视频安防后进行选择， 可以选择的摄像头产品类型共有四种：普通摄像头、违停球相机、人脸抓拍机、周界相机

协议类型：请基于设备所支持的协议选择协议类型。目前我们支持海康，大华，onvif及国标

IP：请填写设备的IP地址。输入示例为30.44.19.161

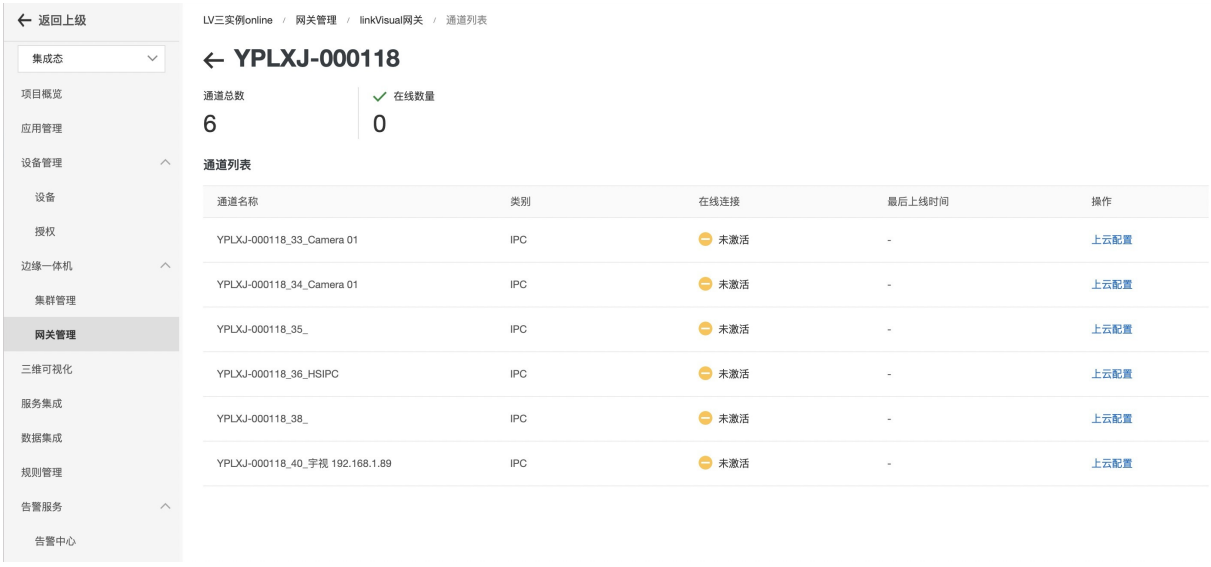
端口：请填写设备的端口

用户名：请填写设备的用户名

密码：请填写设备的秘密

1.1.4.有NVR接入摄像头设备配置

在2.1.3节添加完成某个NVR设备后，云端控制台上会自动拉取的通过该NVR设备连接的所有IPC摄像头设备，即**有NVR接入摄像头**设备，如下图所示。

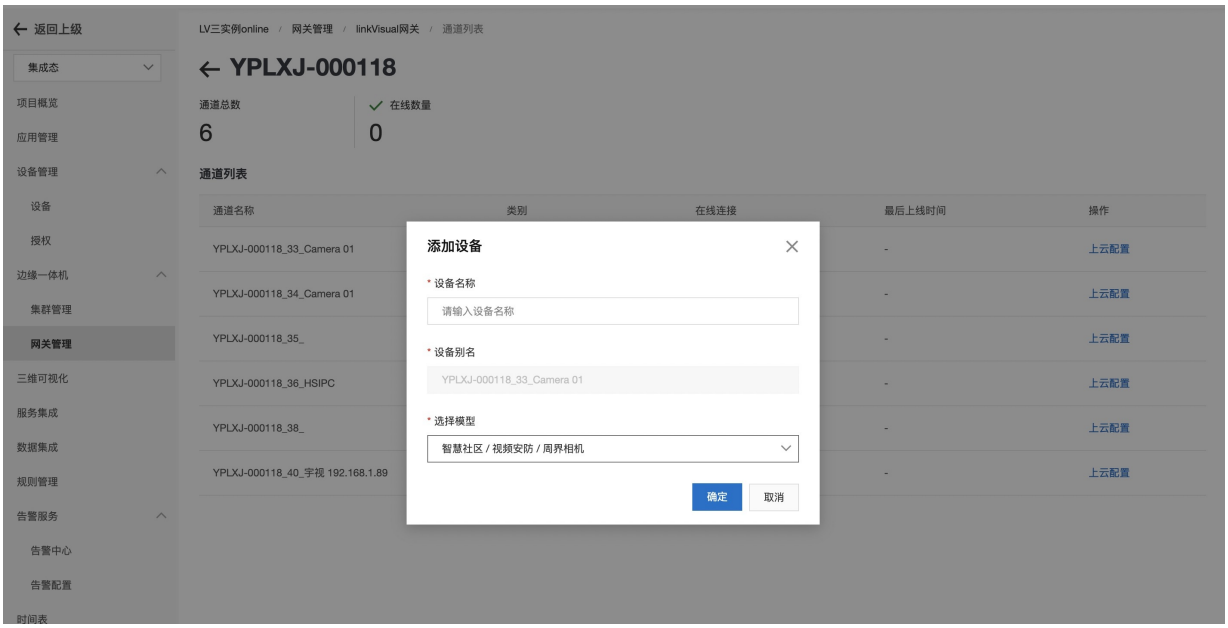


上面显示的就是所有通过某个NVR设备连接的所有IPC摄像头设备。**有NVR接入摄像头**的设备名称由三部分组成：

NVR设备的IoT编码+“有NVR接入摄像头”设备的通道号+“有NVR接入摄像头”设备的空间信息。

例如：**YPLXJ-000118_33_Camera01**：其中NVR设备的IoT编码是**YPLXJ-000118**，“有NVR接入摄像头”设备的通道号是**33**，“有NVR接入摄像头”设备的空间信息是**Camera01**

点击摄像头设备右侧的**上云配置**按钮，弹出配置框，如下图所示

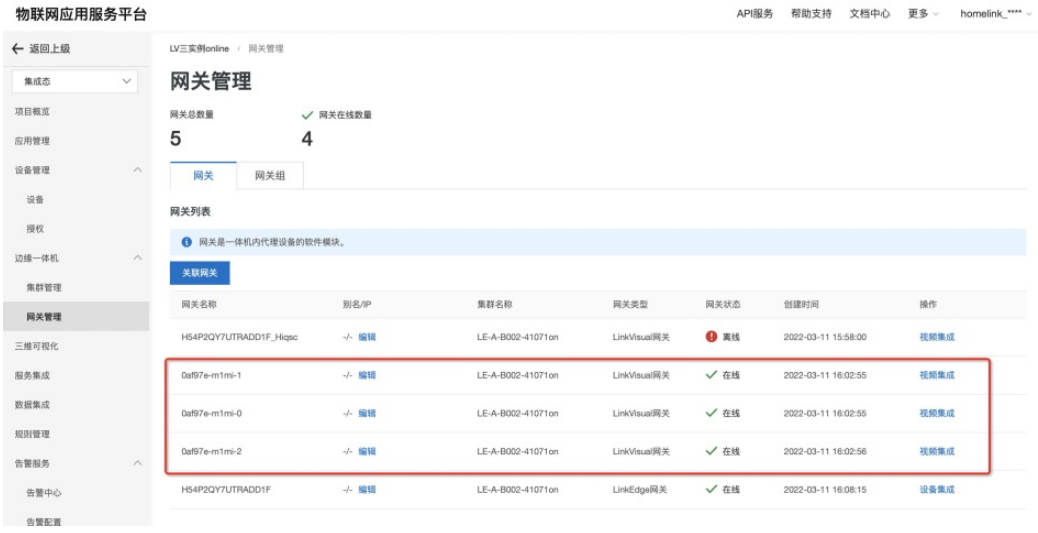


设备名称：填写设备的**IoT编码**，请遵循**IoT编码**规范，并确保IoT编码的唯一性。

选择模型：选择设备的**模型**。与1.1.2节类似，点击选择模型，选择智慧社区 - 视频安防后进行选择， 可以选择的摄像头产品类型共有四种：普通摄像头、违停球相机、人脸抓拍机、周界相机。

1.2 添加设备到第二个/第三个LV边缘网关实例

选择网关名称尾部序号为1，点击右侧的**视频集成**，进入第二个LV边缘网关实例的视频集成页面。剩下操作与2.1节类似



最后一个是选择网关名称尾部序号为2，点击右侧的**视频集成**，完成相关的接入操作。

2 异常情况处理

2.1 更换设备DeviceName流程

如果因为某些原因，需要更换LV边缘视频本地控制台里IPC或NVR设备的DeviceName，处理流程如下：

1、在LV边缘视频云端控制台，将IPC设备删除

2、重新添加设备

2.2 NVR下的IPC摄像头无法自动拉取处理方法

如果NVR设备内网不通且不在线，这时在LV边缘视频的云端控制台添加NVR设备，无法自动拉取NVR下面挂载的IPC摄像头，即使NVR设备内网联通上线后也不行。

出现该问题后的处理方法：在LV边缘视频的云端控制台，按照下面顺序处理：

a. 在设备管理里删除NVR设备，

b. 在设备管理里添加NVR设备，

3.3.5.6. EBA设备接入

3.3.5.6.1. ModbusV2协议接入说明

本章将为您介绍如何通过Modbus协议连接局域网内的设备，ModbusV2和Modbus协议的最大区别为本协议接入的EBA设备不会因为设备扩展协议参数不同而导致创建不同产品

新建分组

1. 登录阿里云应用服务平台，左侧菜单栏中找到边缘接入->品类管理。



2. 点击**品类管理**，选择**自有**。查看是否已创建了**智慧社区**分组，如果未创建则进行创建。



- 分组名称：固定填写：智慧社区 (不可更改)。再次强调，分组名称必须填写：智慧社区
- 编码：固定填写：community（不可更改）。再次强调，编码必须填写：community然后点击确定。

新建品类

在智慧社区下面若没有设施设备，则需要新建品类。新建品类方法如下：把鼠标放到智慧社区右上方会出现个齿轮，把鼠标放到齿轮上，然后会出现新建品类，点击新建品类

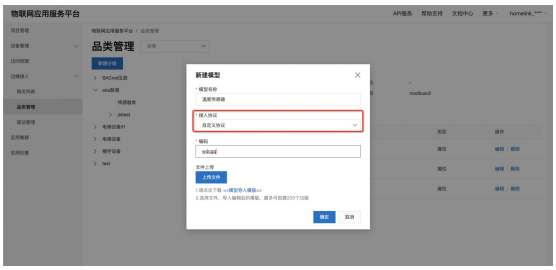


- 品类名称：固定填写：设施设备（不可更改）。再次强调，品类名称必须填写：设施设备
- 编码：固定填写：eba（不可更改）。再次强调，编码必须填写：eba

新建模型

在创建的品类下创建设备模型，把鼠标放到**设施设备**，点击**新建模型**，如下图所示：



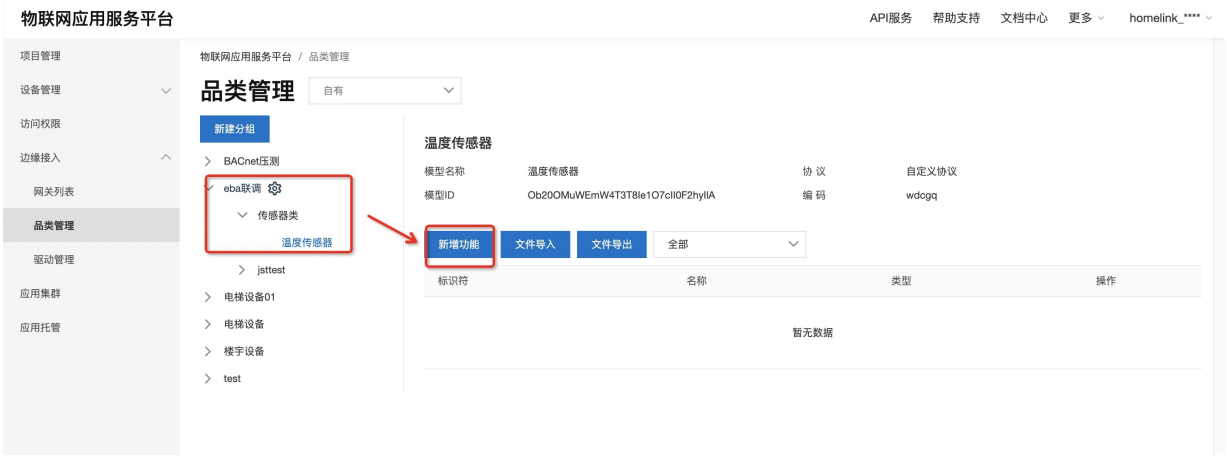


- 模型名称：填写规范：功能名称_厂商名称+序号。功能名称可以参考下面的<EBA新建模型命名规范>，输入示例为：温度控制_霍尼韦尔01。模型名称确定后，注意将<附件2.XX项目-智能化设备安装信息表>的<4.EBA网关（数据采集器）系统> TAB页，将设备(示例：IoT编码为GT-HBA704I-Modbus-003)的模型名称修改为这里定义的模型名称：温度控制_霍尼韦尔01。这样便于后续的项目施工维护
- 接入协议：从下拉列表选择：自定义协议：填写规范：功能名称_厂商。编码填写可以参考下面的<EBA新建模型命名规范>，输入示例为：tpHw01。
- 编码即模型名称对应的英文名称。编码要求支持英文字母、数字和下划线，长度限制1~10。

EBA新建模型命名规范	
模型名称 （填写规范：功能名称_厂商名称+序号） （以下以厂家《霍尼韦尔》作为案例）	编码 （填写规范：功能名称_厂商） （如gdHw01 gd 代表气体传感器，Hw为霍尼韦尔厂家缩写，01 为下标）
气体感测器_霍尼韦尔01	gdHw01
气体感测器_霍尼韦尔02	gdHw02
……	……
风机控制器_霍尼韦尔01	spfHw01
风机控制器_霍尼韦尔02	spfHw02
……	……
照明回路_霍尼韦尔01	lightHw01
照明回路_霍尼韦尔02	lightHw02
……	……
遥信遥控智能仪表_霍尼韦尔01	rciHw01
遥信遥控智能仪表_霍尼韦尔02	rciHw02
……	……
温度传感器_霍尼韦尔01	tempHw01
温度传感器_霍尼韦尔02	tempHw02
……	……
水浸传感器_霍尼韦尔01	wgHw01
水浸传感器_霍尼韦尔02	wgHw02
……	……
温湿度传感器_霍尼韦尔01	thHw01
温湿度传感器_霍尼韦尔02	thHw02
……	……
液位传感器_霍尼韦尔01	lgHw01
液位传感器_霍尼韦尔02	lgHw02
……	……
压力传感器_霍尼韦尔01	psHw01
压力传感器_霍尼韦尔02	psHw02
……	……
智能电力仪表_霍尼韦尔01	pmHw01
智能电力仪表_霍尼韦尔02	pmHw02
……	……
超声波流量计_霍尼韦尔01	fmHw01
超声波流量计_霍尼韦尔02	fmHw02
……	……
温度控制_霍尼韦尔01	tpHw01
温度控制_霍尼韦尔02	tpHw02
……	……

新增功能

为模型添加功能项，功能项输入中功能类型只支持属性，数据类型需要根据点位的实际类型进行选择，同时不要填写扩展参数





说明

- 填写功能类型时，ModbusV2协议只支持属性，不支持服务和事件
- 填写数据类型时，请依据EBA设备厂商提供的每个点位数据类型做选择

设备集成

1. 登录阿里云应用服务平台，找到需要配置的项目，该项目已由项目方创建，如果未创建，请联系项目方。



2. 进入项目内，找到**边缘一体机-网关管理**，如下图所示。点击网关对应的**设备集成**



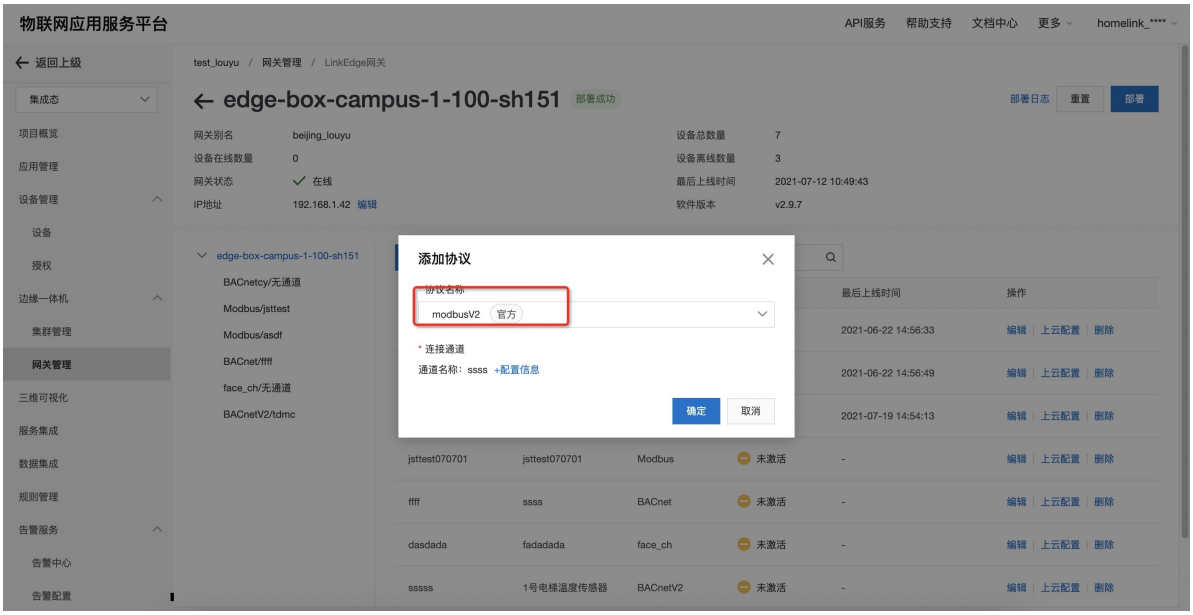
3. 添加ModbusV2协议
   - 首次添加，选择新建协议



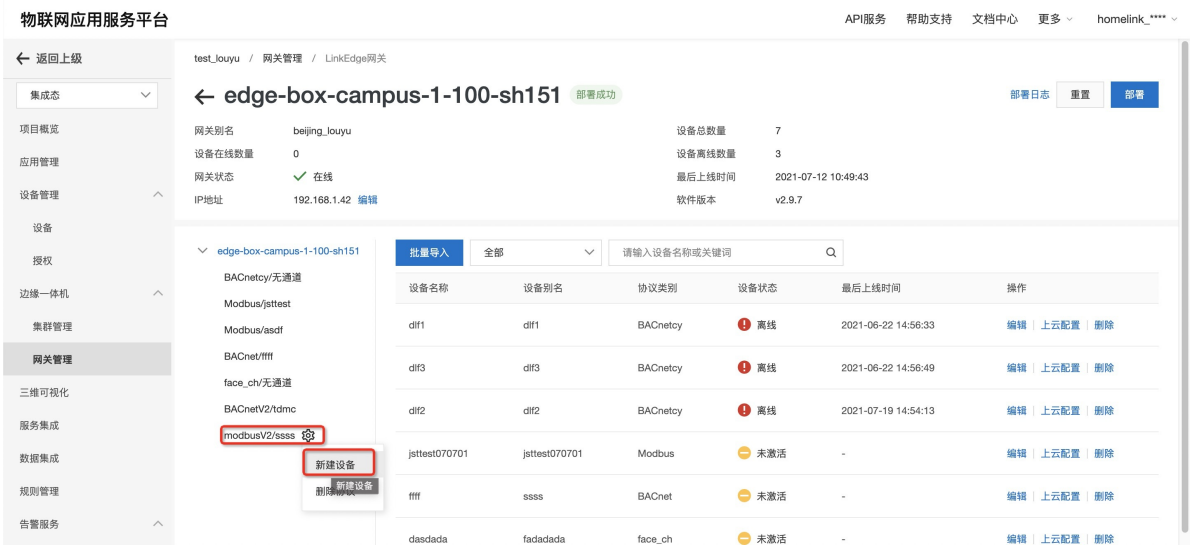
- 网关里已添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



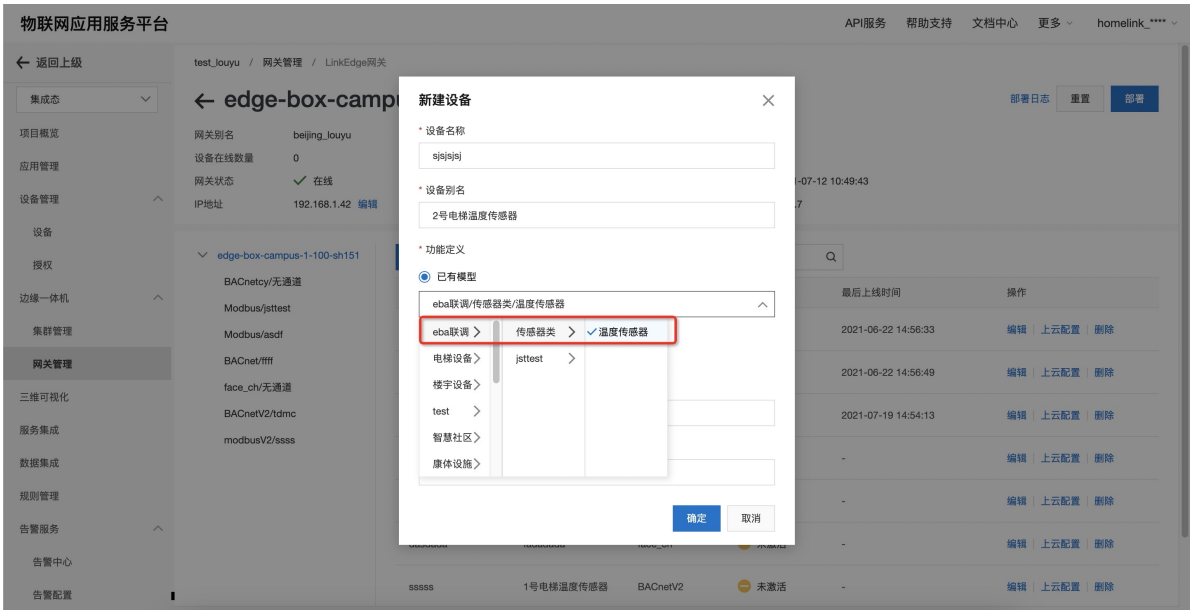
4. 选择ModbusV2的协议，并配置链接通道



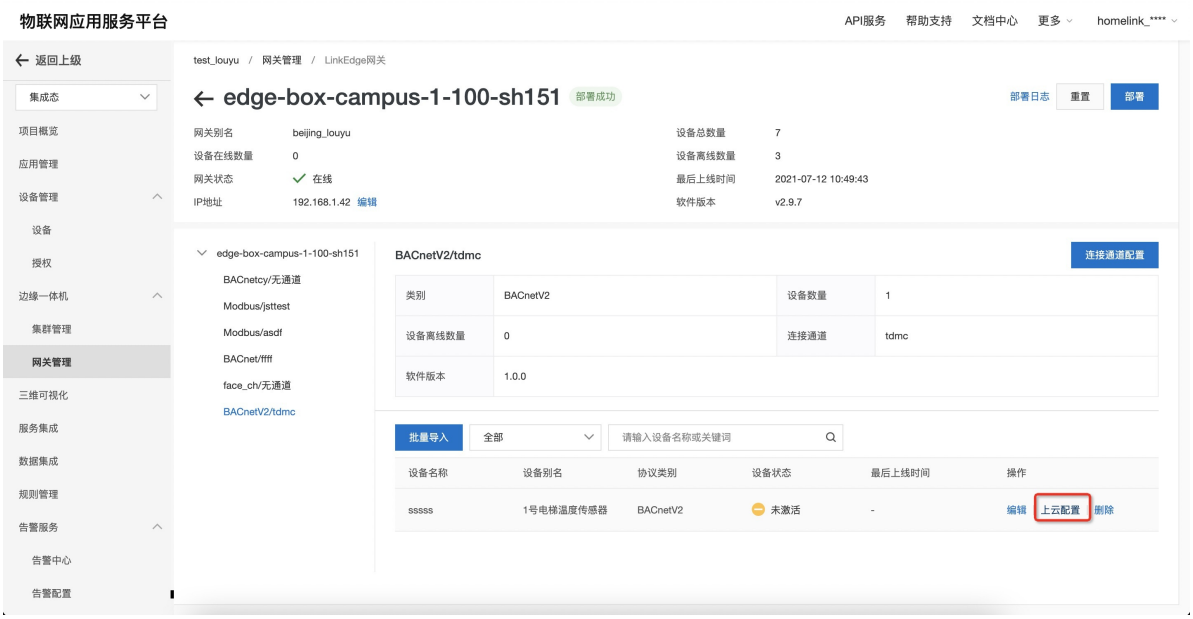
5. 在ModbusV2协议下添加设备



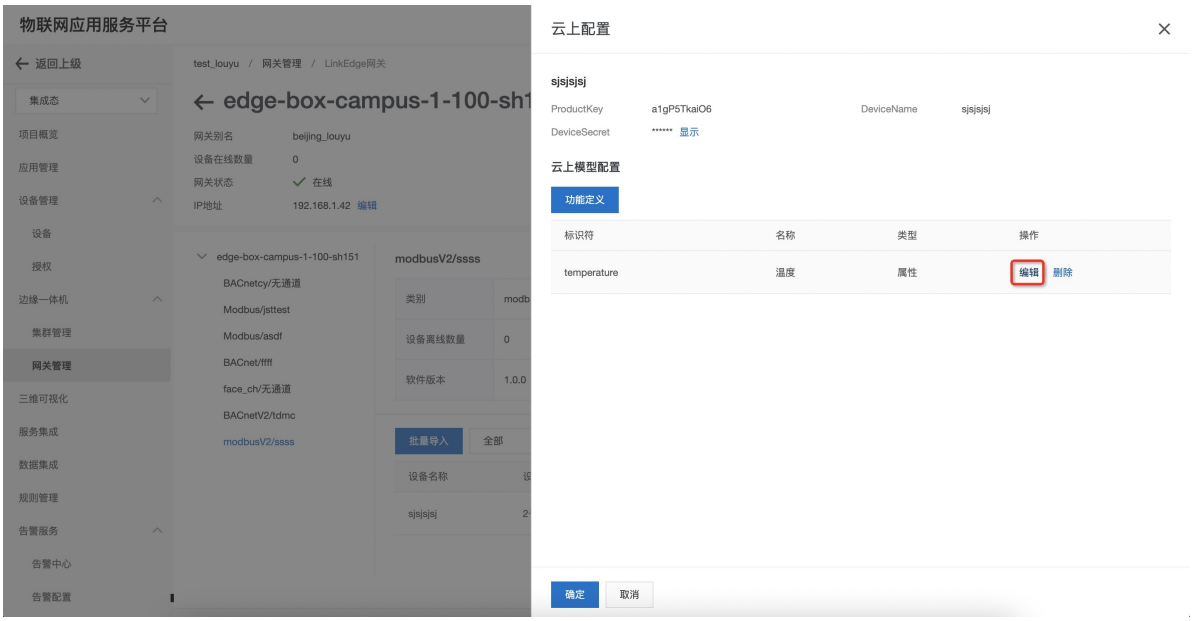
新增设备页面中功能定义选择在品类中创建的模型，并完成协议连接配置相关内容



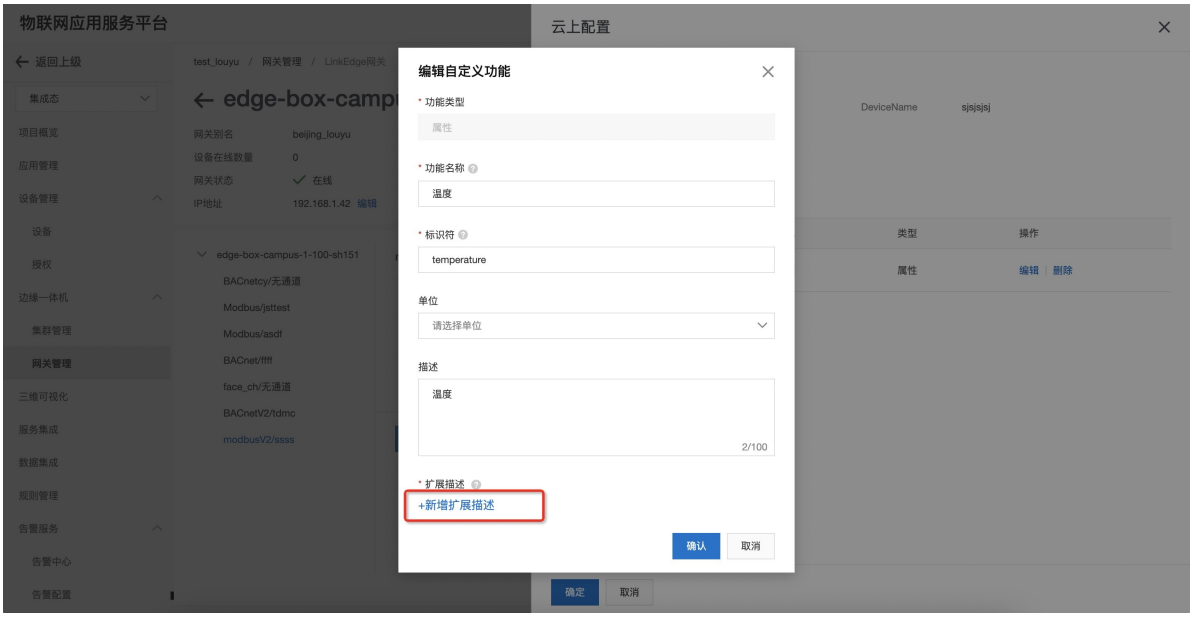
6. 设备创建完成后，进入上云配置模块



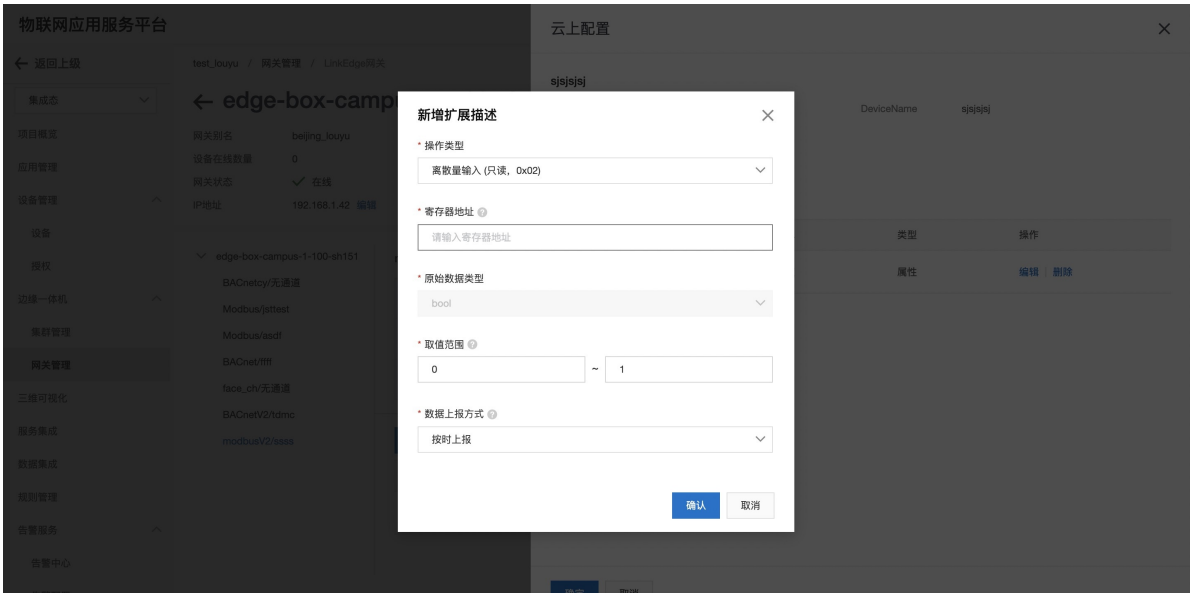
针对每个功能项做编辑



进入功能项的扩展参数



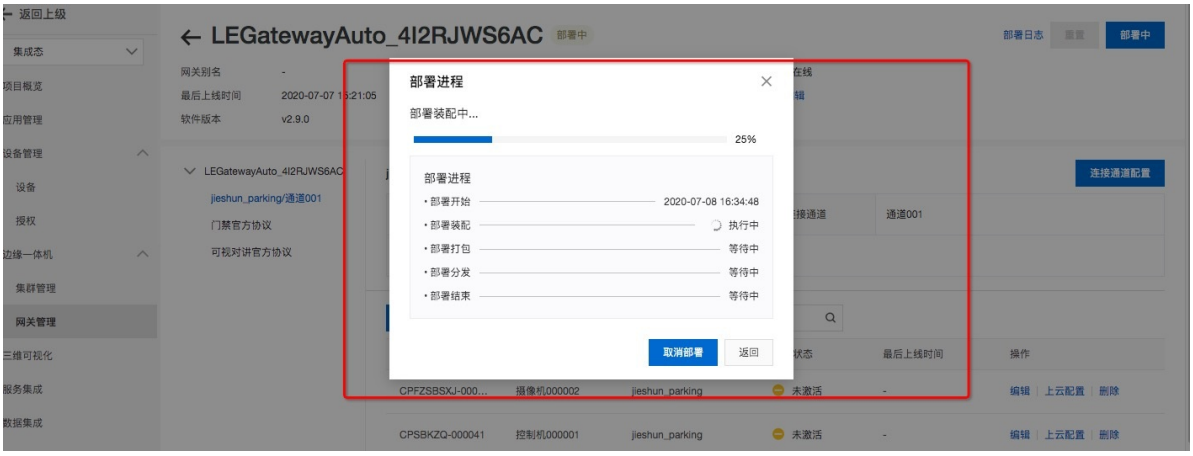
填写当前属性的扩展信息，注意扩展属性中输入的操作类型对应数据类型和品类中定义的数据类型需保持一致



7. 在完成设备的配置后，点击部署，进行设备配置的协议下发到边缘服务器。如下图所示。特别提醒：驱动部署时，网关侧对应驱动将重启，该驱动下的所有设备将出现离线后在线。这期间设备不可用，请部署人员提前与甲方沟通部署时间



点击部署后可以查看设备配置下发的状态和部署进度。部署大概需要5分钟。



说明：如果某个设备有新更新的配置信息，重新点击部署后，只会下发有改动的设备配置协议，其余未修改的配置信息不会再次下发到边缘服务器，部署的时间根据修改的设备数量有关。

3.3.5.6.2. BACnetV2协议接入说明

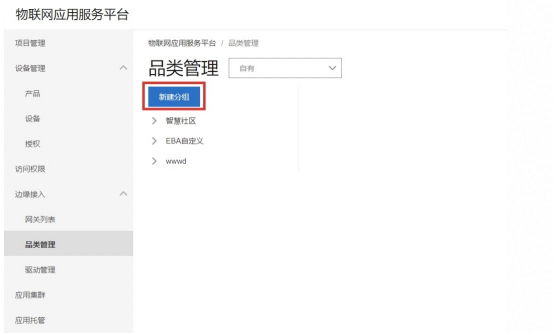
本章将为您介绍如何通过BACnet协议连接局域网内的设备，BACnetV2和BACnet协议的最大区别为本协议接入的EBA设备不会因为设备扩展协议参数不同而导致创建不同产品

新建分组

1. 登录阿里云应用服务平台，左侧菜单栏中找到边缘接入->品类管理。



2. 点击**品类管理**，选择**自有**。查看是否已创建了**智慧社区**分组，如果未创建则进行创建。





- 分组名称：固定填写：智慧社区 (不可更改)。再次强调，分组名称必须填写：智慧社区
- 编码：固定填写：community（不可更改）。再次强调，编码必须填写：community然后点击确定。

新建品类

在智慧社区下面若没有设施设备，则需要新建品类。新建品类方法如下：把鼠标放到智慧社区右上方会出现个齿轮，把鼠标放到齿轮上，然后会出现新建品类，点击新建品类

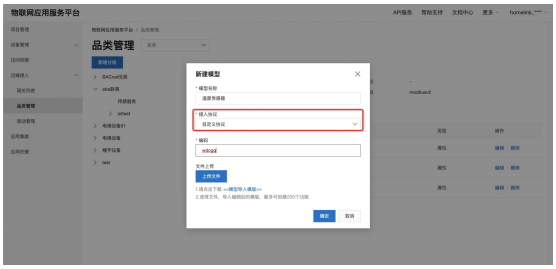


- 品类名称：固定填写：设施设备（不可更改）。再次强调，品类名称必须填写：设施设备
- 编码：固定填写：eba（不可更改）。再次强调，编码必须填写：eba

新建模型

在创建的品类下创建设备模型，把鼠标放到**设施设备**，点击**新建模型**，如下图所示：



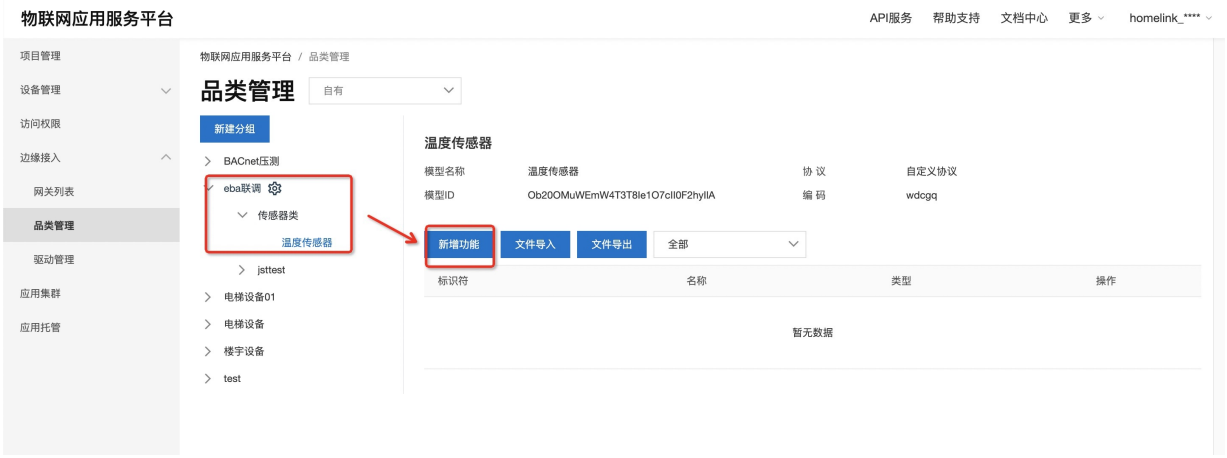


- 模型名称：填写规范：功能名称_厂商名称+序号。功能名称可以参考下面的<EBA新建模型命名规范>，输入示例为：温度控制_霍尼韦尔01。模型名称确定后，注意将<附件2.XX项目-智能化设备安装信息表>的<4.EBA网关（数据采集器）系统> TAB页，将设备(示例：IoT编码为HW-PUC5533-BACnet-001)的模型名称修改为这里定义的模型名称：温度控制_霍尼韦尔01。这样便于后续的项目施工维护
- 接入协议：从下拉列表选择：自定义协议：填写规范：功能名称_厂商。编码填写可以参考下面的<EBA新建模型命名规范>，输入示例为：tpHw01。
- 编码即模型名称对应的英文名称。编码要求支持英文字母、数字和下划线，长度限制1~10。

EBA新建模型命名规范	
模型名称 （填写规范：功能名称_厂商名称+序号） （以下以厂家《霍尼韦尔》作为案例）	编码 （填写规范：功能名称_厂商） （如gdHw01 gd 代表气体传感器，Hw为霍尼韦尔厂家缩写，01 为下标）
气体感测器_霍尼韦尔01	gdHw01
气体感测器_霍尼韦尔02	gdHw02
……	……
风机控制器_霍尼韦尔01	spfHw01
风机控制器_霍尼韦尔02	spfHw02
……	……
照明回路_霍尼韦尔01	lightHw01
照明回路_霍尼韦尔02	lightHw02
……	……
遥信遥控智能仪表_霍尼韦尔01	rciHw01
遥信遥控智能仪表_霍尼韦尔02	rciHw02
……	……
温度传感器_霍尼韦尔01	tempHw01
温度传感器_霍尼韦尔02	tempHw02
……	……
水浸传感器_霍尼韦尔01	wgHw01
水浸传感器_霍尼韦尔02	wgHw02
……	……
温湿度传感器_霍尼韦尔01	thHw01
温湿度传感器_霍尼韦尔02	thHw02
……	……
液位传感器_霍尼韦尔01	lgHw01
液位传感器_霍尼韦尔02	lgHw02
……	……
压力传感器_霍尼韦尔01	psHw01
压力传感器_霍尼韦尔02	psHw02
……	……
智能电力仪表_霍尼韦尔01	pmHw01
智能电力仪表_霍尼韦尔02	pmHw02
……	……
超声波流量计_霍尼韦尔01	fmHw01
超声波流量计_霍尼韦尔02	fmHw02
……	……
温度控制_霍尼韦尔01	tpHw01
温度控制_霍尼韦尔02	tpHw02
……	……

新增功能

为模型添加功能项，功能项输入中功能类型只支持属性，数据类型需要根据点位的实际类型进行选择，同时不要填写扩展参数





说明

- 填写功能类型时，BACnetV2协议只支持属性，不支持服务和事件
- 填写数据类型时，请依据EBA设备厂商提供的每个点位数据类型做选择

设备集成

1. 登录阿里云应用服务平台，找到需要配置的项目，该项目已由项目方创建，如果未创建，请联系项目方。



2. 进入项目内，找到**边缘一体机-网关管理**，如下图所示。点击网关对应的**设备集成**



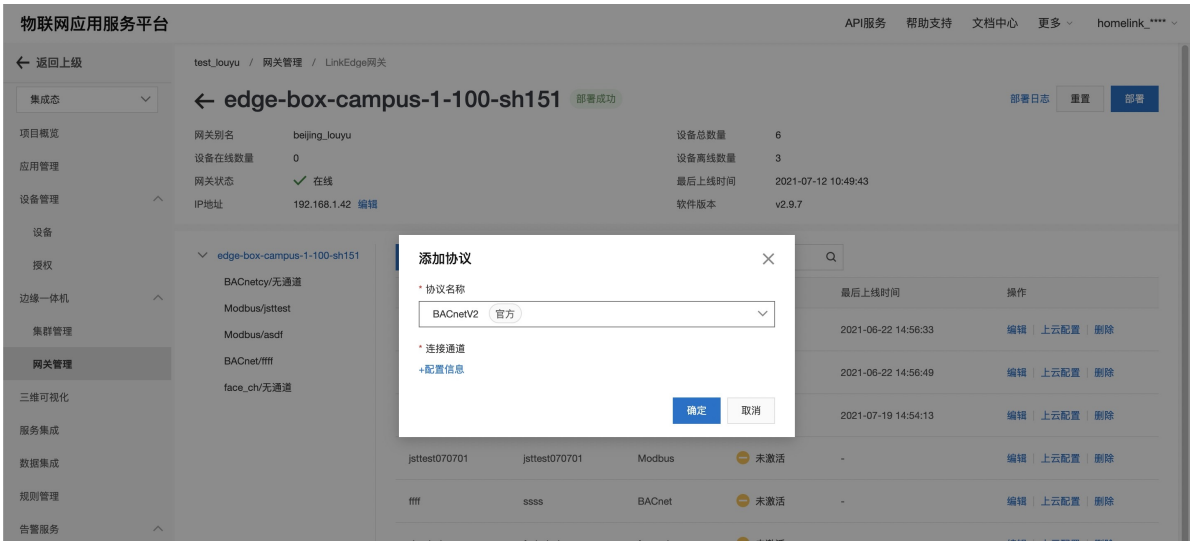
3. 添加BacnetV2协议
   - 首次添加，选择新建协议



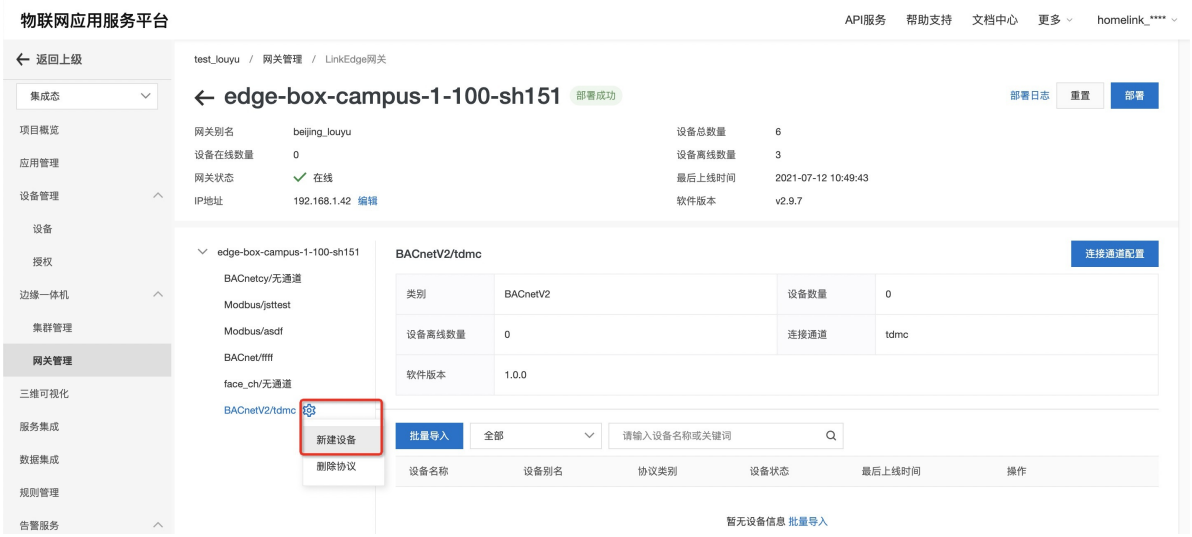
- 网关里已添加过协议，鼠标移动到对应网关右侧的设置图标，点击**添加协议**



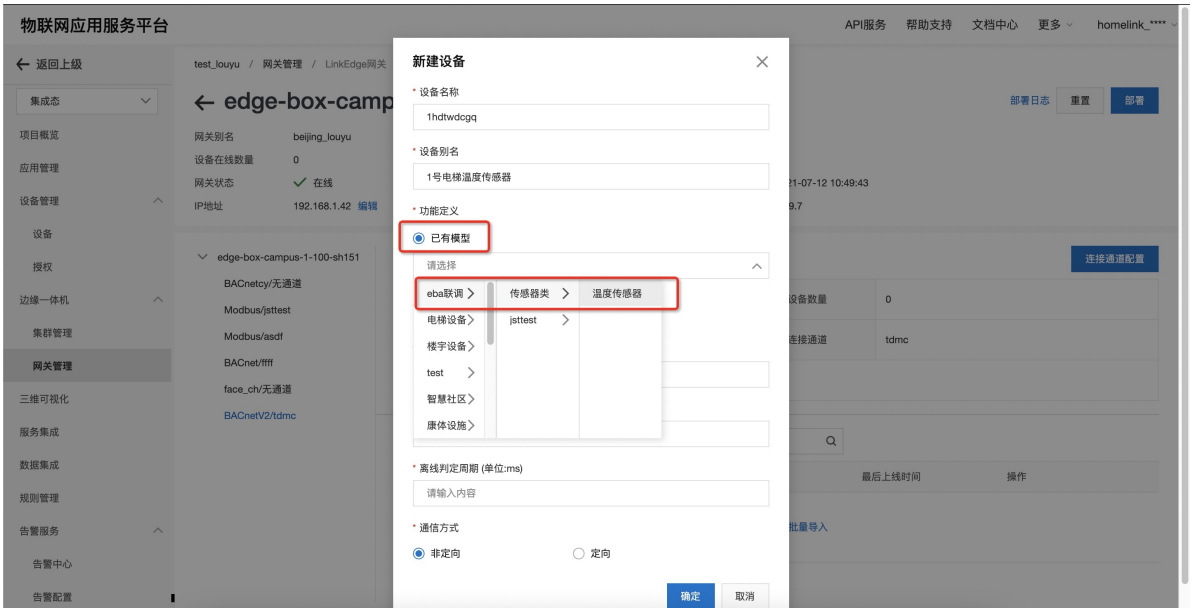
4. 选择BACnetV2的协议，并配置链接通道



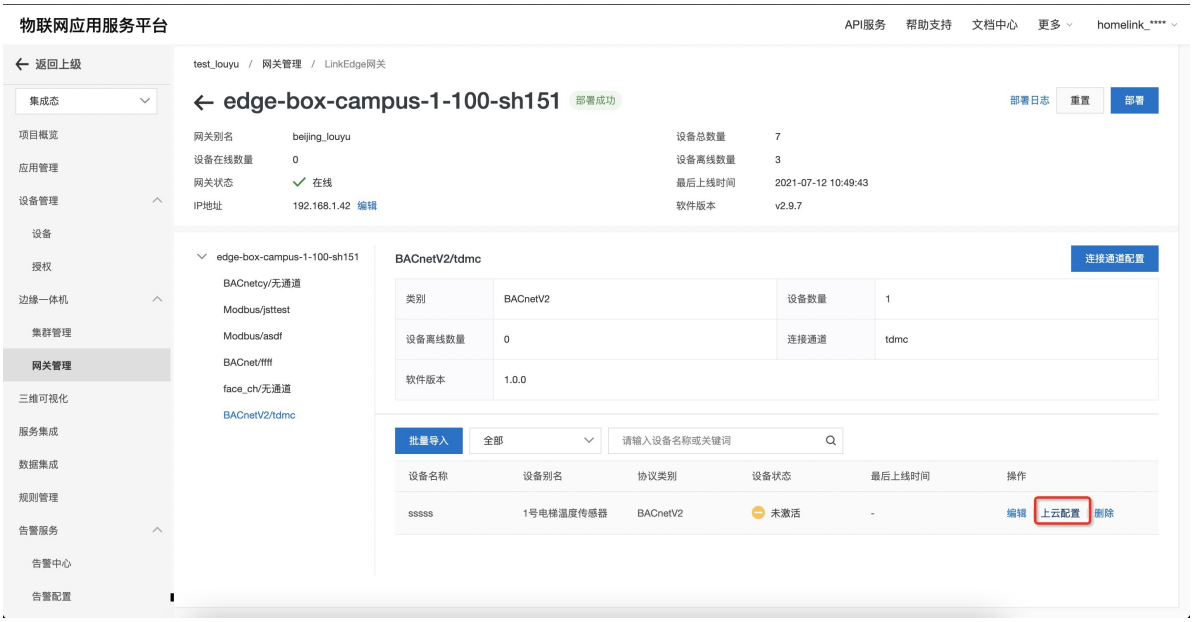
5. 在BACnetV2协议下添加设备



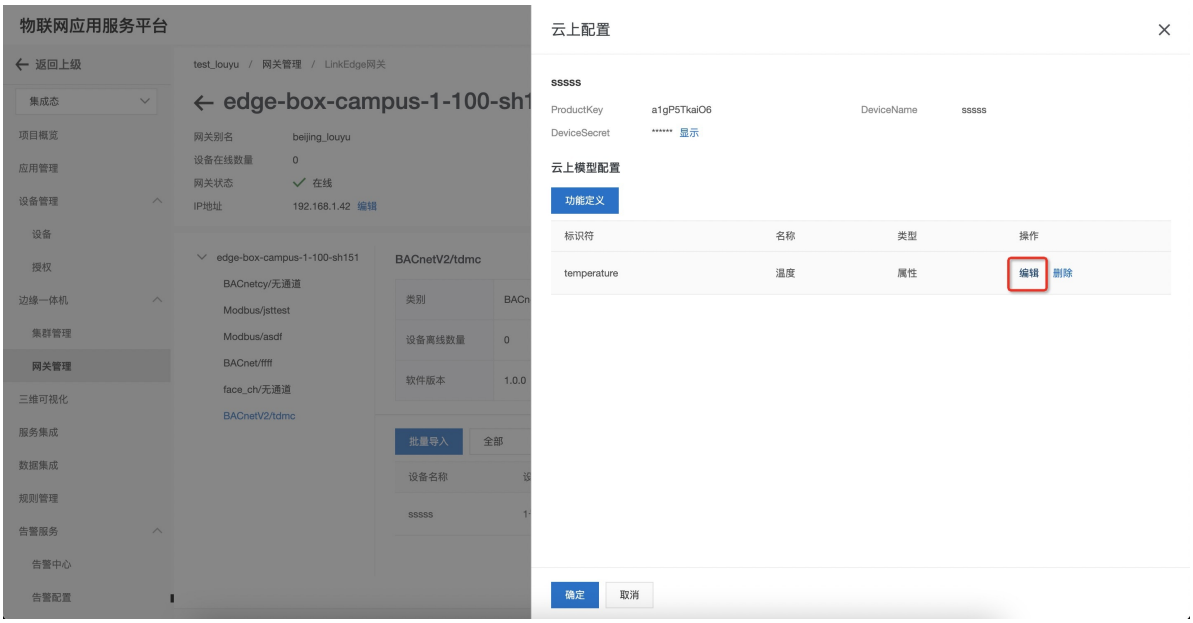
新增设备页面中功能定义选择在品类中创建的模型，并完成协议连接配置相关内容



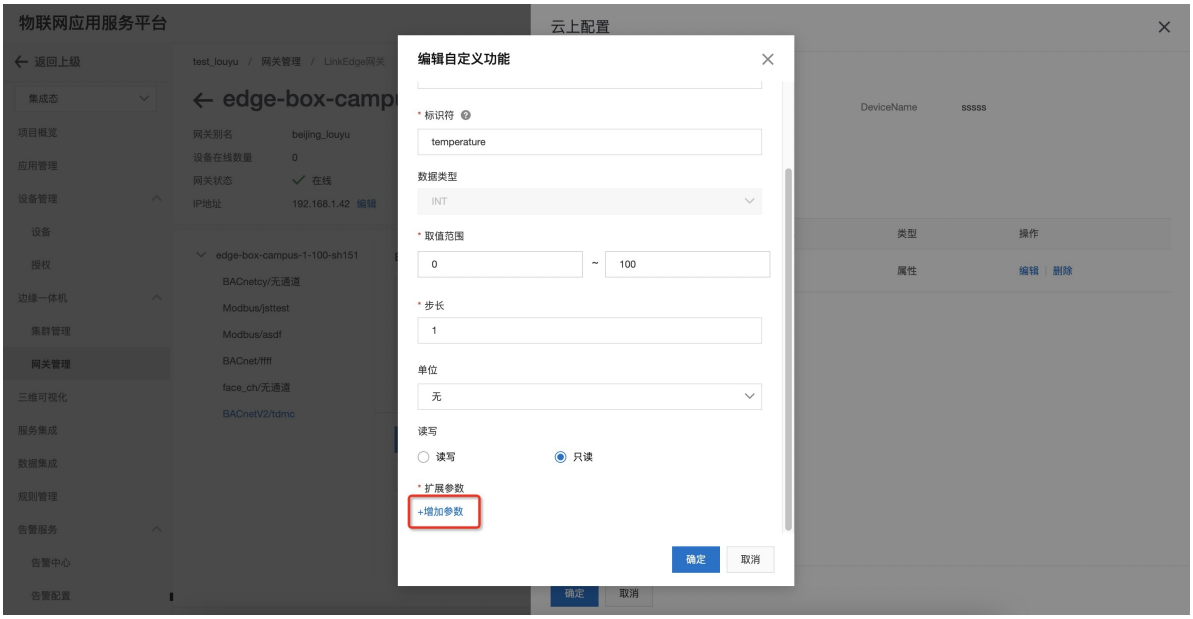
6. 设备创建完成后，进入上云配置模块



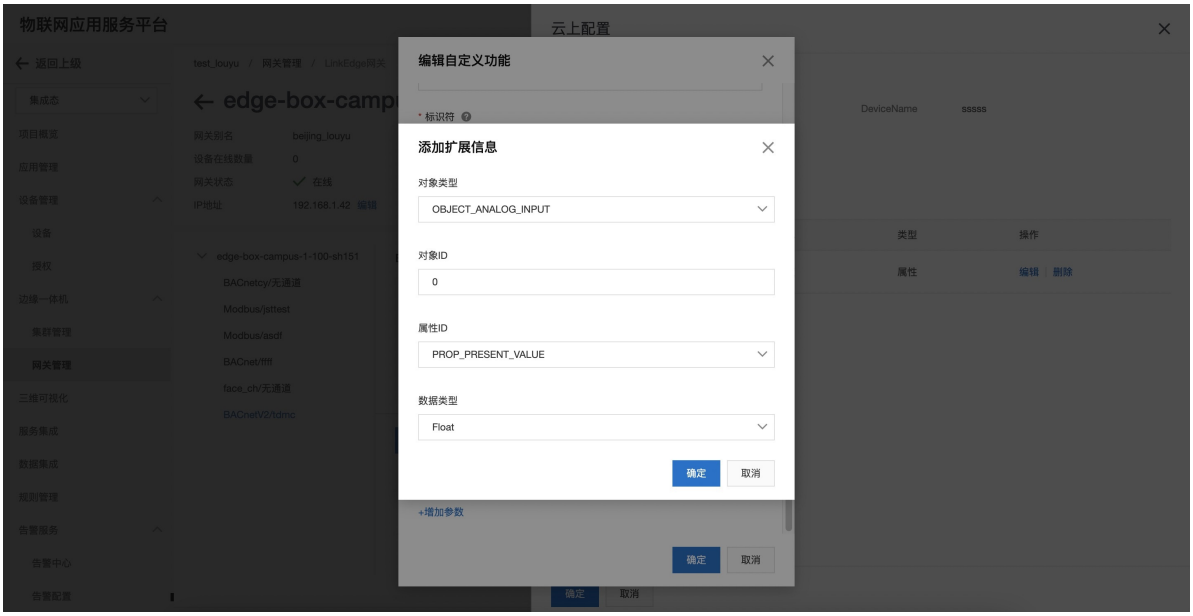
针对每个功能项做编辑



进入功能项的扩展参数



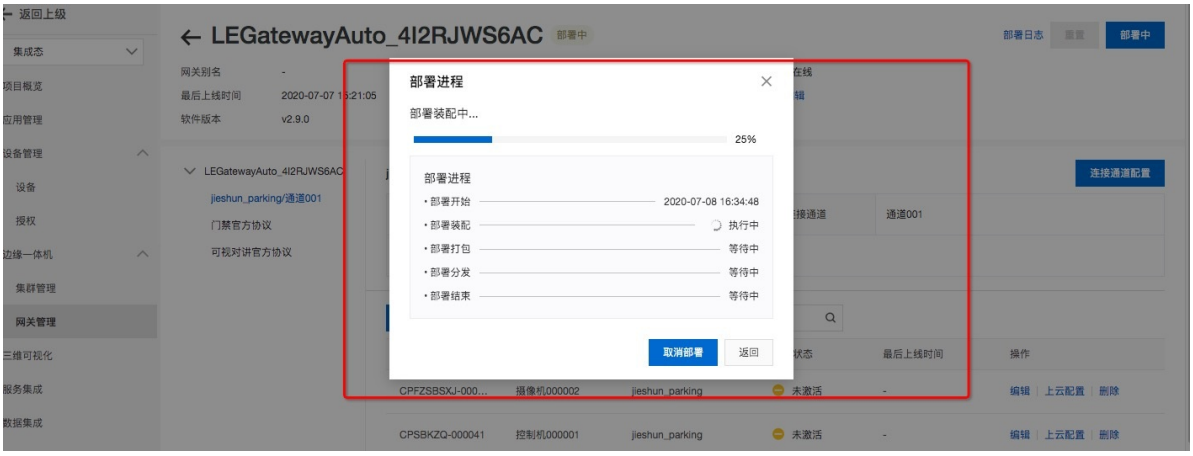
填写当前属性的扩展信息



7. 在完成设备的配置后，点击部署，进行设备配置的协议下发到边缘服务器。如下图所示。特别提醒：驱动部署时，网关侧对应驱动将重启，该驱动下的所有设备将出现离线后在线。这期间设备不可用，请部署人员提前与甲方沟通部署时间



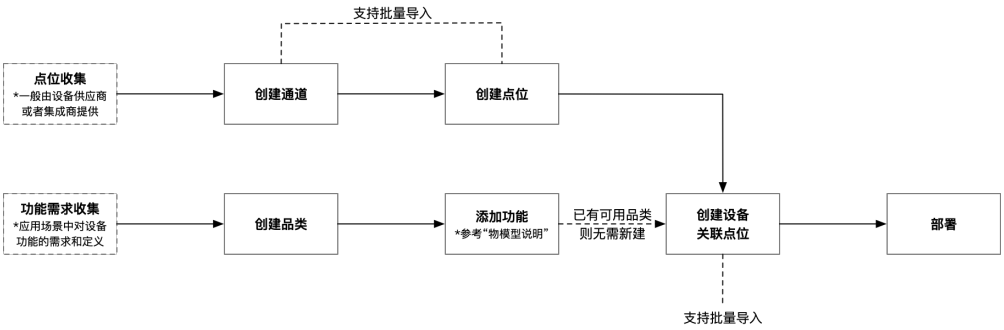
点击部署后可以查看设备配置下发的状态和部署进度。部署大概需要5分钟。



说明：如果某个设备有新更新的配置信息，重新点击部署后，只会下发有改动的设备配置协议，其余未修改的配置信息不会再次下发到边缘服务器，部署的时间根据修改的设备数量有关。

3.3.5.6.3. ModbusV3协议接入说明

1 使用流程说明



提示

1.品类功能定义和设备实际点位无需完全一致

2.点位可以多于功能定义，此时品类功能定义满足业务对数据需求即可，当需要新的点位功能，也可以继续在原有品类新增功能

3.功能定义可以多于设备点位，此时功能定义中多余数据，设备上线后不会更新数据，操作指令也会下发失败

2 添加协议

2.1 选择项目

登录阿里云应用服务平台后，找到需要配置的项目。



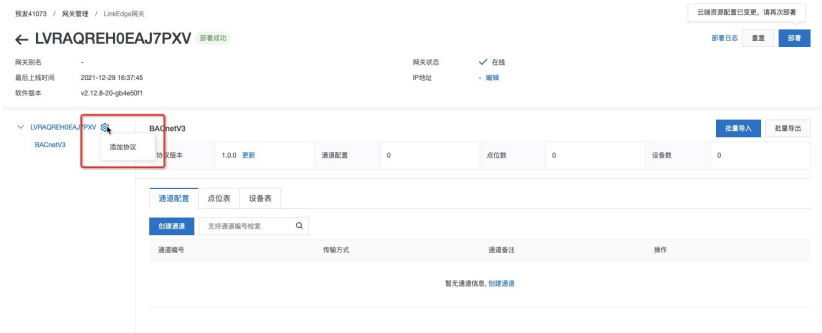
2.2 选择对应网关的设备集成

进入项目内，找到**边缘一体机-网关管理**，如下图所示。点击网关对应的**设备集成**



2.3 添加ModbusV3协议

点击网关右侧的设置按钮中的“添加协议”



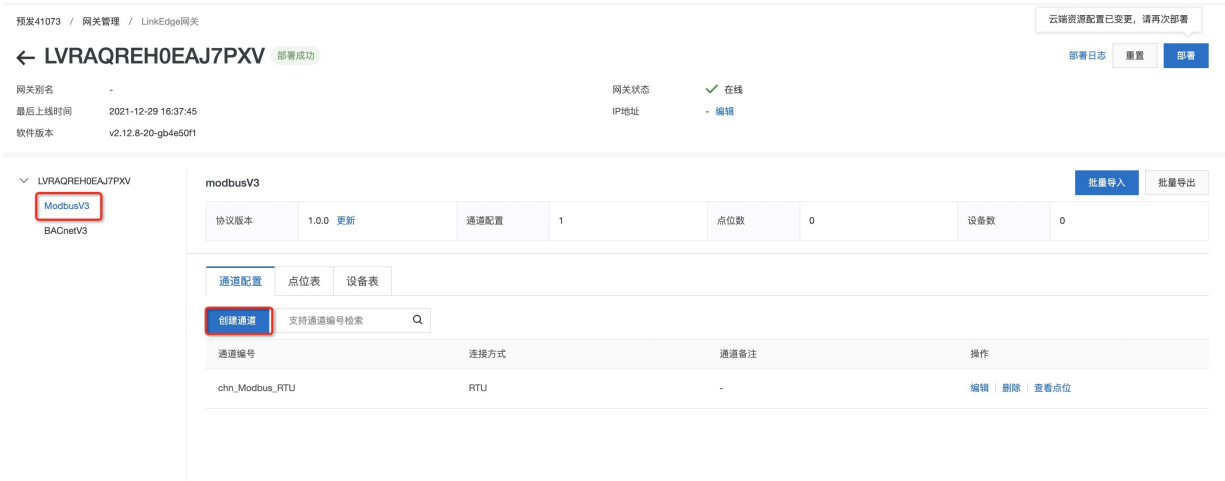
点击网关右侧的设置图标，首先，在弹出菜单中选择**“添加协议”**，然后，在下拉选择的协议列表中选择“**ModbusV3（官方）**”,然后选择协议版本，建议选择**最新版本**进行添加，如下图所示：



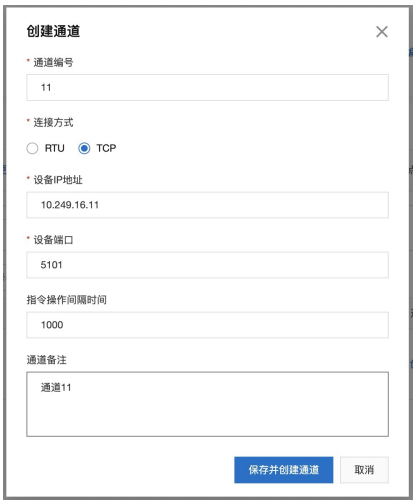
3 配置点位

3.1 添加通道

如下图所示，通过点击“ModbusV3”驱动，右侧展现驱动管理页，进入“通道配置”TAP页，点击“创建通道”按钮



添加配置

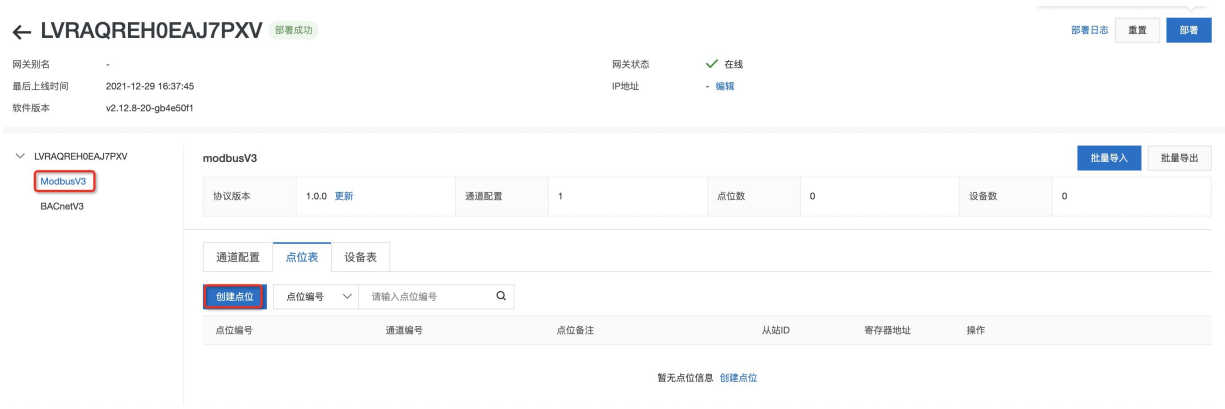


参数说明

名称	说明
通道编号	用于点位与通道关联，自行定义，需要保证协议内唯一性
传输方式	默认选择“TCP”
IP地址	设备端IP地址
端口号	设备端端口号
指令操作间隔时间（单位ms）	设备限制的最小指令操作间隔，设备没有限制时可填0
通道备注	自定义一个备注信息，方便维护

3.2 添加点位

如下图所示，通过点击“ModbusV3”驱动，右侧展现驱动管理页，进入“点位表”TAP页，点击“创建点位”按钮



点位的参数请按下图所示进行配置：



参数说明

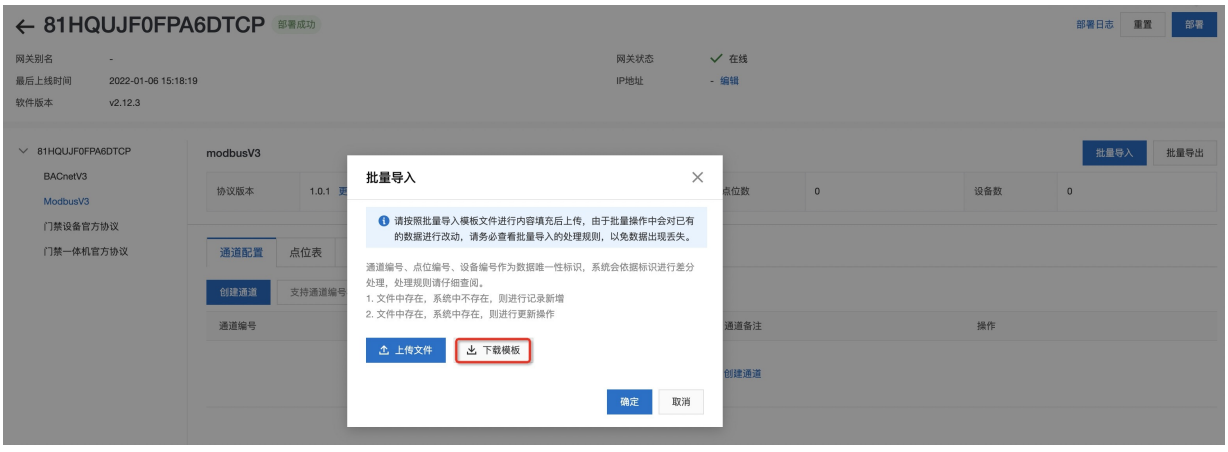
名称	说明
连接通道	点位连接通道

点位编号	点位编号，全局唯一
从站ID	设备从站号
寄存器地址	点位寄存器地址，十六进制
寄存器操作类型	操作点位寄存器的类型功能码
数据类型	点位数据类型，操作类型为“线圈状态”系列和“离散量输入”时，数据类型只支持bool；操作类型为“保持寄存器”系列和“输入寄存器”时，数据类型支持：int16、uint16、int32、uint32、int64、uint64、float、double、string、bool、custom、bits
比特位置	点位数据为采样数据的指定比特位置，数据类型为bits有效
寄存器个数	点位的寄存器个数，数据类型为custom和string时有效
是否交换寄存器内16位数据的高低字节	是否交换16位数据的高低字节序，数据类型为int16、uint16、int32、uint32、int64、uint64、float、double时有效。例：原数据：0xAB，真实数据，0xBA
是否交换寄存器内32位数据顺序	是否交换32位数据顺序，数据类型为int32、uint32、int64、uint64、float、double时有效。例：原数据：0xABCD，真实数据，0xCDAB
缩放因子	点位真实数据与采集数据相比的缩放系数，不能为0，数据类型为int32、uint32、int64、uint64、float、double时有效
数据采样周期	点位数据的采样时间间隔，单位ms
数据上报方式	点位数据的上报方式，支持安时上报和变更上报
点位备注	点位的备注信息

3.3 批量导入

3.3.1 下载批量导入模版

首先下载批量导入模版，如下图所示：



3.3.2 导入模版填入通道信息

打开导入模版，进入**“通道配置表”**TAP页，基于甲方提供的<IoT设备信息表>中的<modbusV3通道信息>TAB页的通道列表填入所有通道配置，如下图所示：

连接编号（单项目单协议唯一）	连接方式（TCP或RTU）	指令最小间隔时间（ms）	设备IP（连接方式为TCP时填写）	设备端口（连接方式为TCP时填写）	串口名（连接方式为RTU时填写）	串口波特率（连接方式为RTU时填写）	数据位（连接方式为RTU时填写）	校验方式（连接方式为RTU时填写，0：无校验，1：奇校验，2：偶校验）	停止位（连接方式为RTU时填写）
11	TCP	1000	10.249.16.11	5101					
15	RTU	1000			/dev/ttyS0	9600	8	0	1

通道配置表 | 点位表 | 设备表

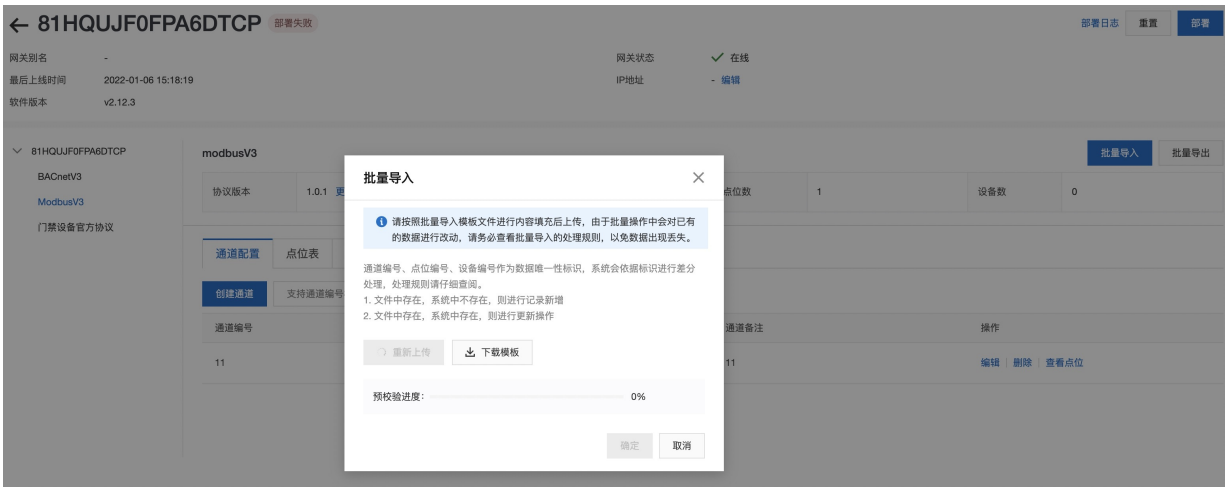
3.3.3 导入模版中填入点位信息

进入“**点位表**”TAP页，如下图所示：

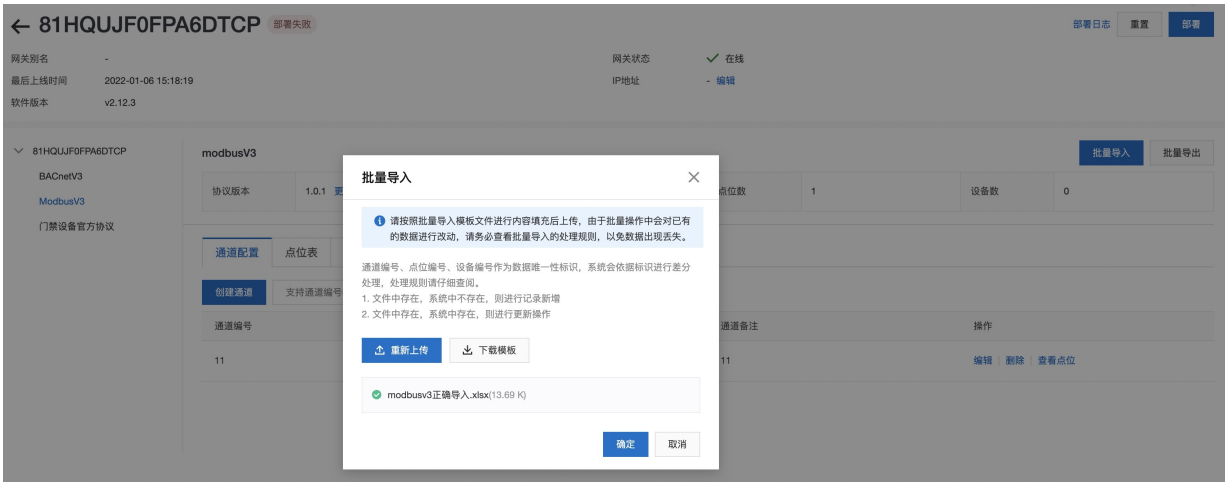
连接通道（保持和通道信息表一致）	点位编号	从站id	寄存器地址（十六进制，0x开头）	寄存器操作类型（操作Modbus的功能码）	数据类型	是否交换寄存器内16位数据的高低字节（是填1，否填0）	是否交换寄存器内32位数据顺序（是填1，否填0）	缩放因子（真实数据跟采集数据相比的缩放系数，不能为0，默认为1）	上报周期（ms）	数据上报方式（按时上报填1，变更上报填2）	寄存器个数（数据类型为custom或string时设置）	比特位置（数据类型为bits时必填）	备注

3.3.4 导入操作

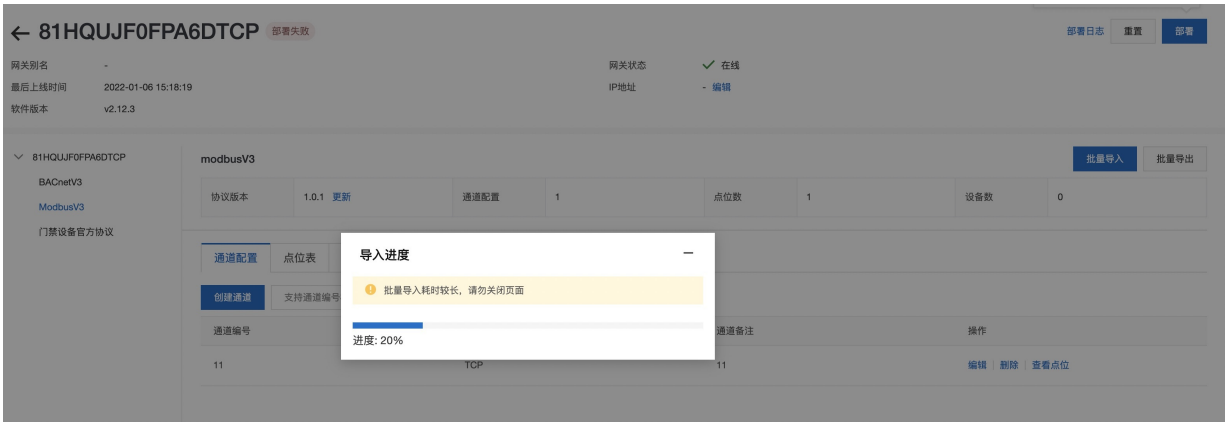
基于导入模版填充好所需导入的配置信息后，点击上传文件，选择所需导入的文件后点击确定按钮，首先会基上传的文件进行预校验，预校验过程主要校验配置内容的合法性，预校验过程中确定按钮为不可点击状态，如下图所示



预校验通过后，确定按钮变为可点击状态，点击后进行实际配置导入，如下图所示：

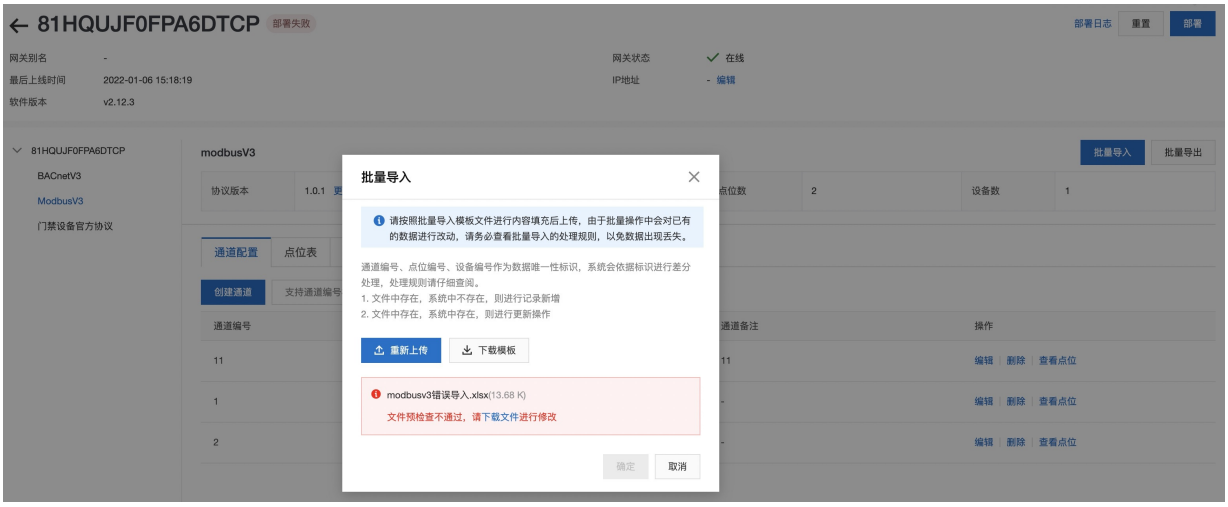


整个导入过程可能需要耗费几分钟时间，请耐心等候（具体由所需导入的点位数），导入进度的展现如下图所示：



3.3.5 预检查失败情况处理

当配置数据存在不合法的情况，预校验会不通过，可下载文件进行失败原因的查看，可基于失败原因对导入文件进行修改，入下图所示：



通道配置表预校验失败原因查看，如下图所示：



连接编号（单项目单协议唯一）	连接方式（TCP或RTU）	设备IP（连接方式为TCP时填写）	设备端口（连接方式为TCP时填写）	串口名（连接方式为RTU时填写）	串口波特率（连接方式为RTU时填写）	数据位（连接方式为RTU时填写）	校验方式（连接方式为RTU时填写）	指令最小间隔时间	停止位（连接方式为RTU时填写）	错误信息
1	TCP	127.9.9.1	80					1000		该通道编号：1重复
1	RTU	127.9.9.1		/dev/a	100	1	1	100	1	该通道编号：1重复

点位表预校验失败原因查看，如下图所示：



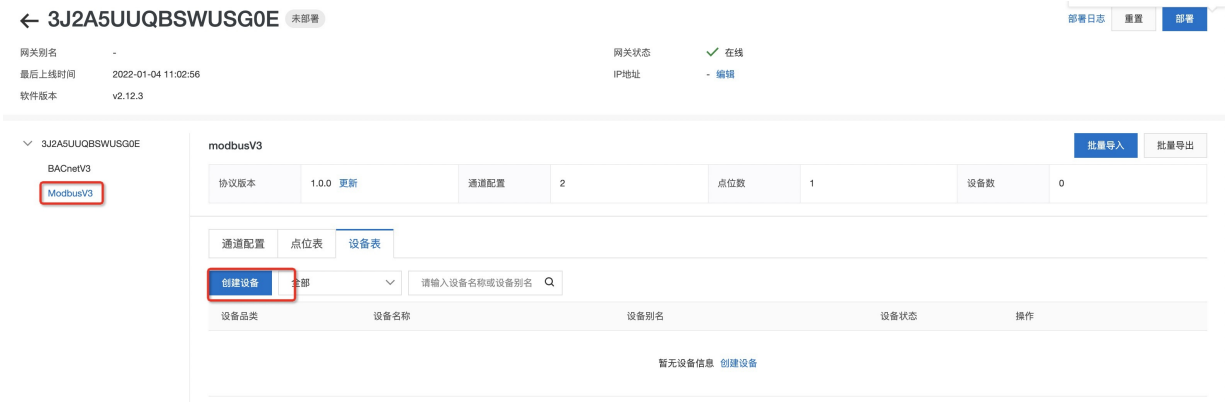
连接通道（保持和通道配置表一致）	点位编号	从站id	寄存器地址（十六进制，0x开头）	寄存器操作类型（操作Modbus的功能码）	数据类型（Data Type	是否交换寄存器内16位数据的高低字节（是填1，否填0）	是否交换寄存器内32位数据顺序	缩放因子（真实数据跟采集数据相比的缩放系数，不能为0，默认为1）	上报周期（ms）	数据上报方式（按时上报填1，变更上报填2）	寄存器个数	位掩码	备注信息	错误信息
100	1	10	0x08	线圈状态（只读，0x01）	int16	1	1	1	1000		10		点位1	通道：100不存在

通道配置表 点位表 设备表

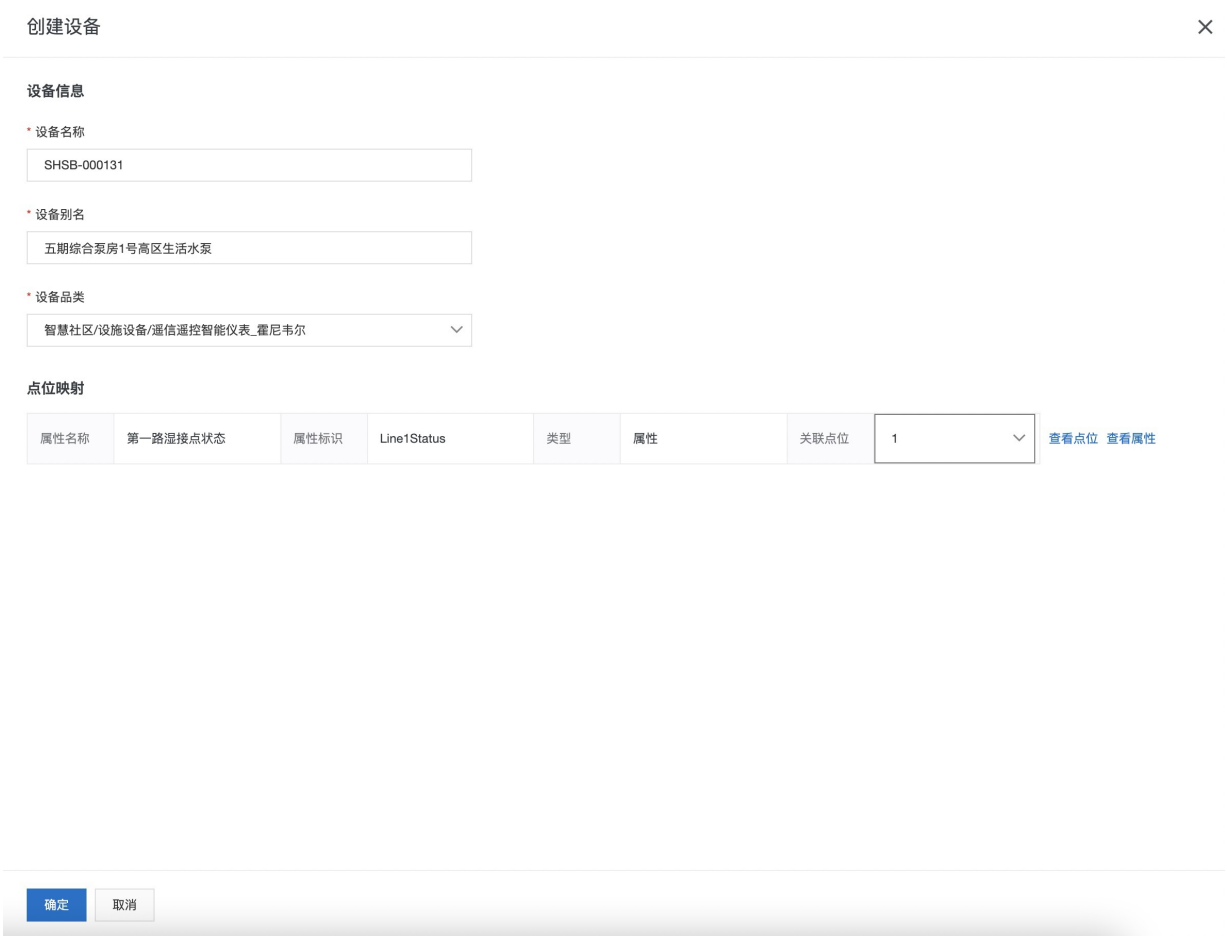
4 添加设备

4.1 设备创建

如下图所示，通过点击**“ModbusV3”**驱动，右侧展现驱动管理页，进入**“设备表”**TAP页，点击**“创建设备”**按钮



设备的参数请按下图所示进行配置：



参数说明

名称	说明
设备名称	设备的IoT编码
设备别名	设备的备注名称
设备品类	设备的物模型品类

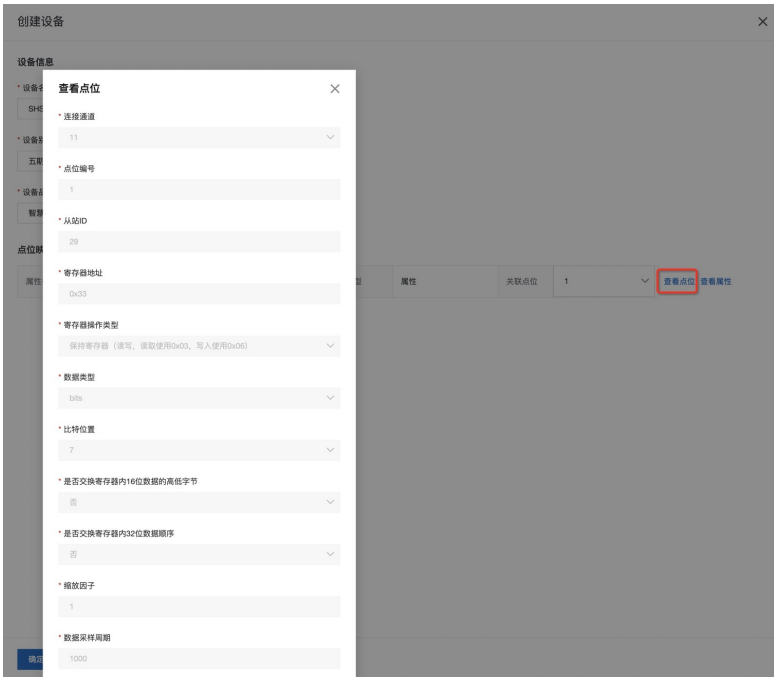
4.2 点位映射

点击下拉列表可以进行点位的选择

点位映射

属性名称	第一路湿接点状态	属性标识	Line1Status	类型	属性	关联点位	1	查看点位 查看属性

点击查看点位可以查询选择的点位的详情



4.3 批量导入

4.3.1 基于导入模版中录入设备信息

查看设备对应的品类信息，本例为遥信遥控智能仪表，在品类管理页找到该品类，导出设备模版，如下图所示：



打开导出的模版，红框部分为该品类所具备的功能属性定义，需要拷贝至导入模版<设备表>TAP页，如下图所示：



然后在导入模版<设备表>TAP页补充**IoT设备ID**，**IoT设备中文名称**，**点位编号**信息



参数说明

名称	说明
IoT设备ID	设备的IoT编码
IoT设备中文名称	设备的备注名称
点位编号	设备的点位编号，全局唯一

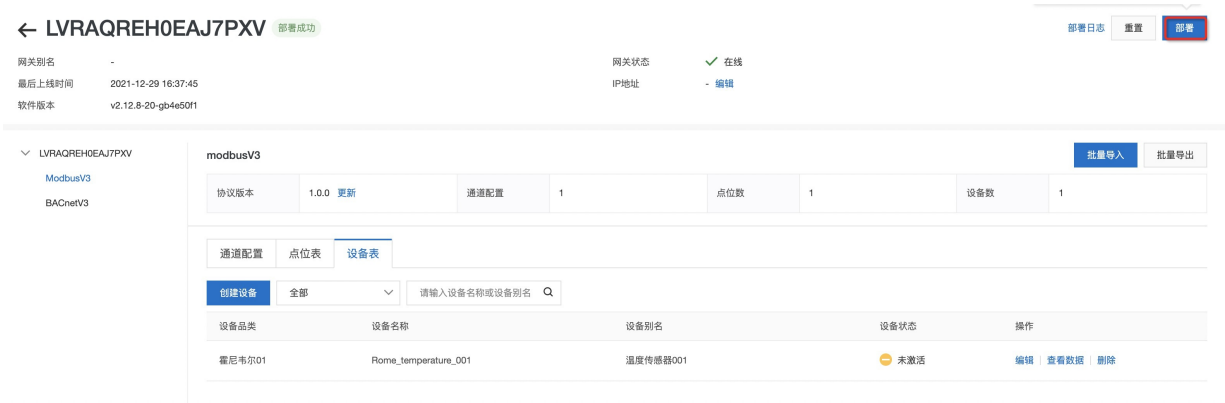
设备表预校验失败原因查看，如下图所示：



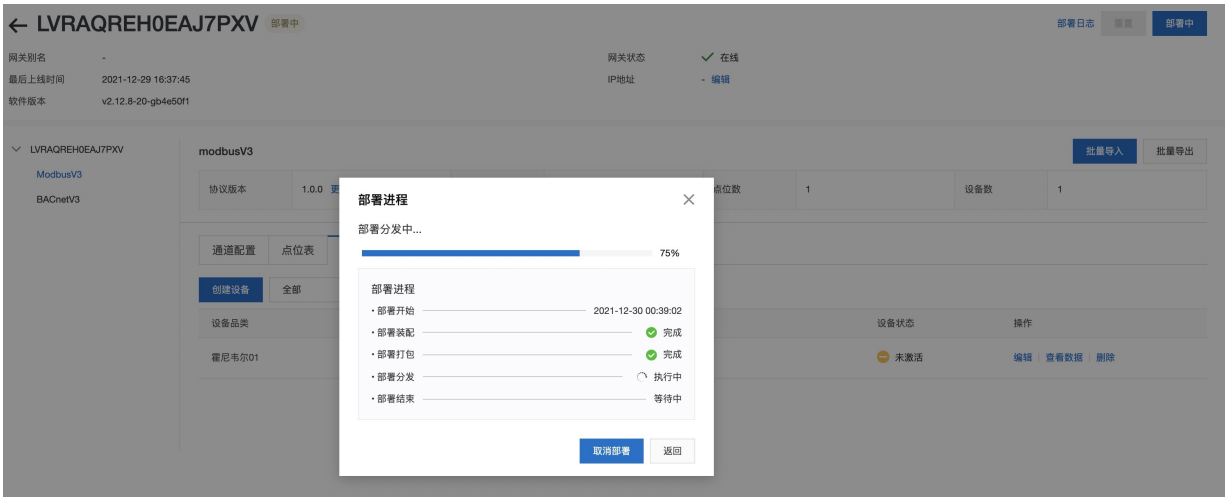
4.4 部署设备配置

在完成驱动和设备的配置后，点击部署，进行设备配置的协议下发到边缘服务器。如下图所示。

特别提醒：驱动部署时，网关侧对应驱动将重启，该驱动下的所有设备将出现离线后在线。这期间设备不可用，请部署人员提前与甲方沟通部署时间



点击部署后可以查看设备配置下发的状态和部署进度。



5 新增品类

5.1 新建分组



弹出页面如下图所示：



参数说明

名称	说明
分组名称	品类分组名称
编码	品类分组名称对应的英文名称。编码要求支持英文字母、数字和下划线，长度限制1~10

5.2 新建品类

把鼠标放到**对应分组**右上方会出现个齿轮，把鼠标放到齿轮上，然后会出现**新建品类**，点击**新建品类**，如下图所示：



弹出页面如下图所示：



参数说明

名称	说明
品类名称	品类名称
编码	品类名称对应的英文名称。编码要求支持英文字母、数字和下划线，长度限制1~10

5.3 新建模型

把鼠标放到**对应分组**，点击**新建模型**，如下图所示：



弹出页面，如下图所示：



参数说明

名称	说明
模型名称	品类的模型名称
接入协议	该模型的设备接入协议，固定为“自定义协议”
编码	模型名称对应的英文名称。编码要求支持英文字母、数字和下划线，长度限制1~10

5.4 新增功能

选择5.3里新建模型遥信遥控智能仪表_霍尼韦尔，点击**新增功能**，进行点位添加，如下图所示：



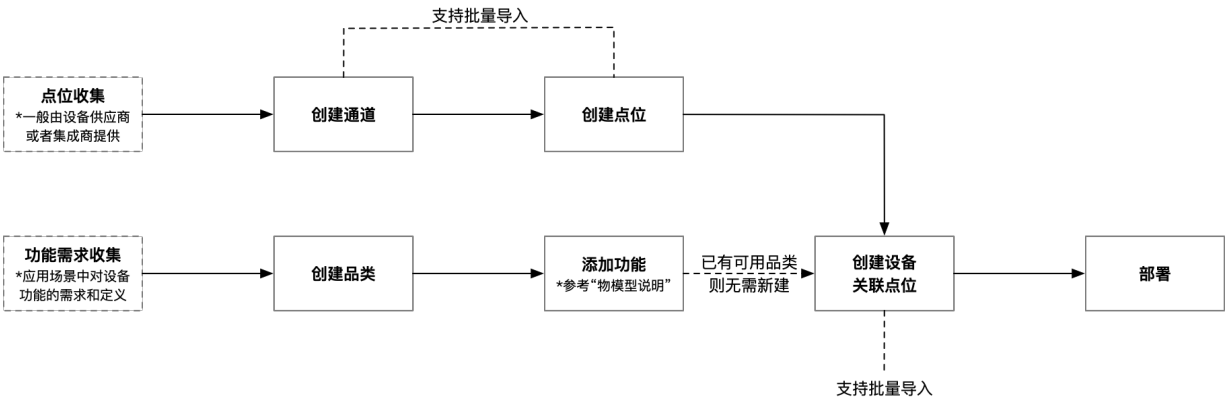
参数说明

名称	说明

功能名称	该功能的功能名称
标识符	功能标识符，用于标识该功能
数据类型	数据类型包括：int（原生）、float（原生）、double（原生）、text（原生）、date（String类型UTC毫秒）、bool（0或1的int类型）、enum（int类型，枚举项定义方法与bool类型定义0和1的值方法相同）、struct（结构体类型，可包含前面7种类型，下面使用"specs":[{}]描述包含的对象）、array（数组类型，支持int、double、float、text、struct）
取值范围	数据的取值范围，超出取值范围的数据会被丢弃，数据类型为int、float、double有效
步长	数据变化的最小粒度，数据类型为int、float、double有效
单位	根据实际情况选择，可选无，数据类型为int、float、double有效
数据长度	最大10240，数据类型为text有效
元素个数	最大512，数据类型为array有效
布尔值	布尔值（0/1）对应的参数描述，数据类型为bool有效
枚举项	枚举项对应的参数描述，数据类型为enum有效
读写类型	数据的读写类型，读写或只读

3.3.5.6.4. BACnetV3协议接入说明

1 使用流程说明



提示

1.品类功能定义和设备实际点位无需完全一致

2.点位可以多于功能定义，此时品类功能定义满足业务对数据需求即可，当需要新的点位功能，也可以继续在原有品类新增功能

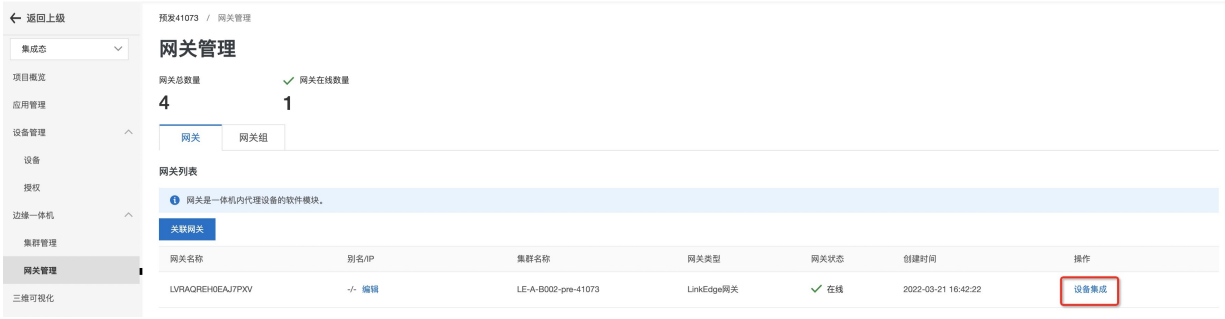
3.功能定义可以多于设备点位，此时功能定义中多余数据，设备上线后不会更新数据，操作指令也会下发失败

2 添加协议

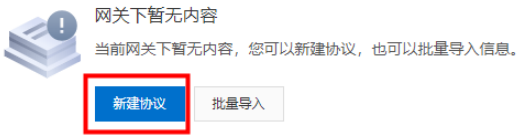
登录阿里云应用服务平台后，找到需要配置的项目，该项目已由项目方创建，如果未创建，请联系项目方。



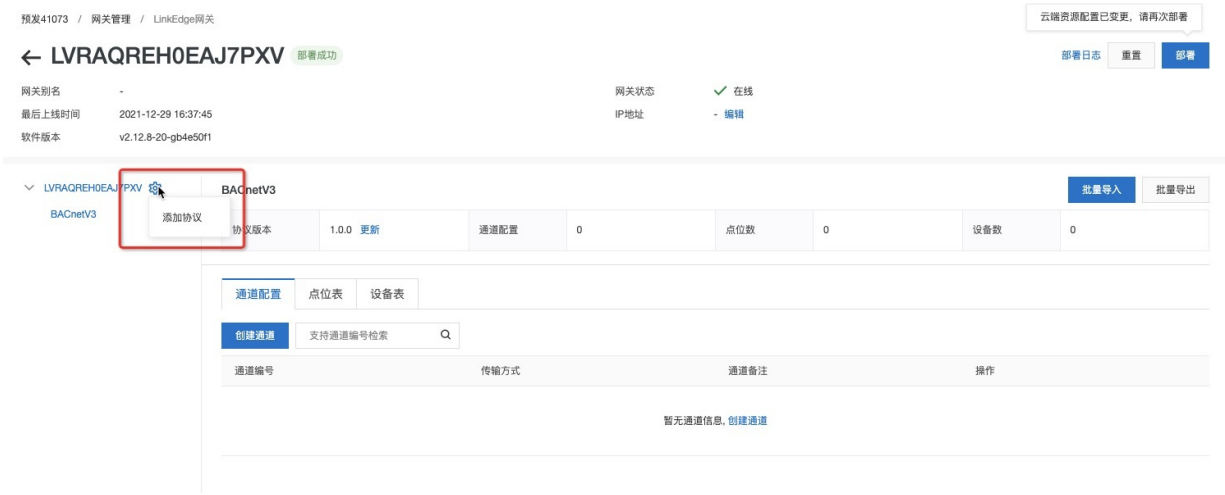
进入项目内，找到**边缘一体机-网关管理**，如下图所示。点击网关对应的**设备集成**



在设备集成页面首次集成时需要点击“**新增协议**”添加协议及设备：



在设备集成页面非首次集成时，需点击网关右侧的设置按钮中的“添加协议”：

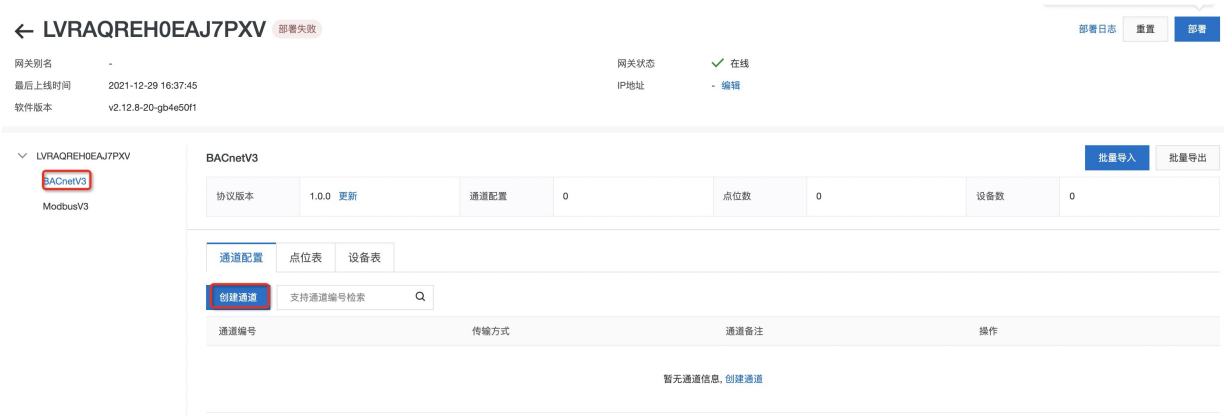


点击网关右侧的设置图标，首先，在弹出菜单中选择**“添加协议”**，然后，在下拉选择的协议列表中选择“**BACnetV3（官方）**”,然后选择协议版本，建议选择**最新版本**进行添加，如下图所示：



3 配置点位

如下图所示，通过点击“**BACnetV3**”驱动，右侧展现驱动管理页，进入“**通道配置**”TAP页，点击“**创建通道**”按钮



BACnet中有通道概念，意指链路。需指定BACnet采用哪种链路进行通信，目前支持UDP方式。

为了提升部署效率，我们扩展了通道属性，把物理设备相关的参数指定在通道中配置

添加配置

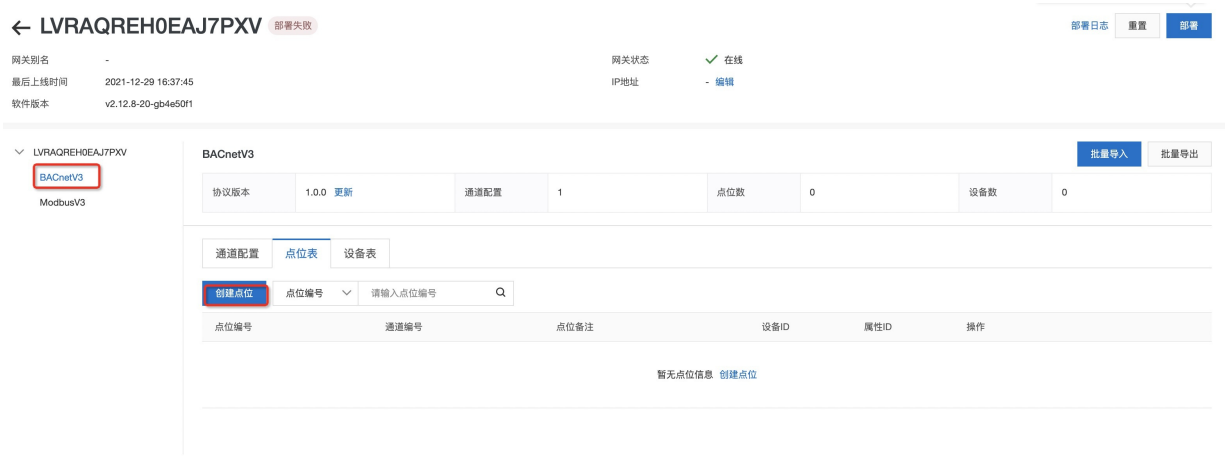


参数说明

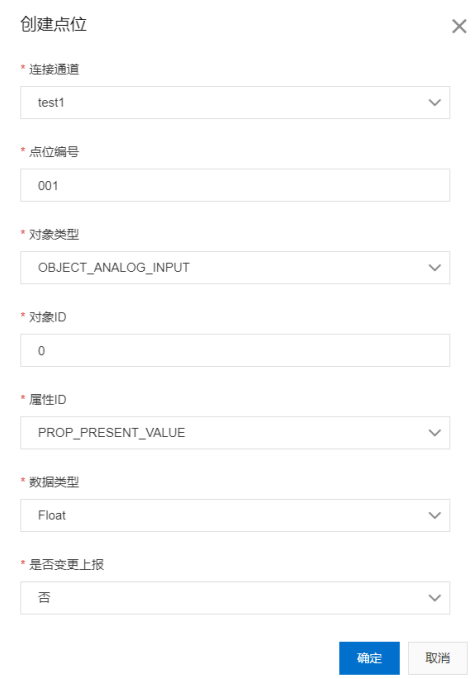
名称	说明
通道编号	用于点位与通道关联，自行定义，需要保证协议内唯一性

传输方式	支持UDP
网卡名称	输入网关接入内网网络的网卡名。
设备ID	BACnet物理设备ID
上报周期	点位数据上报时间间隔
离线判定周期	设备离线的判定时间，建议为上报周期*10
通信方式	定向：根据定向ip、port发现设备；非定向：通过广播方式发现设备
最大报文长度	传输数据的最大长度，通信方式为“定向”时有效
地址	设备ID地址和端口号，格式：ip:port，通信方式为“定向”时有效
通道备注	自定义一个备注信息，方便查看

如下图所示，通过点击**“BACnetV3”**驱动，右侧展现驱动管理页，进入**“点位表”**TAP页，点击**“创建点位”**按钮



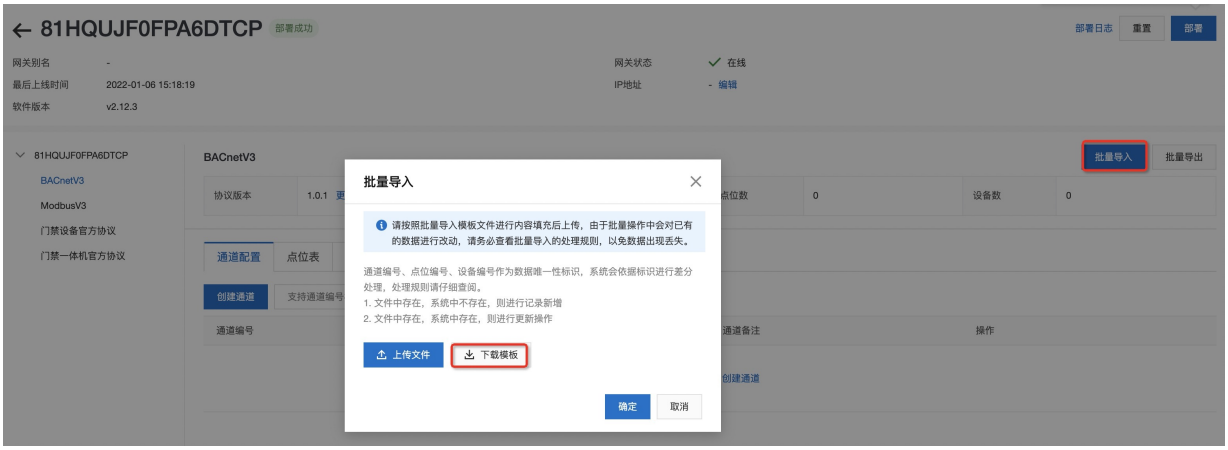
点位的参数请按下图所示进行配置：



参数说明

名称	说明
链接通道	点位连接通道
点位编号	点位编号，全局唯一
对象类型	BACnet设备固有参数，由设备商提供，支持模拟量输入、二进制输入等等，详情见“BACnet点位配置表”
对象ID	BACnet设备固有参数，由设备商提供，详情见“BACnet点位配置表”
属性ID	BACnet设备固有参数，由设备商提供，详情见“BACnet点位配置表”
数据类型	支持bool，int，float，double等类型，详情见“BACnet点位配置表”
是否变更上报	数据发生变化时触发上报

首先下载批量导入模版，如下图所示：



打开导入模版，进入**“通道配置表”**TAP页，如下图所示：

连接编号（单项目单协议唯一）	传输方式（目前仅支持UDP）	网卡名称	设备ID（BACnet ID）	上报时间（大于等于1000ms）	离线判定时间（单位ms,需大于上报时间，且大于等于9000ms）	通信方式（定向-主动添加-模式填1；非定向-广播模式填0）	最大报文长度（通讯方式为"定向"时填写）	访问地址（通讯方式为"定向"时填写）

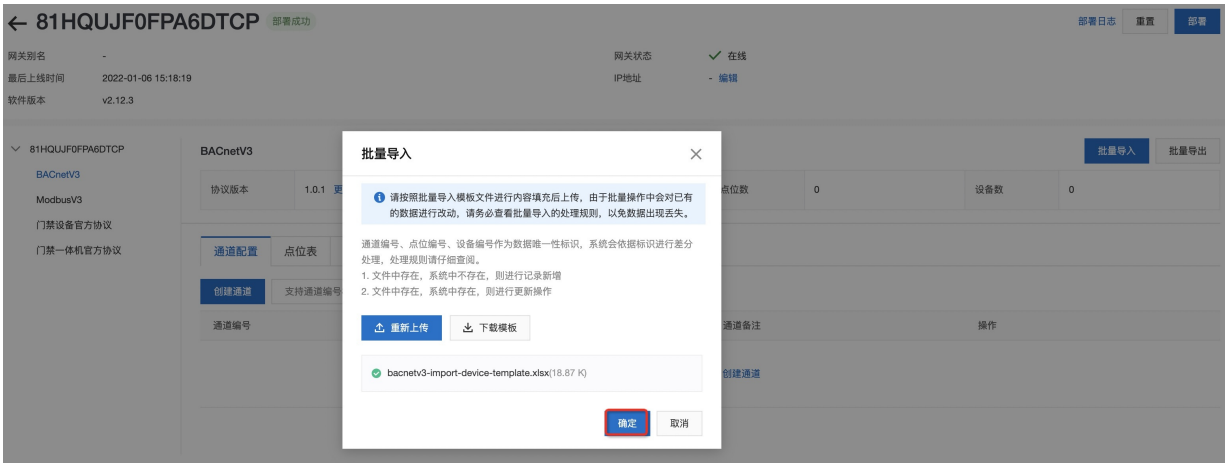
进入**“点位表”**TAP页，如下图所示：

连接通道（保持和通道配置表一致）	点位编号	对象类型（Object Type，详见备注）	对象ID（Object Identifier）	属性ID（Property Identifier，详见备注）	扩展数据类型（详见备注）	是否上报（数据变更上报填1，不上报填0）	备注信息

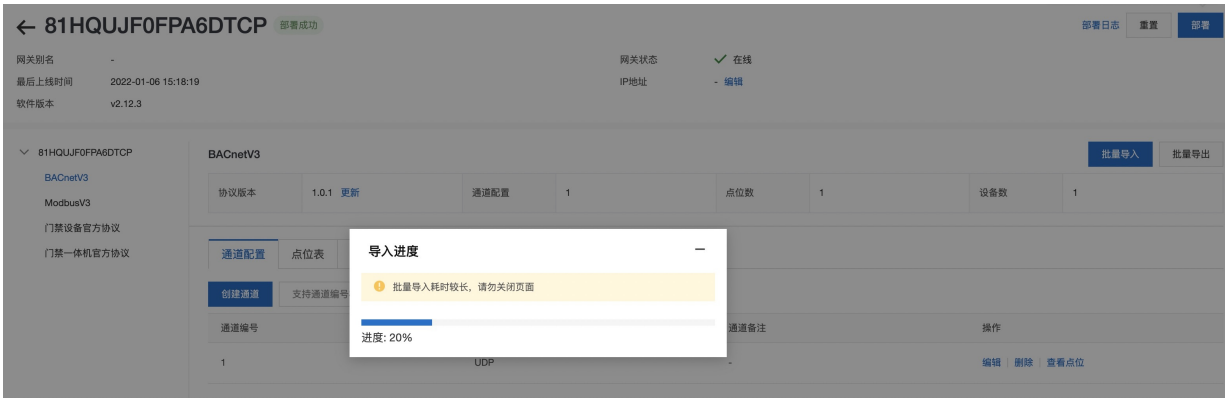
基于导入模版填充好所需导入的配置信息后，点击上传文件，选择所需导入的文件后点击确定按钮，首先会基上传的文件进行预校验，预校验过程主要校验配置内容的合法性，预校验过程中确定按钮为不可点击状态，如下图所示



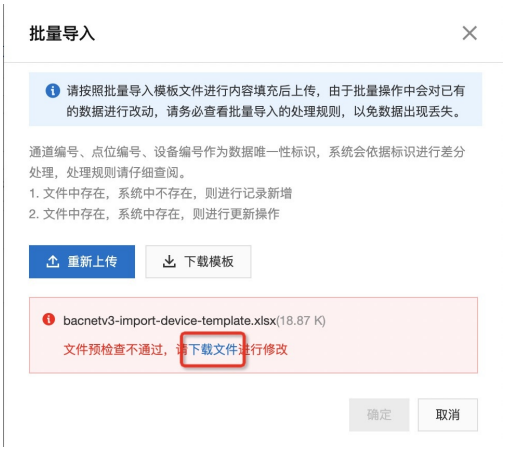
预校验通过后，确定按钮变为可点击状态，点击后进行实际配置导入，如下图所示：



整个导入过程可能需要耗费几分钟时间，请耐心等候（具体由所需导入的点位数，设备数决定），导入进度的展现如下图所示：



当配置数据存在不合法的情况，预校验会不通过，可下载文件进行失败原因的查看，可基于失败原因对导入文件进行修改，入下图所示：



通道配置表预校验失败原因查看，如下图所示：



连接编号（单项目单协议唯一）	传输方式（仅支持UDP）	网卡名称（传输方式为UDP时填写连接内网）	设备ID（BACnet ID）	上报时间（大于等于1000ms）	离线判定时间（单位ms,需大于上报时间，且大于等于9000ms）	通信方式（定向-主动添加-模式填1；非定向-广播模式填0）	最大报文长度（通讯方式为"定向"时填写）	访问地址（通讯方式为"定向"时填写）	错误信息
编号1	UDP	eps01	13	1000	10000	0			通道编号不合法，支持英文字母、数字和特殊字符 "-" 、 "_" 、 "." ，长度限制1~20

通道配置表 点位表 设备表 +

点位表预校验失败原因查看，如下图所示：

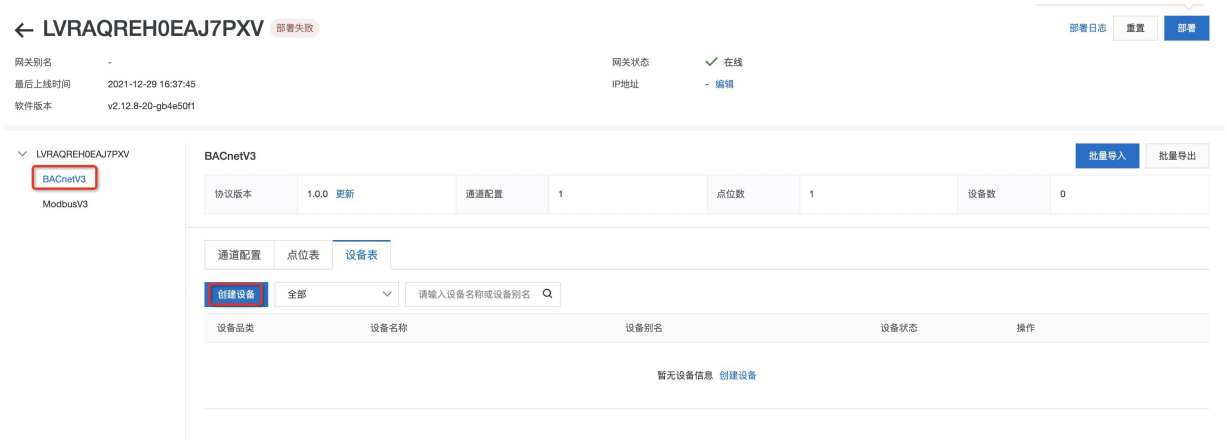


连接通道（保持和通道配置表一致）	点位编号	对象类型（Object Type）	对象ID（Object Identifier）	属性ID（Property Identifier）	扩展数据类型	是否上报（数据变更上报填1，不上报填0）	备注信息	错误信息
30	1	OBJECT_ANALOG_OUTPU	111	PROP_ACTION	0	1		通道：30不存在

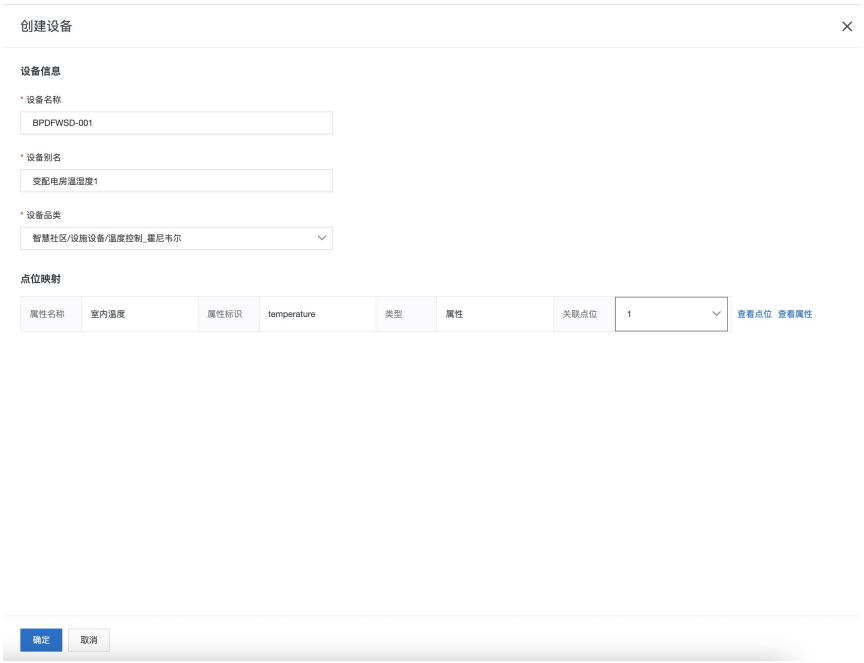
通道配置表 点位表 设备表 +

4 添加设备

如下图所示，通过点击**“BACnetV3”**驱动，右侧展现驱动管理页，进入**“设备表”**TAP页，点击**“创建设备”**按钮



设备的参数请按下图所示进行配置：



参数说明

名称	说明
设备名称	设备的IoT编码
设备别名	设备的备注名称
设备品类	设备的物模型品类

设备表预校验失败原因查看，如下图所示：



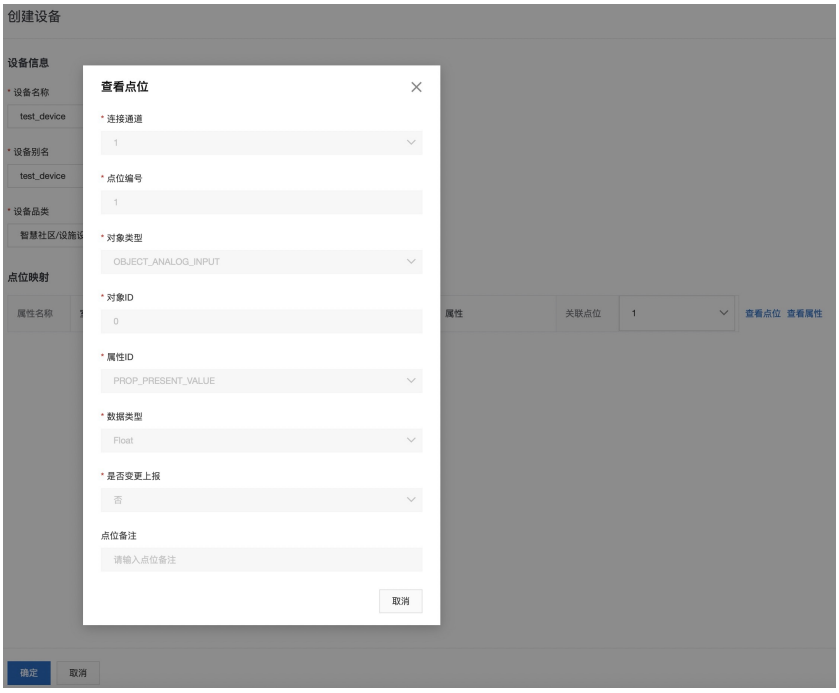
IoT品类ID（遵从设备标准文件中的品类ID要求）	IoT设备ID（遵从设备标准文件中的设备ID要求）	IoT设备中文名称（遵从设备标准文件中的设备名称要求）	IoT功能标识符（遵从设备标准文件中的标识符要求）	点位编号（遵从点位表中的点位编号）	错误信息
ddefrew7ZO8MkGLd2Qxw	BPDFWSD-001	变配电房温湿度1	temperatu	1	功能标识符在品类中不存在

通道配置表 点位表 设备表

点击下拉列表可以进行点位的选择



点击查看点位可以查询选择的点位的详情



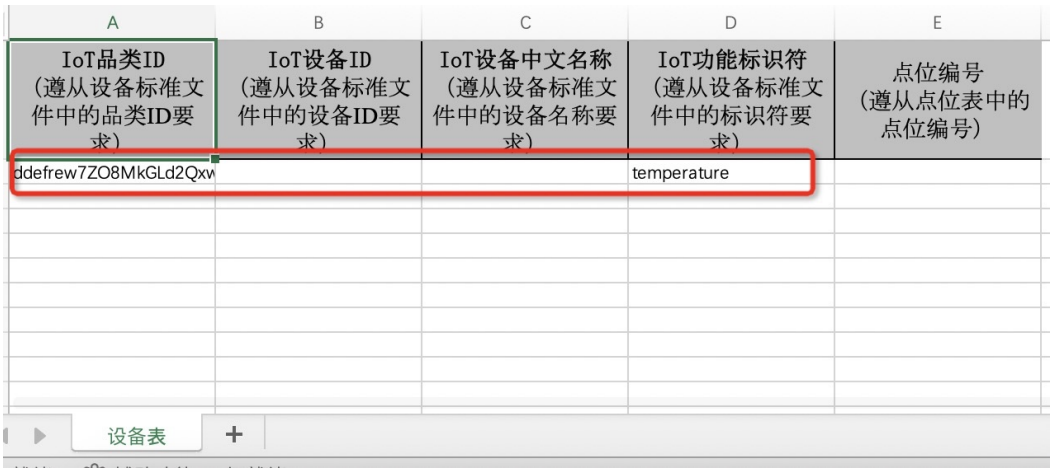
支持基于导入模版批量进行通道信息，点位信息，设备信息的导入

4.3.1 基于导入模版中录入设备信息

查看设备对应的品类信息，本例为温度控制_霍尼韦尔，在品类管理页找到该品类，导出设备模版，如下图所示：



打开导出的模版，红框部分为该品类所具备的功能属性定义，需要拷贝至导入模版<设备表>TAP页，如下图所示：



然后在导入模版<设备表>TAP页补充**IoT设备ID**，**IoT设备中文名称**，**点位编号**信息

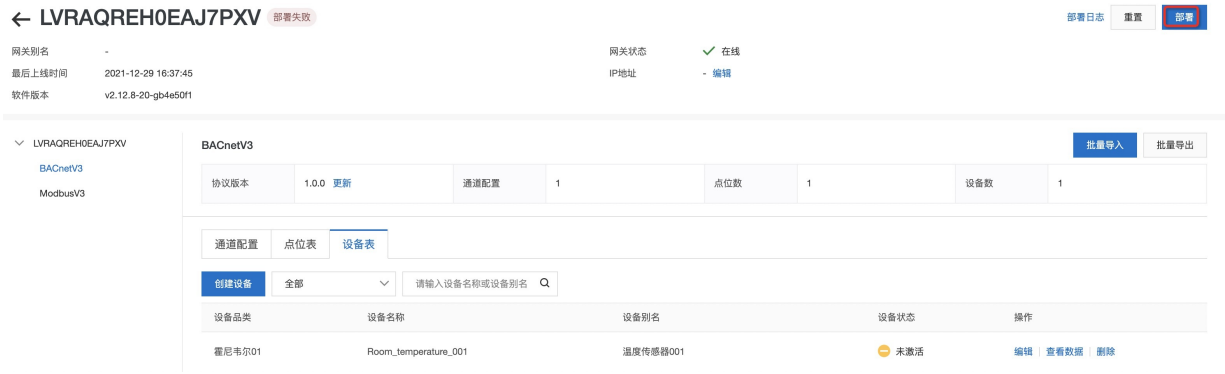


参数说明

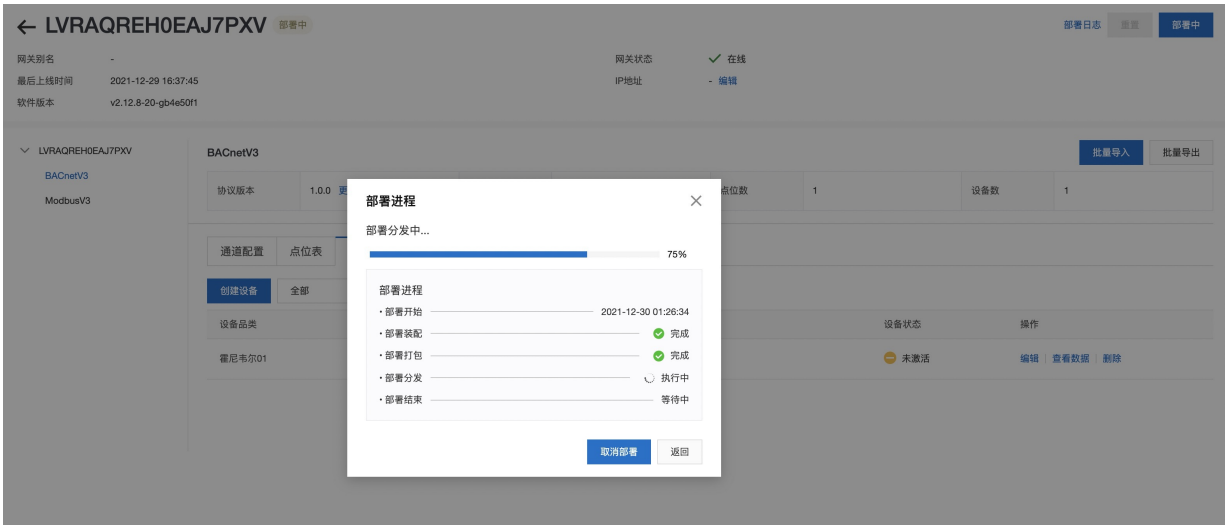
名称	说明
IoT设备ID	设备的IoT编码
IoT设备中文名称	设备的备注名称
点位编号	设备的点位编号，全局唯一

在完成驱动和设备的配置后，点击部署，进行设备配置的协议下发到边缘服务器。如下图所示。

特别提醒：驱动部署时，网关侧对应驱动将重启，该驱动下的所有设备将出现离线后在线。这期间设备不可用，请部署人员提前与甲方沟通部署时间



点击部署后可以查看设备配置下发的状态和部署进度。



5 新建模型



弹出页面如下图所示：



参数说明

名称	说明
分组名称	品类分组名称
编码	品类分组名称对应的英文名称。编码要求支持英文字母、数字和下划线，长度限制1~10

把鼠标放到**智慧社区**右上方会出现个齿轮，把鼠标放到齿轮上，然后会出现**新建品类**，点击**新建品类**，如下图所示：



弹出页面如下图所示：



参数说明

名称	说明
品类名称	品类名称
编码	品类名称对应的英文名称。编码要求支持英文字母、数字和下划线，长度限制1~10

把鼠标放到**设施设备**，点击**新建模型**，如下图所示：



弹出页面，如下图所示：



参数说明

名称	说明
模型名称	品类的模型名称
接入协议	该模型的设备接入协议，固定为“自定义协议”
编码	模型名称对应的英文名称。编码要求支持英文字母、数字和下划线，长度限制1~10

根据5.3里新建模型温度控制_霍尼韦尔，点击**新增功能**，进行点位添加，如下图所示：



参数说明

名称	说明
功能名称	该功能的功能名称
标识符	功能标识符，用于标识该功能
数据类型	数据类型包括：int（原生）、float（原生）、double（原生）、text（原生）、date（String类型UTC毫秒）、bool（0或1的int类型）、enum（int类型，枚举项定义方法与bool类型定义0和1的值方法相同）、struct（结构体类型，可包含前面7种类型，下面使用"specs":[{}]描述包含的对象）、array（数组类型，支持int、double、float、text、struct）
取值范围	数据的取值范围，超出取值范围的数据会被丢弃，数据类型为int、float、double有效
步长	数据变化的最小粒度，数据类型为int、float、double有效
单位	根据实际情况选择，可选无，数据类型为int、float、double有效
数据长度	最大10240，数据类型为text有效
元素个数	最大512，数据类型为array有效
布尔值	布尔值（0/1）对应的参数描述，数据类型为bool有效
枚举项	枚举项对应的参数描述，数据类型为enum有效

读写类型	数据的读写类型，读写或只读

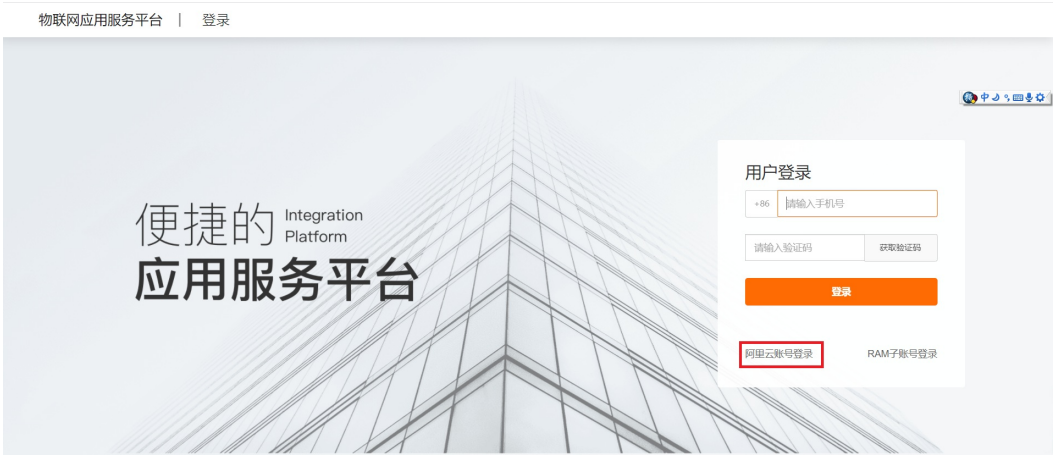
3.3.6. 物业管理一体机账号管理员添加与授权

1 前言

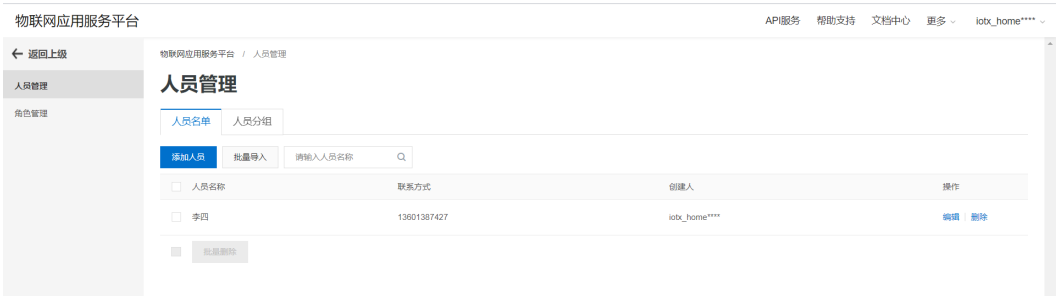
物业管理一体机的部署实施，包括人行/车行/安防等设备的部署实施，可由智能施工人员操作实施。智能施工人员的账号，是通过账号管理员进行添加和授权。本文主要说明如何添加和授权账号管理员。

2 登录阿里云应用服务平台

点击阿里云应用服务平台：请使用**阿里主账号登录**，如下图所示：

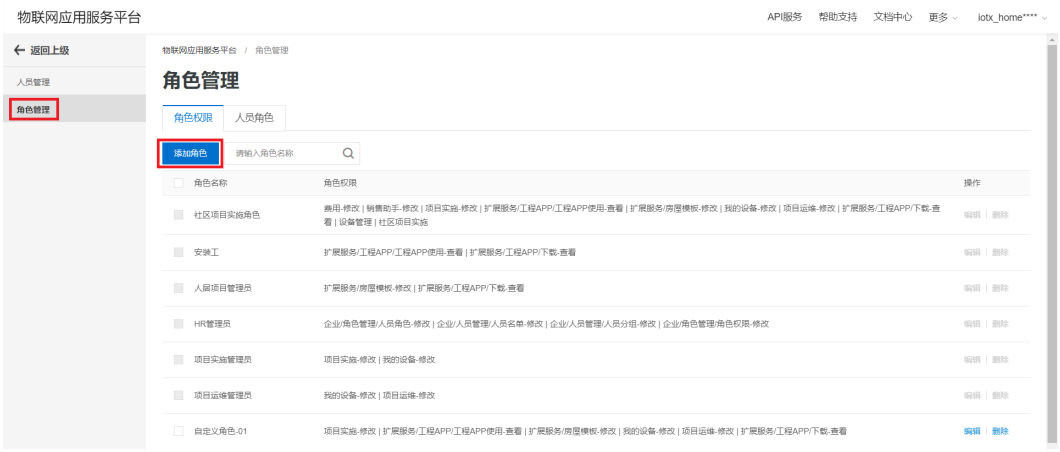


登录成功后，将进入物联网应用服务平台的人员管理界面，如下图所示：



3 创建账号管理员角色

点击**角色管理**，**添加角色**，如下图所示：



弹出页面如下图所示：

角色名称：输入账号管理员。如果该角色已经创建，可以跳过本节："3 创建账号管理员角色"

勾选企业，企业下面的所有角色权限自动勾选。



点击**保存**。

4 添加账号管理员角色和权限

4.1 添加账号管理员账号

点击**人员管理**，在人员名单下点击**添加人员**：如下图所示

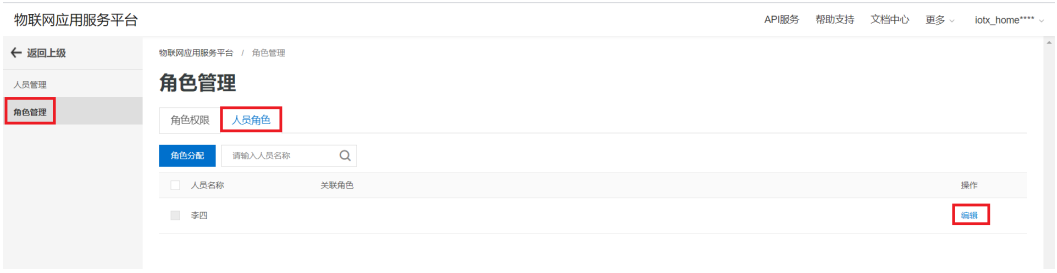


输入已收集好的账号管理员的名字和电话号码，如下图所示



4.2 对账号管理员添加权限

1、点击**角色管理**，选择**人员角色**，如下图所示，

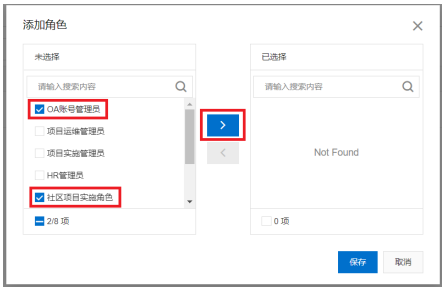


点击某个人员名字右侧的编辑

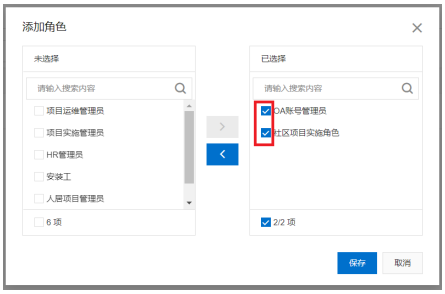
2、点击**添加角色**



选择角色列表里的**账号管理员和社区项目实施角色**，如下图所示。



勾选角色并保存，如下图所示



最后点击确定，完成账号管理员的权限添加。



5 项目对账号管理员进行授权

查看**人员管理**，点击**人员分组**。如下图所示，**公司**下面列出的，就是所有项目名称，即对应的小区名称。

找到需要进行授权的项目/小区名称进行授权。下面示例项目名称为OA账号验证项目，再点击**默认**分组，最后点击**添加人员**，如下图所示：



勾选需要添加到该项目的账号管理人员，可以选择**批量添加**，或者一个个**添加**，如下图所示：



添加完成后，该项目的默认组下面将列出所有添加到人员名单，如下图所示：



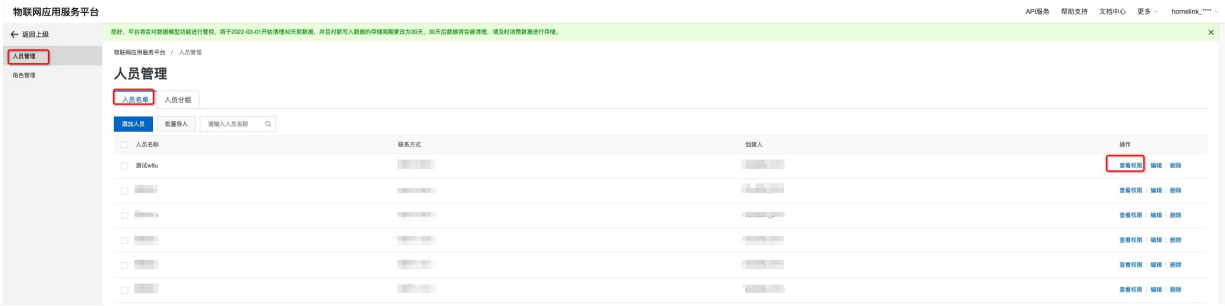
到此，就完成了智能施工人员账号添加，权限授权，以及添加到具体的施工项目里。

6 人员批量绑定/解绑项目分组

6.1人员批量绑定项目分组

查看人员权限：

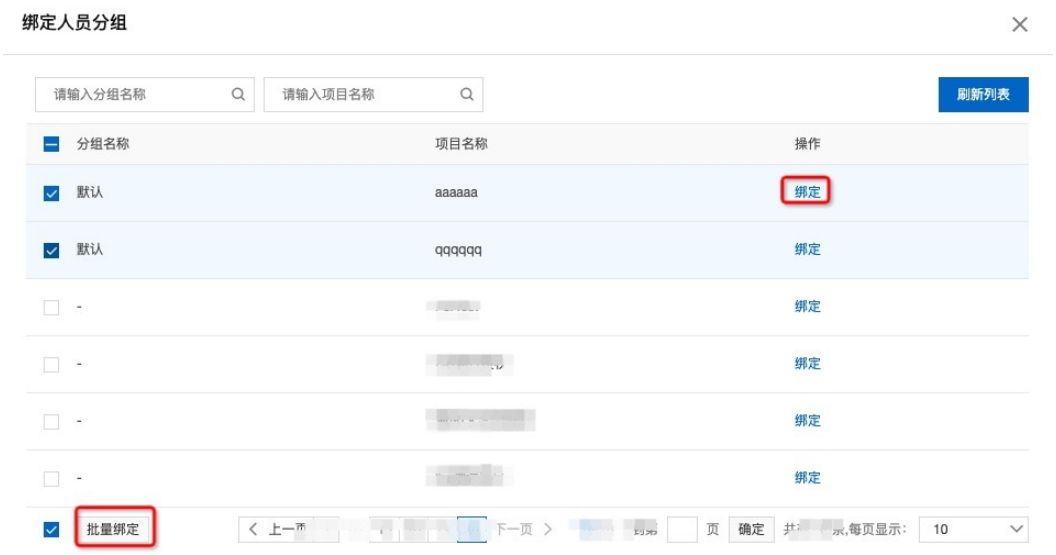
如下图所示，列出的是所有人员名称，找到需要进行绑定项目分组的人员，点击**查看权限**



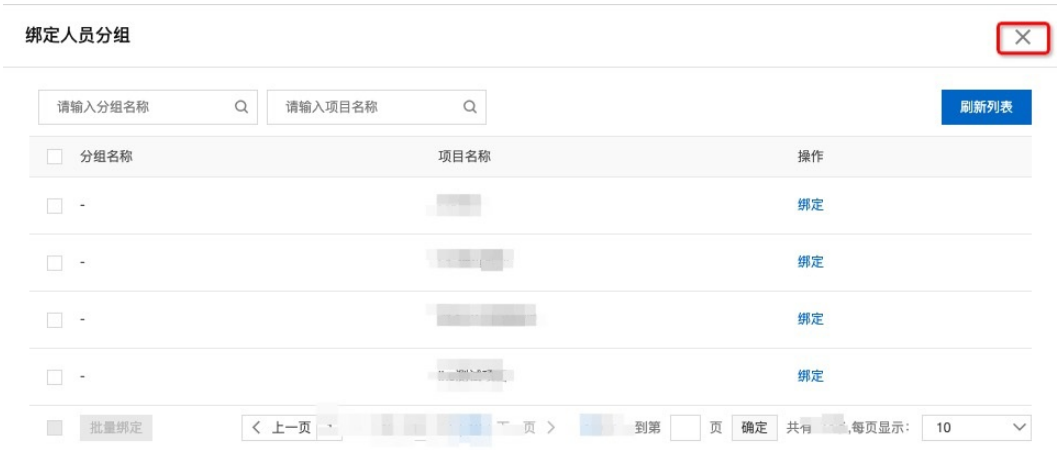
如下图所示，列出的是所有该人员已经绑定的分组列表。如果该人员未绑定分组，列表则没有数据。点击**绑定人员分组**，出现该人员的未绑定分组分页列表，如下图所示



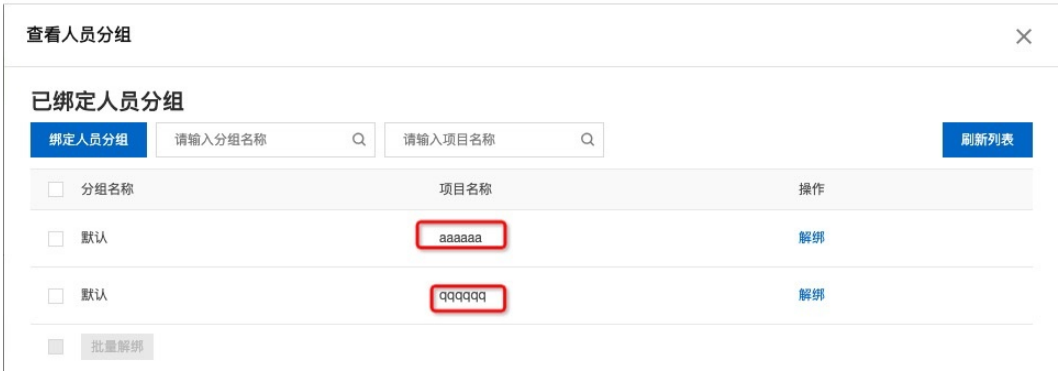
勾选需要绑定的项目分组名称，点击右侧的**绑定**或者下面的**批量绑定**：



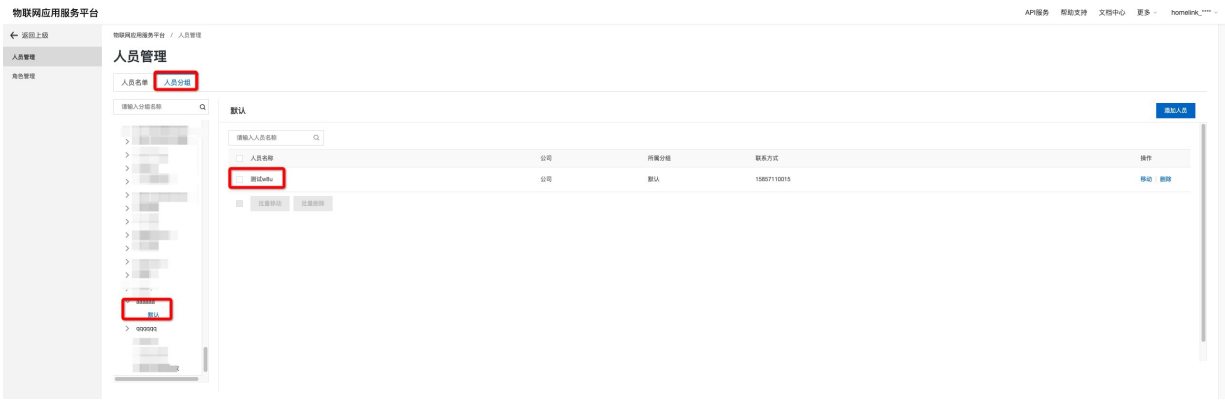
点击按钮 X，退出绑定人员分组



这时显示已绑定人员分组：

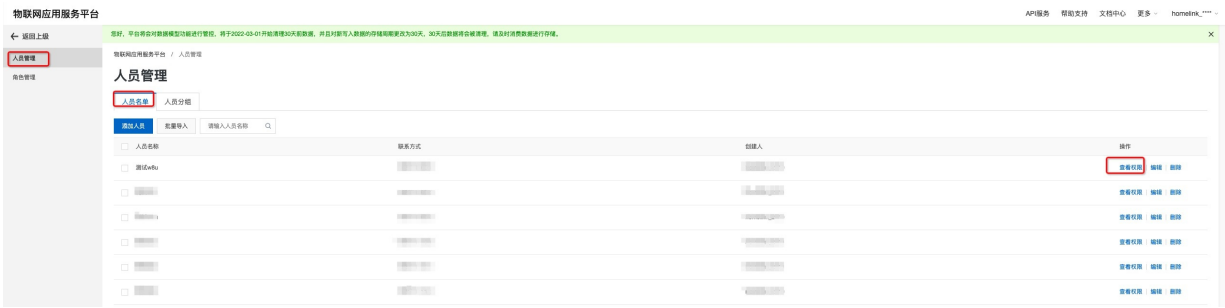


点击**人员分组**，打开相应公司名称和分组名称，可以看到此时人员已经在对应的分组之中，如下图所示：

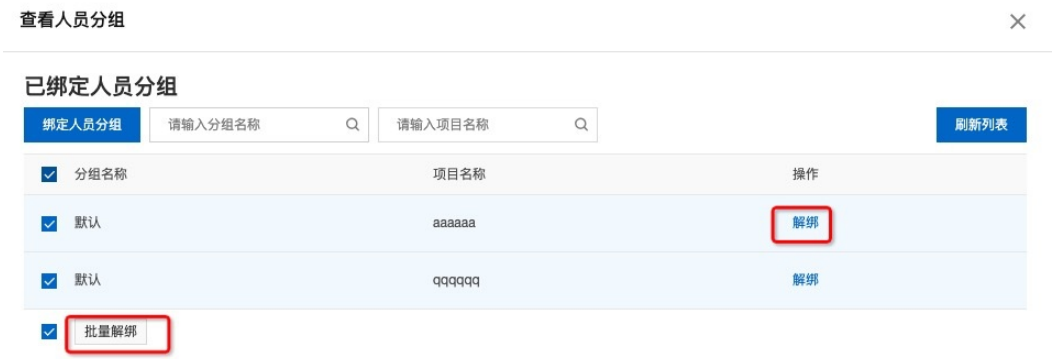


6.2. 人员批量解绑项目分组

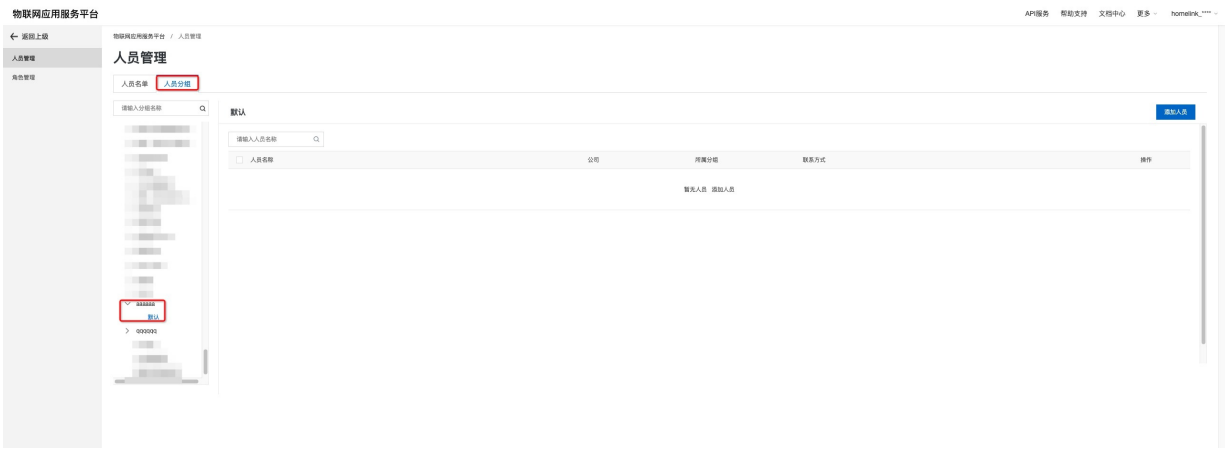
点击**人员管理**，**人员名单**，可对需要进行解绑项目分组的人员进行解绑。点击**查看权限**，如下图所示：



勾选需要解绑的分组名称，点击右侧的**解绑**或者左下角的**批量解绑**：

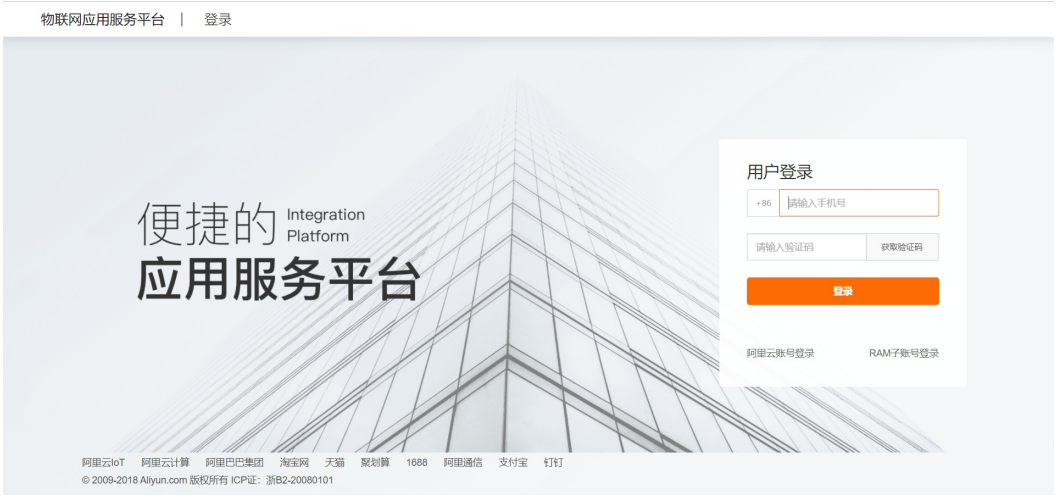


打开**人员分组**，**人员管理**，点击上面已解绑的公司名称和分组名称，对应的分组中已经没有该人员的数据。如下图所示：



7 账号管理员登录方法

账号管理员通过手机短信验证码登录物联网应用服务平台，如下图所示：



3.3.7. 冷备操作手册

前序步骤已经完成了一体机的全部实施配置流程。在后续运行中，如一体机发生硬件故障，本文档描述如何一键替换主机，快速恢复现场业务。

一 支持声明

冷备支持机型

- 物业管理一体机-停车版、物业管理一体机-人行版、物业管理一体机-视频版、物业管理一体机-EBA版、物业管理一体机-通行版、物业管理一体机-综合版

冷备恢复内容

- **应用**：官方组件、三方应用、协议驱动
- **数据**：协议设备配置（非视频类）、刷卡门禁权限、二维码门禁权限、人脸门禁底库、人脸门禁权限、云对讲组关系、人梯联动规则、eba告警规则、边缘空间数据、边缘数据模型授权

二 使用限制

替换机型限制

- 替换机器型号相同，新机器和被替换机器底座版本均为2.1x版本及以上（版本查看方案参考《实施手册》）

应用版本限制

- 协议"**门禁设备官方协议**"版本要求：2.10.0版本及以上（升级参考6.1 协议版本升级，如故障前未升级，可在“四 主机替换操作”之后升级版本，再进行“五 数据恢复操作”）
- 协议"**门禁一体机官方协议**"版本要求：1.0.3版本及以上（升级参考6.1 协议版本升级，如故障前未升级，可在“四 主机替换操作”之后升级版本，再进行“五 数据恢复操作”）
- 组件"**边缘规则引擎**"版本要求：V3.1版本及以上（升级参考6.2 组件版本升级，如故障前未升级，可在“四 主机替换操作”之后升级版本，再进行“五 数据恢复操作”）
- 组件"**边缘通行服务**"版本要求：V3.1版本及以上（升级参考6.1 组件版本升级，如故障前未升级，可在“四 主机替换操作”之后升级版本，再进行“五 数据恢复操作”）

三 前置声明

- 被替换机器断网断电
- 新机器通网通电，并配置成原有网络配置（建议把正在运维的所有项目网络配置备份一份）

四 主机替换操作

使用手机扫描全新的边缘一体机上的二维码进行扫码主机替换。

4.1 扫描二维码激活

找到张贴在边缘一体机上的硬件标签，扫描二维码，激活边缘一体机。以LE-V-B015型号边缘一体机为例，硬件标签张贴在一体机底部。



4.2 使用阿里云账户，登录激活页面（支持子账户登录）

注意：此处登录的阿里云账号，将作为边缘一体机的归属账号，因此请确保使用您的正式生产账号。



4.3 点击立即替换


返回 关闭
激活主机流程
更多
物联网边缘计算主机
主机型号：LE-V-B015
序列号 (SN)
替换主机
用户手册
立即激活

4.4 选择待替换的主机


10:48
替换主机
企业实例
公共实例
被替换主机
请选择被替换主机
确 定

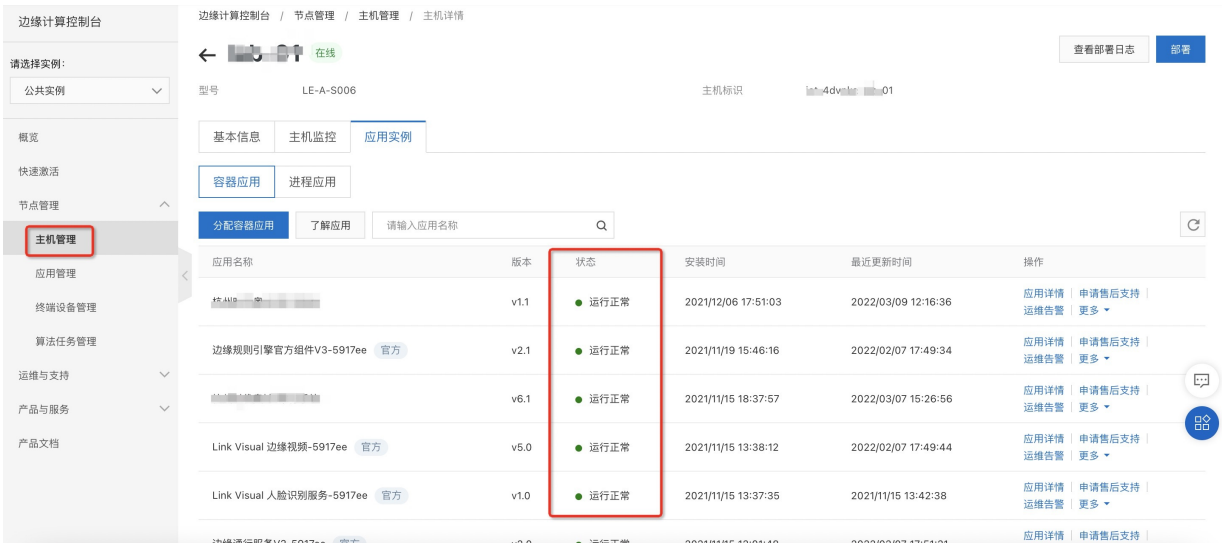
4.5 在“安装电源线和插入网线”提示直接点击下一步

4.6 开始替换



4.7 检查替换进度

替换开始后在 iotedge.console.aliyun.com 中查看进展，替换完成可进入应用管理观察应用运行状态是否恢复成**运行正常**



五 数据恢复操作

5.1 登录控制台项目

登录链接：阿里云应用服务平台，找到相关项目



5.2 设备配置恢复

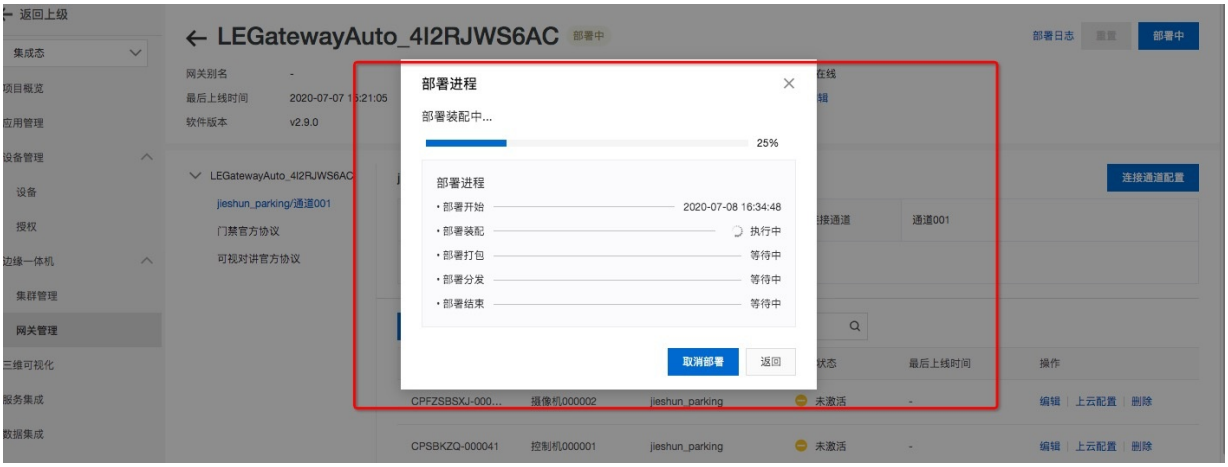
进入项目内，找到**边缘一体机-网关管理**，如下图所示。点击网关对应的**设备集成**



点击部署，推送设备配置到边缘一体机



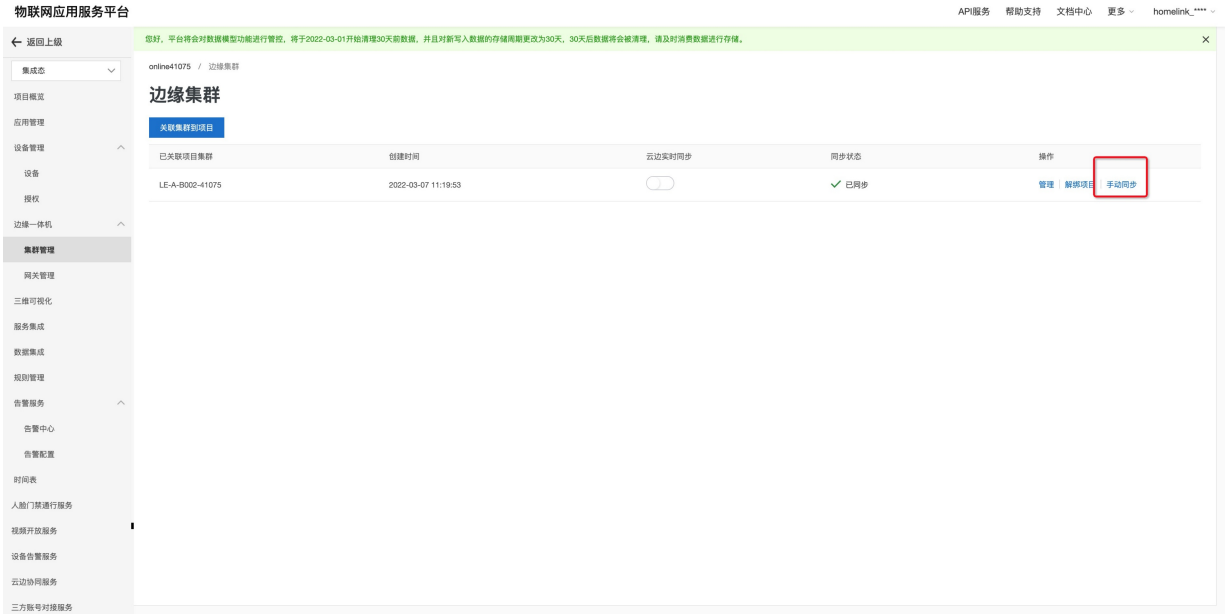
点击部署后可以查看设备配置下发的状态和部署进度



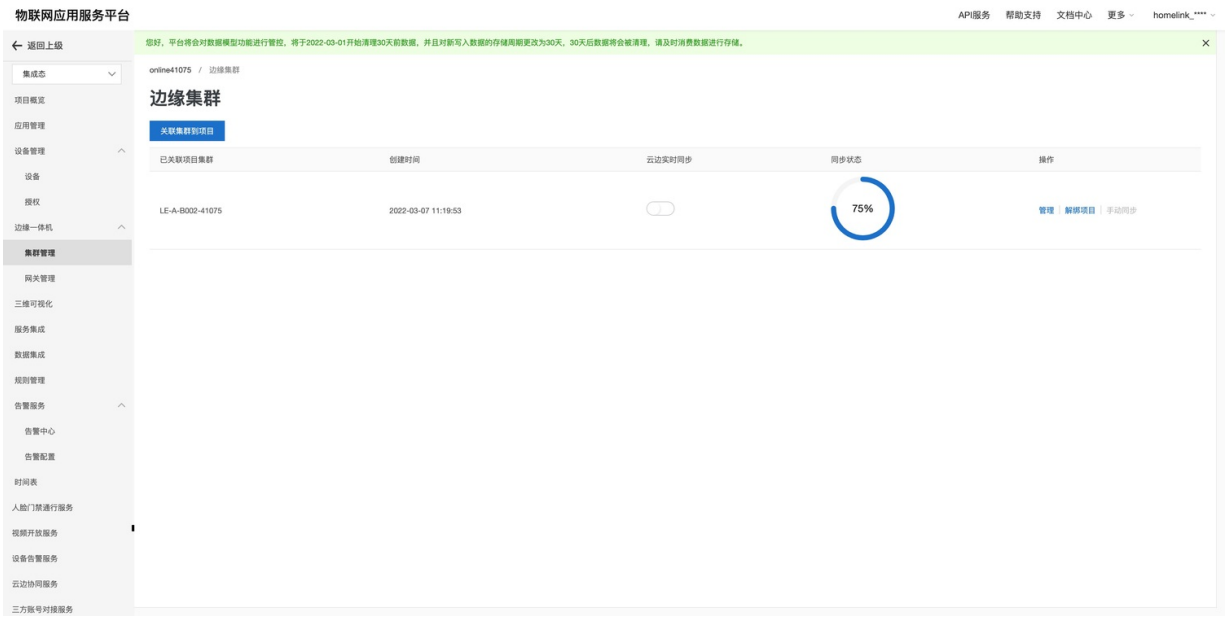
5.3 IoT业务数据恢复

5.3.1 开始数据同步

进入**集群管理**，点击**手动同步**

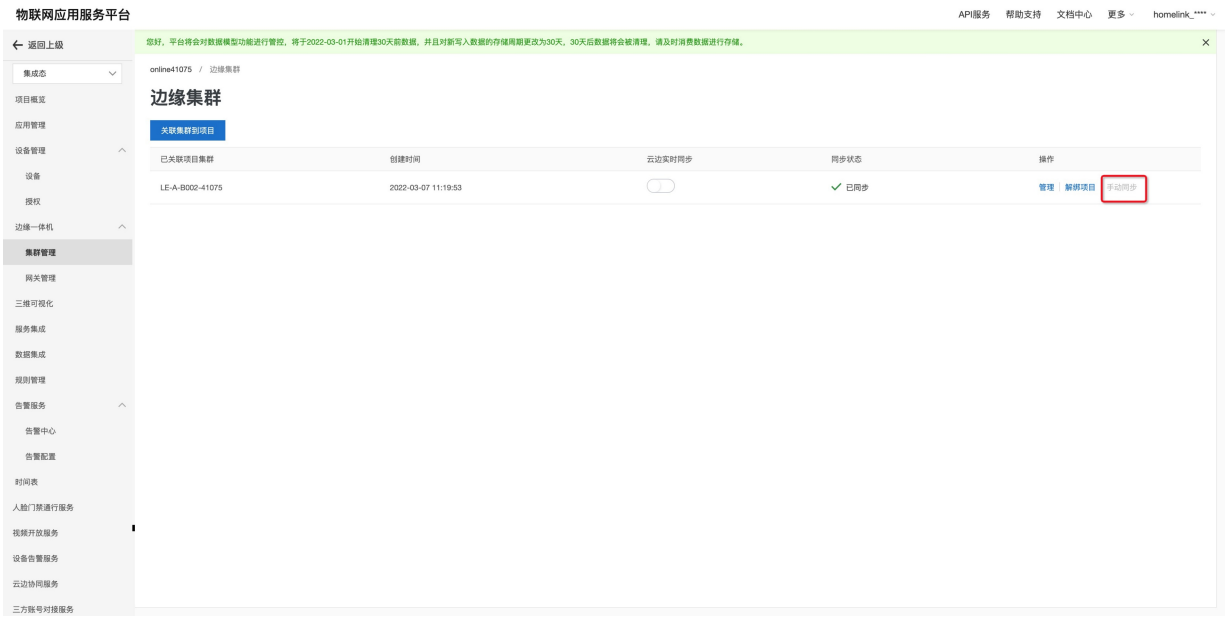


5.3.2 等待同步结束



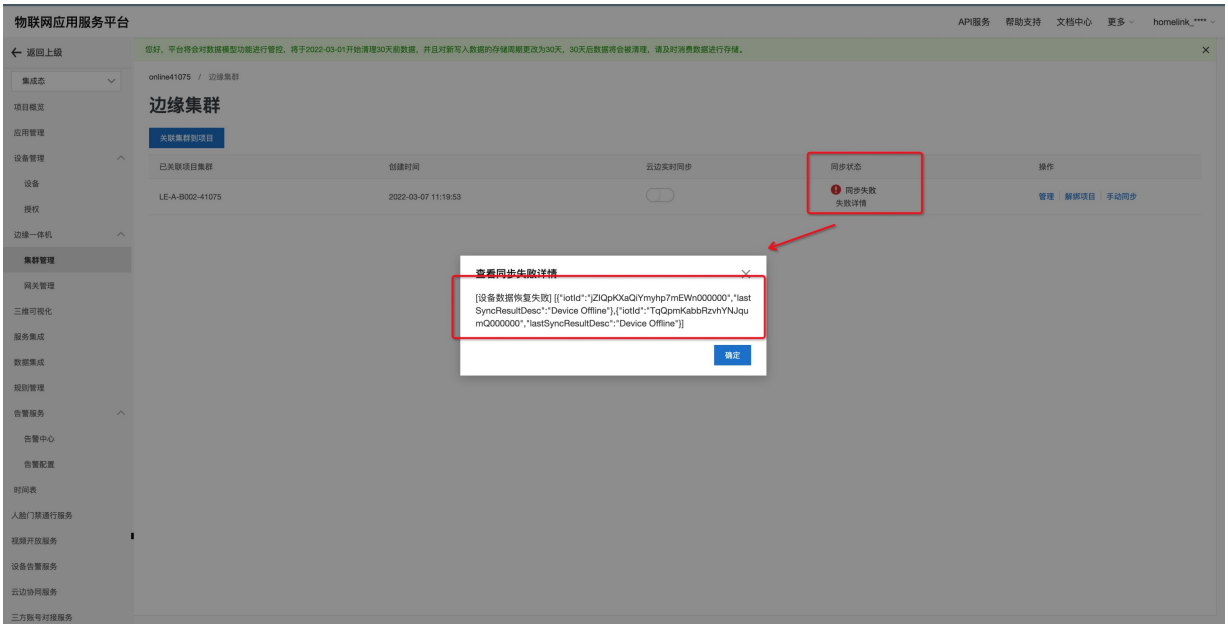
5.3.3 同步成功

如手动同步依旧可点，需重复执行5.3.2操作



5.3.4 同步失败

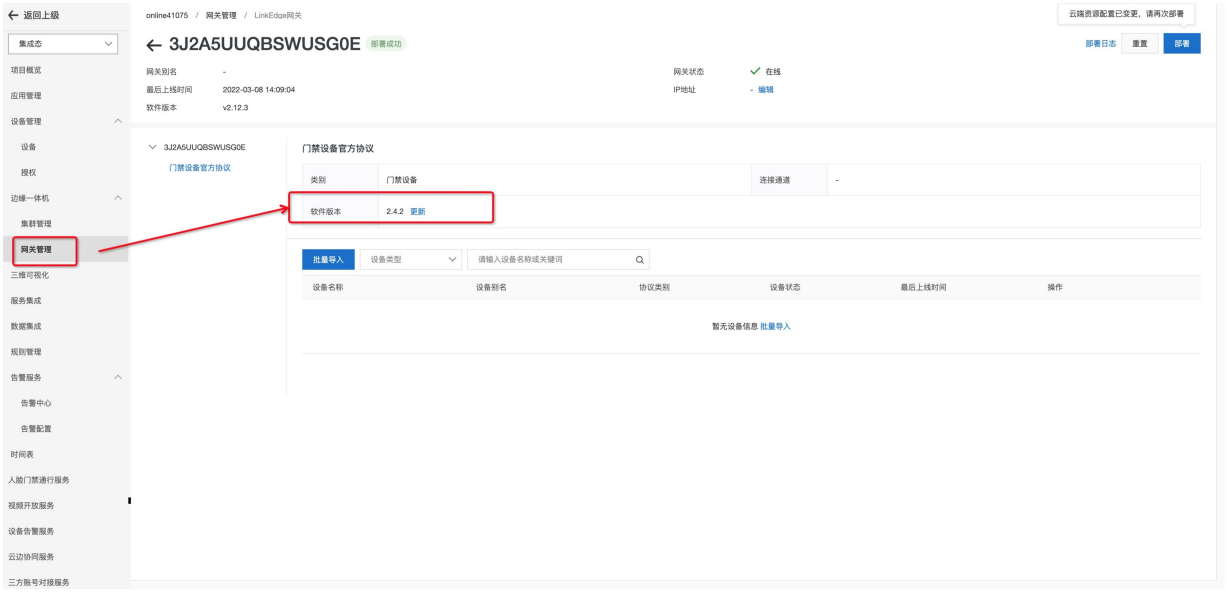
如遇到失败，可点击失败详情，查明原因，恢复驱动或者应用状态后，继续点击手动同步重试。



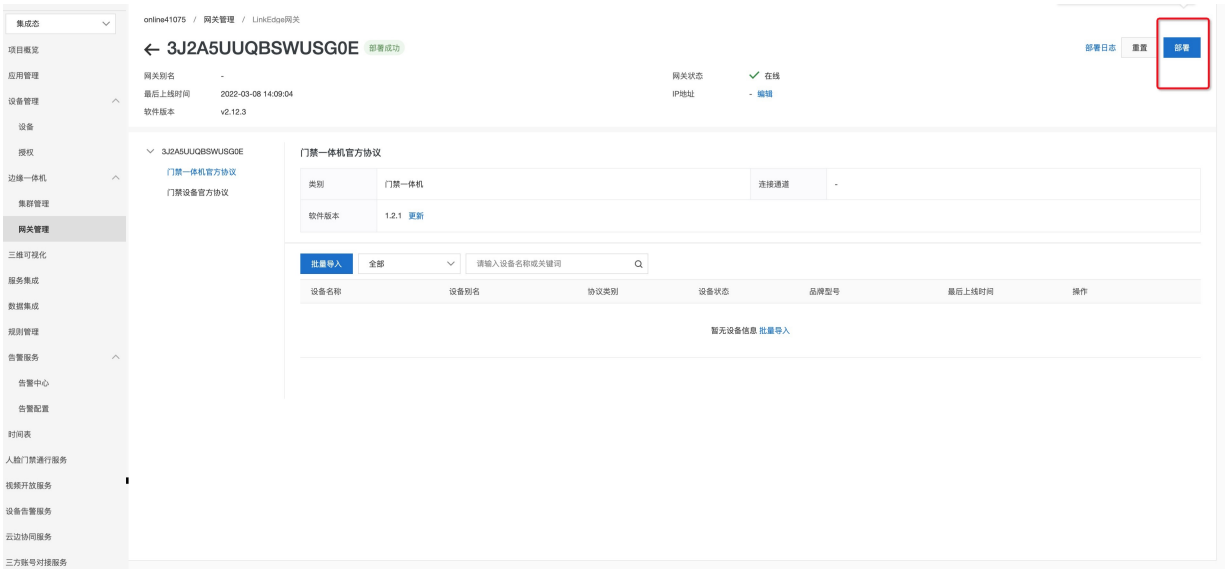
六 附录

6.1 协议版本升级

进入项目，进入**网关管理->设备集成，找到对应的协议名称**，软件版本点击更新选择版本

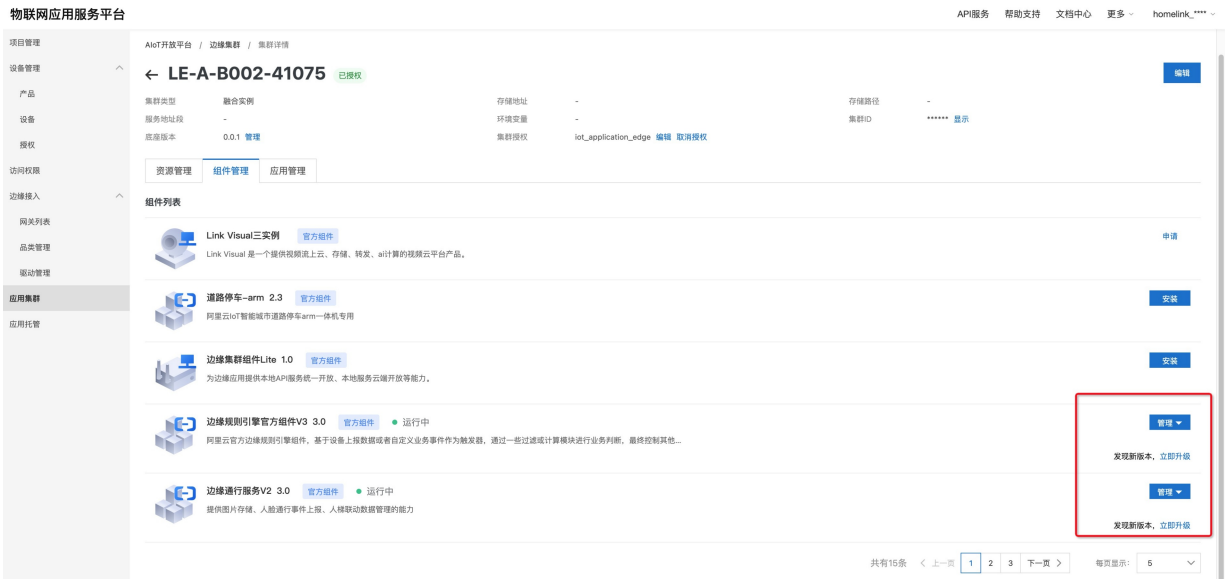


点击“**部署**”更新一体机的协议



6.2 组件版本升级

选择**应用集群**，进入相应的集群，在**组件管理**里，找到对应组件名称，点击“**立即升级**”



3.4. 开放服务

3.4.1. 基础对接指引

API调用对接指引

数据模型对接指引

设备数据订阅指引

3.4.2. 基础接口

3.4.2.1. 应用设备管理服务API

1. 概述

提供应用下设备列表、设备属性及状态获取，设备属性设置及设备服务调用。

2. 接口定义

2.1设备查询

2.1.1 批量获取应用的产品所有设备属性状态

路径	/app/thing/properties/batch/get
版本号	1.0.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
productKey	String	是	产品pk
pageNo	Integer	是	当前页
pageSize	Integer	是	页面大小
lpInstanceId	String	否	LP实例ID

返回数据

名称	**类型**	**描述**
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

data:

名称	**类型**	**描述**
total	Integer	总页数
pageNo	Integer	当前页
pageSize	Integer	每页大小
data	SimpleDeviceInfoAndPropert	分页数据对象列表

分页数据对象 SimpleDeviceInfoAndPropert :

名称	**类型**	**描述**
productName	String	产品名称

productKey	String	产品pk
deviceName	String	设备名称
nodeType	Integer	设备节点
status	String	设备状态
activeTime	String	设备激活时间
lastOnlineTime	String	设备最好一次上线时间
createTime	String	设备创建时间
childDeviceCount	Long	自设备连接数
iotId	String	设备iotId
rangeQueryDTO	DeviceStatusDTO	设备属性快照列表

设备属性快照列表 DeviceStatusDTO:

名称	类型	描述
iotId	String	设备iotId
batchId	String	设备上报的标示。 可以批量上报 通一个批次上报的这个batchid一样
attribute	String	设备属性
group	Integer	设备分组
value	String	属性值
gmtModified	String	修改时间

示例

请求示例:

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 设置参数
request.putParam("productKey","value1");
request.putParam("pageNo","value2");
request.putParam("pageSize","value3");
request.putParam("lpInstanceId","value4");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/properties/batch/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "code":200,
    "data":{
        "total":1,
        "data":[
            {
                "activeTime":"2019-12-12 20:13:35",
                "nodeType":0,
                "productKey":"a17nEF9qLYJ",
                "deviceName":"mIwfRzXY9wLpRFz5RfJJ",
                "utcCreateTime":"2019-12-04T09:00:27.000Z",
                "productName":"studio体验产品",
                "utcActiveTime":"2019-12-12T12:13:35.164Z",
                "utcLastOnlineTime":"2019-12-12T12:13:35.164Z",
                "iotId":"mIwfRzXY9wLpRFz5RfJJ000101",
                "createTime":"2019-12-04 17:00:27",
                "lastOnlineTime":"2019-12-12 20:13:35",
                "rangeQueryDTO":[
                    {
                        "identifier":"LightStatus",
                        "unit":"",
                        "dataType":"bool",
                        "name":"工作状态"
                    },
                    {
                        "identifier":"LightAdjustLevel",
                        "unit":"%",
                        "dataType":"int",
                        "name":"调光等级"
                    },
                    {
                        "identifier":"LightVolt",
                        "unit":"V",
                        "dataType":"float",
                        "name":"工作电压"
                    },
                    {
                        "identifier":"LightCurrent",
                        "unit":"A",
                        "dataType":"float",
                        "name":"工作电流"
                    },
```

```
{
    "identifier":"ActivePower",
    "unit":"W",
    "dataType":"float",
    "name":"有功功率值"
},
{
    "identifier":"PowerRatio",
    "unit":"pF",
    "dataType":"float",
    "name":"功率因数"
},
{
    "identifier":"PowerConsumption",
    "unit":"kW·h",
    "dataType":"float",
    "name":"用电量"
},
{
    "identifier":"DrainVoltage",
    "unit":"V",
    "dataType":"float",
    "name":"漏电压"
},
{
    "identifier":"TiltValue",
    "unit":"°",
    "dataType":"int",
    "name":"倾斜角度值"
},
{
    "identifier":"ErrorPowerThreshold",
    "unit":"W",
    "dataType":"int",
    "name":"故障功率门限"
},
{
    "identifier":"ErrorCurrentThreshold",
    "unit":"A",
    "dataType":"float",
    "name":"故障电流门限"
},
{
    "identifier":"TiltThreshold",
    "unit":"°",
    "dataType":"int",
    "name":"倾斜阈值"
},
{
    "identifier":"UnderVoltThreshold",
    "unit":"V",
    "dataType":"int",
    "name":"欠压阈值"
},
{
    "identifier":"OverCurrentThreshold",
    "unit":"A",
    "dataType":"int",
    "name":"过流阈值"
},
{
    "identifier":"OverVoltThreshold",
    "unit":"V",
    "dataType":"int",
    "name":"过压阈值"
},
{
    "identifier":"LightErrorEnable",
    "unit":"",
    "dataType":"bool",
    "name":"灯具故障使能"
},
{
    "identifier":"OverCurrentEnable",
    "unit":"",
    "dataType":"bool",
    "name":"过流告警使能"
},
{
```

```
                        "identifier":"OverVoltEnable",
                        "unit":"",
                        "dataType":"bool",
                        "name":"过压告警使能"
                    },
                    {
                        "identifier":"UnderVoltEnable",
                        "unit":"",
                        "dataType":"bool",
                        "name":"欠压告警使能"
                    },
                    {
                        "identifier":"LeakageEnable",
                        "unit":"",
                        "dataType":"bool",
                        "name":"漏电告警使能"
                    },
                    {
                        "identifier":"OverTiltEnable",
                        "unit":"",
                        "dataType":"bool",
                        "name":"倾斜告警使能"
                    },
                    {
                        "identifier":"LampError",
                        "unit":"",
                        "dataType":"bool",
                        "name":"灯具故障告警"
                    },
                    {
                        "identifier":"OverCurrentError",
                        "unit":"",
                        "dataType":"bool",
                        "name":"过流告警"
                    },
                    {
                        "identifier":"OverVoltError",
                        "unit":"",
                        "dataType":"bool",
                        "name":"过压告警"
                    },
                    {
                        "identifier":"UnderVoltError",
                        "unit":"",
                        "dataType":"bool",
                        "name":"欠压告警"
                    },
                    {
                        "identifier":"OverTiltError",
                        "unit":"",
                        "dataType":"bool",
                        "name":"倾斜告警"
                    },
                    {
                        "identifier":"LeakageError",
                        "unit":"",
                        "dataType":"bool",
                        "name":"漏电告警"
                    },
                    {
                        "identifier":"GeoLocation",
                        "unit":"",
                        "dataType":"struct",
                        "name":"地理位置"
                    }
                ],
                "status":"OFFLINE"
            }
        ],
        "pageNo":1,
        "pageSize":20
    },
    "message":"success"
}
```


异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.1.2 获取物的属性timeline数据

路径	/app/thing/property/timeline/get
版本号	1.0.4
协议	HTTP/HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	否	物的唯一标识
productKey	String	否	物的产品标识
deviceName	String	否	物的名称
identifier	String	是	设备属性标识符
start	Long	是	开始时间
end	Long	是	结束时间
pageSize	Integer	是	单次查询最大条数
ordered	Boolean	是	查询顺序，true-顺序查询，false-倒序查询
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
timestamp	Long	时间戳
items	ThingEventDTO	设备的事件数据列表

设备的事件数据列表 ThingEventDTO:

名称	类型	描述
iotId	String	物的id
eventCode	String	事件 code
eventName	String	事件名称
eventType	String	事件类型
timestamp	Long	时间戳
eventBody	Object	事件 body
batchId	String	批次 id

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.4");
// 设置参数
request.putParam("iotId","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("identifier","value4");
request.putParam("start","value5");
request.putParam("end","value6");
request.putParam("pageSize","value7");
request.putParam("ordered","value8");
request.putParam("lpInstanceId","value9");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/property/timeline/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "code": 200,
  "data": {
    "items": [
      {
        "batchId": "364f191a24c54f4ebf8254ee6945cca4",
        "data": 48.2,
        "property": "Humidity",
        "timestamp": 1604801094028
      },
      {
        "batchId": "8e70129623a04bc18bc152e6f95a50cb",
        "data": 48.2,
        "property": "Humidity",
        "timestamp": 1604808221032
      }
    ],
    "timestamp": 1604808221032
  },
  "message": "success"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.1.3 批量获取应用的设备信息

路径	/app/thing/info/list/get
版本号	1.4.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotIds	List<String>	否	设备id列表
productKey	String	否	设备的产品pk，需要与设备名称列表组合使用
deviceName	List<String>	否	设备名称列表，需要与设备的产品pk组合使用
appId	字符串	否	应用id。传入应用id时，设备id、设备的产品pk、设备名称无需传递
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	List<DeviceDTO>	响应结果: 设备属性列表

设备属性列表 DeviceDTO:

名称	类型	描述
gmtCreate	Date	设备创建时间
gmtModified	Date	设备修改时间
iotId	String	设备id
name	String	设备名称
deviceSecret	String	设备sk
productKey	String	产品pk
mac	String	设备物理地址
region	String	设备区域
status	int	设备状态
activeTime	Date	设备激活时间
netAddress	String	设备网络地址

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.4.1");
// 设置参数
request.putParam("iotIds","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("appId","value4");
request.putParam("lpInstanceId","value5");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/info/list/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
 "code": 200,
 "data": [{
  "gmtModified": 1564971320000,
  "activeTime": 1576214088740,
  "deviceKey": "ErMmUJAE6nRduVjQ7wi9",
  "productKey": "a1QcurjTB1h",
  "statusLast": 3,
  "gmtCreate": 1564971320000,
  "deviceSecret": "7pRFl0XfRmFldKk4NzfGng3y3B3t1mF5",
  "iotId": "ErMmUJAE6nRduVjQ7wi9000101",
  "name": "wjgT_0034",
  "thingType": "DEVICE",
  "region": "cn-hangzhou",
  "rbacTenantId": "87F2BEFA5D9B45B9AE2CD8C41AC40613",
  "status": 1
 }],
 "id": "4f3196f64a1d480bafa45e9c8006f543"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.1.4 获取物的基本信息

路径	/app/thing/info/get
版本号	1.0.3

协议	HTTP/HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	字符串	否	物的唯一标识
productKey	字符串	否	物的产品标识
deviceName	字符串	否	物的名称
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果： 设备属性 DeviceDTO

设备属性 DeviceDTO:

名称	类型	描述
gmtCreate	Date	设备创建时间
gmtModified	Date	设备修改时间
iotId	String	设备id
name	String	设备名称
deviceSecret	String	设备sk
productKey	String	产品pk
mac	String	设备物理地址
region	String	设备区域
status	int	设备状态

activeTime	Date	设备激活时间
netAddress	String	设备网络地址

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.3");
// 设置参数
request.putParam("iotId","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("lpInstanceId","value4");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/info/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "code":200,
    "data":{
        "gmtModified":1607334316000,
        "deviceKey":"1g84blz9FRQOrlqB0OT1",
        "productKey":"aljemBywtIo",
        "statusLast":0,
        "gmtCreate":1607334316000,
        "productName":"apigatewaytest",
        "iotId":"1g84blz9FRQOrlqB0OT1000101",
        "deviceSecret":"683c84dc5eb88705bd5cf4fbf65f6802",
        "name":"apigatewaydevice",
        "nickname":"apigatewaydevice",
        "thingType":"DEVICE",
        "region":"cn-hangzhou",
        "rbacTenantId":"87F2BEFA5D9B45B9AE2CD8C41AC40613",
        "status":0
    },
    "id":"244bc36c-5fd0-4fec-bd1a-b57f38db3769"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止"
}
```

错误码

错误码	错误信息	错误描述
40062	not authorization query device	没有权限查询设备

2.1.5 根据三元组获取物的模板

路径	/app/thing/dsl/get
版本号	1.0.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	否	物的唯一标识
productKey	String	否	物的产品标识
deviceName	String	否	物的名称
lpInstanceId	String	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	json格式物模版信息TSL

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 设置参数
request.putParam("iotId","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("lpInstanceId","value4");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/dsl/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
 "code": 200,
 "data": "{\"schema\":\"https://iotx-tsl.oss-ap-southeast-1.aliyuncs.com/schema.json\",\"profile\":{\"productKey\":\"a1PHv5Xn2xX\",\"deviceName\":\"xixi\"},\"link\":\"/sys/a1PHv5Xn2xX/xixi/thing/\",\"services\":[{\"outputData\":[],\"identifier\":\"set\",\"inputData\":[{\"identifier\":\"CurtainOperation\",\"dataType\":{\"specs\":{\"0\":\"关窗帘\",\"1\":\"开\",\"2\":\"暂停\"},\"type\":\"enum\"},\"name\":\"模式\"},{\"identifier\":\"PowerSwitch\",\"dataType\":{\"specs\":{\"0\":\"关\",\"1\":\"开\"},\"type\":\"bool\"},\"name\":\"电源开关\"},{\"identifier\":\"CurtainPosition\",\"dataType\":{\"specs\":{\"unit\":\"%\",\"min\":\"0\",\"unitName\":\"百分比\",\"max\":\"100\",\"step\":\"1\"},\"type\":\"int\"},\"name\":\"位置\"},{\"identifier\":\"LocalTimer\",\"dataType\":{\"specs\":{\"item\":{\"specs\":[{\"identifier\":\"Timer\",\"dataType\":{\"specs\":{\"length\":\"255\"},\"type\":\"text\"},\"name\":\"定时时间\"},{\"identifier\":\"Enable\",\"dataType\":{\"specs\":{\"0\":\"关\",\"1\":\"开\"},\"type\":\"bool\"},\"name\":\"启用\"},{\"identifier\":\"IsValid\",\"dataType\":{\"specs\":{\"0\":\"无效\",\"1\":\"有效\"},\"type\":\"bool\"},\"name\":\"可执行\"},{\"identifier\":\"Targets\",\"dataType\":{\"specs\":{\"length\":\"2048\"},\"type\":\"text\"},\"name\":\"定时动作\"}],\"type\":\"struct\"},\"size\":\"109\"},\"type\":\"array\"},\"name\":\"本地定时\"}],\"method\":\"thing.service.property.set\",\"name\":\"set\",\"required\":true,\"callType\":\"async\",\"desc\":\"属性设置\"},{\"outputData\":[{\"identifier\":\"CurtainOperation\",\"dataType\":{\"specs\":{\"0\":\"关窗帘\",\"1\":\"开\",\"2\":\"暂停\"},\"type\":\"enum\"},\"name\":\"模式\"},{\"identifier\":\"PowerSwitch\",\"dataType\":{\"specs\":{\"0\":\"关\",\"1\":\"开\"},\"type\":\"bool\"},\"name\":\"电源开关\"},{\"identifier\":\"CurtainPosition\",\"dataType\":{\"specs\":{\"unit\":\"%\",\"min\":\"0\",\"unitName\":\"百分比\",\"max\":\"100\",\"step\":\"1\"},\"type\":\"int\"},\"name\":\"位置\"},{\"identifier\":\"LocalTimer\",\"dataType\":{\"specs\":{\"item\":{\"specs\":[{\"identifier\":\"Timer\",\"dataType\":{\"specs\":{\"length\":\"255\"},\"type\":\"text\"},\"name\":\"定时时间\"},{\"identifier\":\"Enable\",\"dataType\":{\"specs\":{\"0\":\"关\",\"1\":\"开\"},\"type\":\"bool\"},\"name\":\"启用\"},{\"identifier\":\"IsValid\",\"dataType\":{\"specs\":{\"0\":\"无效\",\"1\":\"有效\"},\"type\":\"bool\"},\"name\":\"可执行\"},{\"identifier\":\"Targets\",\"dataType\":{\"specs\":{\"length\":\"2048\"},\"type\":\"text\"},\"name\":\"定时动作\"}],\"type\":\"struct\"},\"size\":\"109\"},\"type\":\"array\"},\"name\":\"本地定时\"}],\"identifier\":\"get\",\"inputData\":[\"CurtainOperation\",\"PowerSwitch\",\"CurtainPosition\",\"LocalTimer\"],\"method\":\"thing.service.property.get\",\"name\":\"get\",\"required\":true,\"callType\":\"async\",\"desc\":\"属性获取\"}],\"properties\":[{\"identifier\":\"CurtainOperation\",\"dataType\":{\"specs\":{\"0\":\"关窗帘\",\"1\":\"开\",\"2\":\"暂停\"},\"type\":\"enum\"},\"name\":\"模式\",\"accessMode\":\"rw\",\"required\":false},{\"identifier\":\"PowerSwitch\",\"dataType\":{\"specs\":{\"0\":\"关\",\"1\":\"开\"},\"type\":\"bool\"},\"name\":\"电源开关\",\"accessMode\":\"rw\",\"required\":false},{\"identifier\":\"CurtainPosition\",\"dataType\":{\"specs\":{\"unit\":\"%\",\"min\":\"0\",\"unitName\":\"百分比\",\"max\":\"100\",\"step\":\"1\"},\"type\":\"int\"},\"name\":\"位置\",\"accessMode\":\"rw\",\"required\":false},{\"identifier\":\"LocalTimer\",\"dataType\":{\"specs\":{\"item\":{\"specs\":[{\"identifier\":\"Timer\",\"dataType\":{\"specs\":{\"length\":\"255\"},\"type\":\"text\"},\"name\":\"定时时间\"},{\"identifier\":\"Enable\",\"dataType\":{\"specs\":{\"0\":\"关\",\"1\":\"开\"},\"type\":\"bool\"},\"name\":\"启用\"},{\"identifier\":\"IsValid\",\"dataType\":{\"specs\":{\"0\":\"无效\",\"1\":\"有效\"},\"type\":\"bool\"},\"name\":\"可执行\"},{\"identifier\":\"Targets\",\"dataType\":{\"specs\":{\"length\":\"2048\"},\"type\":\"text\"},\"name\":\"定时动作\"}],\"type\":\"struct\"},\"size\":\"109\"},\"type\":\"array\"},\"name\":\"本地定时\",\"accessMode\":\"rw\",\"required\":true,\"desc\":\"用于为设备配置本地控制，其中前三个参数必选，其它参数自主填写，用于表示该产品哪些功能作为“可被本地控制的功能”，这些参数请和设备的标准功能保持一致。\"}],\"events\":[{\"outputData\":[{\"identifier\":\"CurtainOperation\",\"dataType\":{\"specs\":{\"0\":\"关窗帘\",\"1\":\"开\",\"2\":\"暂停\"},\"type\":\"enum\"},\"name\":\"模式\"},{\"identifier\":\"PowerSwitch\",\"dataType\":{\"specs\":{\"0\":\"关\",\"1\":\"开\"},\"type\":\"bool\"},\"name\":\"电源开关\"},{\"identifier\":\"CurtainPosition\",\"dataType\":{\"specs\":{\"unit\":\"%\",\"min\":\"0\",\"unitName\":\"百分比\",\"max\":\"100\",\"step\":\"1\"},\"type\":\"int\"},\"name\":\"位置\"},{\"identifier\":\"LocalTimer\",\"dataType\":{\"specs\":{\"item\":{\"specs\":[{\"identifier\":\"Timer\",\"dataType\":{\"specs\":{\"length\":\"255\"},\"type\":\"text\"},\"name\":\"定时时间\"},{\"identifier\":\"Enable\",\"dataType\":{\"specs\":{\"0\":\"关\",\"1\":\"开\"},\"type\":\"bool\"},\"name\":\"启用\"},{\"identifier\":\"IsValid\",\"dataType\":{\"specs\":{\"0\":\"无效\",\"1\":\"有效\"},\"type\":\"bool\"},\"name\":\"可执行\"},{\"identifier\":\"Targets\",\"dataType\":{\"specs\":{\"length\":\"2048\"},\"type\":\"text\"},\"name\":\"定时动作\"}],\"type\":\"struct\"},\"size\":\"109\"},\"type\":\"array\"},\"name\":\"本地定时\"}],\"identifier\":\"post\",\"method\":\"thing.event.property.post\",\"name\":\"post\",\"type\":\"info\",\"required\":true,\"desc\":\"属性上报\"}]}",
 "message": "success"
}
```


2.1.6 获取物的服务记录

路径	/app/thing/service/data/get
版本号	1.0.3
协议	HTTP/HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	否	物的唯一标识
productKey	String	否	物的产品标识

deviceName	String	否	物的名称
identifier	String	是	服务标识符
start	String	是	开始时间
end	String	是	结束时间
pageSize	Integer	是	单次查询最大条数
ordered	String	是	查询顺序，true-顺序查询，false-倒序查询
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
nextTime	Long	最近调用时间戳
data	List<ServiceInfoDTO>	服务数据列表

服务数据列表 ServiceInfoDTO:

名称	类型	描述
identifier	String	服务标识
name	String	服务名称
time	Long	服务时间
inputData	String	输入参数
outputData	String	输出参数

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.3");
// 设置参数
request.putParam("iotId","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("identifier","value4");
request.putParam("start","value5");
request.putParam("end","value6");
request.putParam("pageSize","value7");
request.putParam("ordered","value8");
request.putParam("lpInstanceId","value9");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/service/data/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "code": 200,
  "data": {
    "list": [
      {
        "identifier": "testServiceSync",
        "inputData": "{\"structArgs\":{\"structchildINT62005ae\":50,\"structchildDOUBLE6d967d2\":50.22,\"structchildDATEaec2a80\":\"1605
786342845\",\"structchildBOOLa05d374\":0,\"structchildFLOAT76c6d83\":50.11,\"structchildTEXT710c059\":\"xiaoyuershi\",\"structchildENUMe
811671\":1},\"enumArgs\":1,\"boolArgs\":0,\"floatArgs\":50.11,\"intArgs\":10,\"dateArgs\":\"1605786342845\",\"doubleArgs\":50.22,\"array
Args\":[1,2,3],\"textArgs\":\"你好helloworld\"}",
        "name": "testServiceSyncName",
        "outputData": "{\"code\":200,\"data\":{\"structArgs\":{\"structchildINT62005ae\":50,\"structchildDOUBLE6d967d2\":50.22,\"structc
hildDATEaec2a80\":\"1605786342845\",\"structchildBOOLa05d374\":0,\"structchildFLOAT76c6d83\":50.11,\"structchildTEXT710c059\":\"xiaoyuer
shi\",\"structchildENUMe811671\":1},\"enumArgs\":1,\"boolArgs\":0,\"floatArgs\":50.11,\"intArgs\":10,\"dateArgs\":\"1605786342845\",\"do
ubleArgs\":50.22,\"arrayArgs\":[1,2,3],\"textArgs\":\"你好helloworld\"},\"id\":\"Akeg8vmMdQ\",\"message\":\"success\",\"version\":\"1.0\
"}",
        "time": 1605786348669
      }
    ],
    "nextTime": 1605786348670,
    "nextValid": false
  },
  "message": "success"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

错误码：

错误码	错误信息	错误描述
400	request error.	系统繁忙

2.1.7 获取物的连接状态

路径	/app/thing/status/get
版本号	1.0.3
协议	HTTP/HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	否	物的唯一标识
productKey	String	否	物的产品标识
deviceName	String	否	物的名称
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

返回结果 data:

名称	类型	描述
deviceName	String	设备名称
status	Integer	设备状态 0 - 未激活， 1 - 在线， 3 - 离线， 8 - 禁用

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.3");
// 设置参数
request.putParam("iotId","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("lpInstanceId","value4");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/status/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "code":200,
    "data":{
        "deviceName":"apigatewaydevice",
        "status":1
    },
    "id":"be4cb9d0-5d71-4e44-bc10-866eb718731c"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

错误码：

错误码	错误信息	错误描述
400	request error.	系统繁忙

2.1.8 获取物的拓扑关系

路径	/app/thing/topo/get
版本号	1.0.0
协议	HTTP/HTTPS
请求方法	POST

授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	否	物的唯一标识
productKey	String	否	物的产品标识
deviceName	String	否	物的名称
pageNo	Integer	是	页面ID
pageSize	Integer	是	分页大小

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
total	整型	总页数
pageNo	整型	当前页
pageSize	整型	每页大小
data	SimpleDeviceDTO	简单的设备信息对象，只包含设备ID和三元组信息

SimpleDeviceDTO :

名称	类型	描述
iotId	String	设备id
deviceSecret	String	设备名称
deviceSecret	String	设备sk
productKey	String	产品pk

2.1.9 获取物的标签列表

路径	/app/thing/prop/get
版本号	1.0.1
协议	HTTP/HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	否	物的唯一标识
productKey	String	否	物的产品标识
deviceName	String	否	物的名称
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
props	String	设备标签属性，json String结构

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 设置参数
request.putParam("iotId","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("lpInstanceId","value4");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/prop/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

2.1.10 获取物的事件timeline数据

路径	/app/thing/event/timeline/get
版本号	1.0.3
协议	HTTP/HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	否	物的唯一标识
productKey	String	否	物的产品标识
deviceName	String	否	物的名称
identifier	String	是	事件标识符
eventType	String	是	事件类型
start	Long	是	timeline起始时间

end	Long	是	timeline结束时间
pageSize	Integer	是	分页大小
ordered	Boolean	是	顺序（true）；逆序（false）查询
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
timestamp	Long	
items	ThingEventDTO	事件数据

事件数据 ThingEventDTO:

名称	类型	描述
iotId	String	物的id
eventCode	String	事件 code
eventName	String	事件名称
eventType	String	事件类型
timestamp	Long	事件发生时间
eventBody	Object	事件Body
batchId	String	批次 id

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.3");
// 设置参数
request.putParam("iotId","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("identifier","value4");
request.putParam("eventType","value5");
request.putParam("start","value6");
request.putParam("end","value7");
request.putParam("pageSize","value8");
request.putParam("ordered","value9");
request.putParam("lpInstanceId","value10");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/event/timeline/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "code":200,
    "data":{
        "items":[{
            "batchId":"c0c0eab124bc4231b25aee934dfa234c",
            "eventBody":{
                "boolArgs":0,
                "floatArgs":1.23,
                "intArgs":4
            },
            "eventCode":"testAlertEvent",
            "eventName":"testAlertEventName",
            "eventType":"alert",
            "iotId":"1gHTxTa1NdKXl6wGE3jx000000",
            "timestamp":1605775253666
        }],
        "timestamp":1605775253666
    },
    "message":"success"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

错误码：

错误码	错误信息	错误描述
400	request error.	系统繁忙

2.1.11 批量获取物的状态

路径	/app/thing/status/batch/get
版本号	1.0.3
协议	HTTP/HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
productKey	String	是	物的产品标识
deviceNames	List<String>	是	物的名称列表，一次最多支持查询10个
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Integer	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	List<ThingStatusInfoDTO>	设备状态信息列表

设备状态信息列表 ThingStatusInfoDTO:

名称	类型	描述
deviceName	String	设备名称
status	Integer	设备状态 0 - 未激活， 1 - 在线， 3 - 离线， 8 - 禁用

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.3");
// 设置参数
request.putParam("productKey","value1");
request.putParam("deviceNames","value2");
request.putParam("lpInstanceId","value3");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/status/batch/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "code": 200,
  "data": [{
    "deviceName": "apigatewaydevice",
    "status": 0
  }],
  "id": "ed663083-c963-49e4-aa36-f0a6d62a6a23"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

错误码：

错误码	错误信息	错误描述
400	request error.	系统繁忙

2.1.12 查询应用物的列表

路径	/app/thing/list
版本号	1.0.4
协议	HTTP/HTTPS
请求方法	POST

授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
tagList	JSONAarry	否	标签列表
pageNo	Integer	否	当前页，从1开始
pageSize	Integer	否	每页大小，最大200
productKeyList	JSONAarry	否	productkeylist
categoryKeyList	JSONAarry	否	品类列表
appId	字符串	否	应用id
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
total	Integer	总页数
pageNo	Integer	当前页
pageSize	Integer	每页大小
data	List<SimpleDeviceInfo>	

SimpleDeviceInfo:

名称	类型	描述
productName	String	产品名称
productKey	String	产品pk

deviceName	String	设备名称
nodeType	Integer	设备网络类型
status	String	设备专题
activeTime	String	设备激活时间
lastOnlineTime	String	设备最近上线时间
createTime	String	设备创建时间
childDeviceCount	Long	子设备数量
iotId	String	设备id

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.4");
// 设置参数
request.putParam("tagList","value1");
request.putParam("pageNo","value2");
request.putParam("pageSize","value3");
request.putParam("productKeyList","value4");
request.putParam("categoryKeyList","value5");
request.putParam("appId","value6");
request.putParam("lpInstanceId","value7");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/list", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
 "code": 200,
 "data": {
  "total": 1,
  "data": [{
   "activeTime": "2019-12-13 13:14:48",
   "productKey": "alQcurjTB1h",
   "nodeType": 0,
   "deviceName": "wjgT_0034",
   "utcCreateTime": "2019-08-05T02:15:20.000Z",
   "productName": "授权设备",
   "utcActiveTime": "2019-12-13T05:14:48.740Z",
   "utcLastOnlineTime": "2019-12-13T05:17:54.315Z",
   "iotId": "ErMmUJAE6nRduVjQ7wi9000101",
   "createTime": "2019-08-05 10:15:20",
   "lastOnlineTime": "2019-12-13 13:17:54",
   "status": "ONLINE"
  }],
  "pageNo": 1,
  "pageSize": 20
 },
 "id": "107595bf12f9465e9530850b0baf60f3"
}
```


异常返回示例：


```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```


错误码：

错误码	错误信息	错误描述
40063	app not exist	应用不存在
40106	saas tenant not exist	saas租户不存在
400	request error.	系统繁忙
40061	query device error	查询设备错误

2.1.13 获取物的属性

路径	/app/thing/properties/get
版本号	1.0.4
协议	HTTP/HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述

iotId	String	否	物的唯一标识
productKey	String	否	物的产品标识
deviceName	String	否	物的名称
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	List<DeviceStatusDTO>	设备属性列表

DeviceStatusDTO:

名称	类型	描述
iotId	String	设备iotId
batchId	String	设备上报的标示。 可以批量上报 通一个批次上报的这个batchid一样
attribute	String	设备属性
group	Integer	设备分组
value	String	属性值
gmtModified	String	修改时间

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.4");
// 设置参数
request.putParam("iotId","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("lpInstanceId","value4");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/properties/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "code": 200,
  "data": [
    {
      "iotId": "1g84blz9FRQOrlqB0OT1000101",
      "gmtModified": 1607338094175,
      "attribute": "Humidity",
      "batchId": "c9bdb91fb7c948d2a0af68f0af96105b",
      "value": 8
    },
    {
      "iotId": "1g84blz9FRQOrlqB0OT1000101",
      "gmtModified": 1607338053955,
      "attribute": "Speed",
      "batchId": "3cc1e8c95ca3426c95f148636a189f87",
      "value": 70
    }
  ],
  "message": "success"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

错误码：

错误码	错误信息	错误描述
400	request error.	系统繁忙

2.2设备控制

2.2.1 设置物的属性

路径	/app/thing/properties/set
版本号	1.0.3
协议	HTTP/HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	否	物的唯一标识
productKey	String	否	物的产品标识
deviceName	String	否	物的名称
items	JSONObject	是	设备属性map，包含属性key及value
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.3");
// 设置参数
request.putParam("iotId","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("items","value4");
request.putParam("lpInstanceId","value5");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/properties/set", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

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

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

错误码：

错误码	错误信息	错误描述
400	request error.	系统繁忙

2.2.2 触发物的服务

路径	/app/thing/service/invoke
版本号	1.0.3
协议	HTTP/HTTPS
请求方法	POST
授权类型	app签名

超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	否	物的唯一标识
productKey	String	否	物的产品标识
deviceName	String	否	物的名称
args	JSONObject	是	服务参数
identifier	String	是	服务标识符
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.3");
// 设置参数
request.putParam("iotId","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("args","value4");
request.putParam("identifier","value5");
request.putParam("lpInstanceId","value6");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/service/invoke", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "code": 200,
  "data": {
    "data": {
      "SyncFileStatus": 0
    },
    "messageId": "854287045"
  },
  "message": "success"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

错误码：

错误码	错误信息	错误描述
400	request error.	系统繁忙

3. 设备消息订阅

设备数据订阅指引

3.4.2.2. 项目管理API

1. 概述

应用服务平台项目的创建和查询API。

2. 接口定义

2.1 新建项目

路径	/open/project/univisal/create
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
name	String	是	项目名称
clientName	String	是	客户名称
description	String	否	描述
startTime	Long	是	开始时间

endTime	Long	是	结束
solutionUuid	String	否	解决方案uuid，智慧社区为98b4181b749f4de9aaef6e9f20cea702
solutionVersionUuid	String	否	解决方案版本uuid，可以填null
coverUrl	List<String>	否	封面url
contractUrl	List<String>	否	协议url

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	String	项目ID

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("name","value1");
request.putParam("clientName","value2");
request.putParam("description","value3");
request.putParam("startTime","value4");
request.putParam("endTime","value5");
request.putParam("solutionUuid","value6");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/open/project/univisal/create", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": "projectId"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2 分页查询项目列表

路径	/open/project/univisal/query
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	8000

请求参数

名称	类型	是否必选	描述
pageNo	Integer	是	页号
pageSize	Integer	是	每页数量

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSONArray	项目列表

项目列表：data

名称	类型	描述
pageNo	Integer	页号
pageSize	Integer	每页数量

total	Integer	总数
dataList	List<ProjectDetail>	项目列表

项目列表：dataList：

名称	类型	描述
projectId	String	项目ID
name	String	项目名称
clientName	String	客户名称
description	String	描述
startTime	Long	开始时间
endTime	Long	结束时间
defaultAppUuid	String	默认应用uuid
defaultAppKey	String	默认应用appKey
rootSpaceId	String	根空间ID
coverUrl	List<String>	封面
contractUrl	List<String>	协议

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("pageNo","value1");
request.putParam("pageSize","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/open/project/univisal/query", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
        "pageNo":1,
        "pageSize":10,
        "dataList":[
            {
                "projectId":"项目id",
                "name":"项目名称",
                "clientName":"客户名称",
                "description":"描述",
                "startTime":1581075936000,
                "endTime":1581075936000,
                "coverUrl":["url","url"],
                "contractUrl":["contractUrl","contractUrl"],
                "rootSpaceId":"rootSpaceId"
            }
            ]
    }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.3 查询项目详情

路径	/open/project/univisal/get
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
projectId	String	是	页号

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

data	JSONObject	响应结果

响应结果 data:

名称	类型	描述
projectId	String	项目ID
name	String	项目名称
clientName	String	客户名称
description	String	描述
startTime	Long	开始时间
endTime	Long	结束时间
defaultAppUuid	String	默认应用uuid
defaultAppKey	String	默认应用appKey
rootSpaceId	String	根空间ID
clusterIdList	List<String>	边缘集群idList
coverUrl	List<String>	封面
contractUrl	List<String>	协议

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("projectId","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/open/project/univisal/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
        "projectId":"项目id",
        "name":"项目名称",
        "clientName":"客户名称",
        "description":"项目描述",
        "startTime":1581075936000,
        "endTime":1581075936000,
        "coverUrl":["url","url"],
        "contractUrl":["url","url"],
        "rootSpaceId":"rootSpaceId",
        "clusterIdList":"clusterIdList"
    }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.4 分页查询解决方案

路径	/open/solution/basicinfo/query
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
pageNo	Integer	是	页号
pageSize	Integer	是	每页数量

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

data	JSONObject	响应结果

响应结果 data:

名称	类型	描述
pageNo	Integer	页号
pageSize	Integer	每页数量
total	Integer	总数
dataList	List<SolutionMetaInfo>	解决方案列表

解决方案列表 dataList:

名称	类型	描述
solutionUuid	String	解决方案uuid
solutionCode	String	解决方案code
solutionName	String	解决方案名称
solutionDesc	String	描述
solutionStatus	String	状态
versionCode	String	版本code
versionUuid	String	版本uuid

示例

请求示例:

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("pageNo","value1");
request.putParam("pageSize","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/open/solution/basicinfo/query", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data":
    {
        "total":100,
        "pageNo":1,
        "pageSize":1,
        "dataList":
        [
            {
                "solutionUuid":"947957393830278589375982947 8303",
                "solutionCode":"testCode",
                "solutionName":"解决方案名称",
                "solutionDesc":"解决方案描述",
                "solutionStatus":"CREATE",
                "versionCode":"1.0.0",
                "versionUuid":"879389492746409372839472651096 48"
            }
        ]
    }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.5 查询项目默认应用appSecret

路径	/open/project/defaultapp/get
版本号	1.0.0

协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
projectId	String	是	项目id

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSONObject	响应结果

响应结果 data:

名称	类型	描述
defaultAppSecret	String	默认应用appSecret
defaultAppKey	String	默认应用appKey

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("projectId","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/open/project/defaultapp/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
        "defaultAppKey":"默认应用appKey",
        "defaultAppSecret":"默认应用appSecret"
    }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

3.4.2.3. 物模型品类定义

本文档旨在明确物模型品类定义以及解释相关概念。

1 概述

1.1 编写目的

本文档是智慧社区项目物模型品类标准。

1.2 名词解释

名词	解释
IoT	物联网

2 品类应用

2.1 与设备关联

- 在阿里云IoT 应用服务平台创建的设备会自动把品类名称和品类编码写入设备标签。 品类名称标签的key为“**CategoryName**”，value为品类名称三级路径

- 品类编码标签的key为“**CategoryCode**”，value为品类编码三级路径。
- 设备能力标签的key为 "**Capability**"，value为能力定义List。

只有基于门禁一体机品类创建的设备才支持设备能力标签，用来标识该设备具备哪些能力

示例： 二维码门禁标签：

"CategoryName"： "智慧社区/人员通行/二维码门禁"

"CategoryCode"： "community/ people/qrcode"

设备能力标签：

"Capability"：["face","qrcode"]

2.2 云端查询设备标签接口

路径	/app/thing/prop/get
版本号	1.0.1
协议	HTTP, HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	否	物的唯一标识
productKey	String	否	物的产品标识
deviceName	String	否	物的名称
lpInstanceId	字符串	否	LP实例ID

返回数据

名称	**类型**	**描述**
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
props	String	设备标签属性，json String结构

示例

请求示例：

```
// https: //github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 设置参数
request.putParam("iotId","value1");
request.putParam("productKey","value2");
request.putParam("deviceName","value3");
request.putParam("lpInstanceId","value4");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/app/thing/prop/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

2.3 边缘查询设备标签接口

参考文档：边缘应用 Link IoT Edge对接

获取分组设备列表Topic：{srcApp}/request/linkedge/getDeviceList 获取网关设备列表Topic：{srcApp}/request/linkedge/{pk}/{dn}/getDeviceList 请求Payload：

```
{
  "requestId": "0bc458b5-47f6-472f-bd88-3624696e9777", # [Required]
  "timestamp": 157121985, #ms [Optional]
  "timeout": 3, # 最少1s,最多120s [Optional: default 4s]
  "version": "1.0" # [Optional]
}
```

获取分组设备列表应答Topic：linkedge/response/{srcApp}/getDeviceList 应答Payload

```
{
  "params": [ #数组，包含所有网关下的子设备
    {
      "gwProductKey": "a1PQ9828qvr", #网关的productkey
      "gwDeviceName": "yinlong_gw_dev1",#网关的devicename
      "code": 0,
      "message": "success",
      "deviceList": [ #设备列表
        {
          "productKey": "a1KRepmC2XJ", #设备的productkey
          "deviceName": "modbus_tcp_dev_1", #设备的devicename
          "localOnline": "false",  #设备与网关的连接状态
          "cloudOnline": "false",  #设备与云端的连接状态
          "activationState": "notActivated",#设备在云端是否被激活
          "nickName": "xxx",  #设备别名
          "tags": [ #设备包含的标签
          ]
        },
        {
          "productKey": "a1ZJTVsqj2y",
          "deviceName": "lightDevNoSpecial",
          "localOnline": "true",
          "cloudOnline": "true",
          "activationState": "activated",
          "nickName": "xxx",  #设备别名
          "tags": [
          ]
        },
        {
          "productKey": "a1ZJTVsqj2y",
          "deviceName": "LightDev2",
          "localOnline": "false",
          "cloudOnline": "false",
          "activationState": "notActivated",
          "nickName": "xxx",  #设备别名
          "tags": [  #设备包含的标签
          {
            "key": "type",   #设备标签的key
            "value": "light" #设备标签的value
          },
          {
            "key": "owner",
            "value": "yinlong"
          },
          {
            "key": "location",
            "value": "master bedroom"
          },
          {
            "key": "color",
            "value": "red"
          }
          ]
        }
      ]
    }
  ],
  "requestId": "0bc458b5-47f6-472f-bd88-3624696e9777",
  "version": "1.0",
  "timestamp": 157225013,
  "code": 0,
  "message": "success"
}
```

获取网关设备列表应答Topic：linkedge/response/`{srcApp}`/{pk}/{dn}/getDeviceList

应答Payload：

```
{
"requestId": "0bc458b5-47f6-472f-bd88-3624696e9777",
"timestamp":  157121985,
"version": "1.0",
"code": 0,
"message": "success",
"params": [       #设备列表
    {
    "productKey": "a1Mtx4p2fHG",   #设备的productkey
    "deviceName": "device1",       #设备的devicename
    "localOnline": "true",         #设备和网关的连接状态
    "cloudOnline": "true",         #设备和云端的连接状态
    "activationState":  "activated/notActivated/activationFailed", #设备在云端的激活状态
    "tag": [
        {
          "key": "coordinate",
          "value": "0: 0"
        }
      ]
    }
  ]
}
```

附录1 智慧社区品类定义

详细定义参见下表：

分组	品类	模型
	name：车辆通行 code：car	name：车辆道闸 code： barke
		name：车牌识别摄像头 code：lprc
		name：费用显示屏 code：screen
	name：人员通行 code：people	name：二维码门禁 code：qrcode
		name：加密二维码门禁 code：rsa_qrcode
		name：人脸门禁 code：face
		name：测温人脸门禁 code：temp_face
		name：蓝牙门禁 code： bluetooth
		name：可视对讲 code：video
		name：刷卡门禁 code：card
		name：梯控设备 code：elevator

name：智慧社区 code：community		name：梯控网关 code：elevators
		name：门禁一体机 code：aio
		name：电梯控制器 code：elc
	name：视频安防 code： security	name：周界报警主机 code：perimeter
		name：普通NVR code：nnvr
		name：智能分析NVR code： invr
		name：人脸识别NVR code： frnvr
		name：普通摄像头 code： ipcamera
		name：违停球相机 code： tptzcamera
		name：人脸抓拍机 code： frcamera
		name：周界相机 code： pmcamera
	name：设备设施 code： eba	name：超声波流量计 code： flowmeter
		name：智能电力仪表 code： powermeter
		name：压力传感器 code： pressure
		name：液位传感器 code： liquid
		name：温湿度传感器 code： temhum
		name：水侵传感器 code： waterlog
		name：温度传感器 code： temper

		name：遥信遥控智能仪表 code： rcii
		name：照明回路 code：light
		name：风机 code：spf
		name：气体感测器 code：gd

附录2 设备能力标签

基于门禁一体机品类创建的设备支持能力标签，用来标识该设备具备哪些能力

例如某个门禁一体机设备具备人脸识别、二维码功能的能力标签定义如下：

Capability： ["face","qrcode"],

目前支持的6种能力值定义如下：

能力值名称	能力值含义
face_temperature	测温：表示该门禁设备具备测温功能
card	刷卡：表示该门禁具备刷卡通行的能力
intercom	可视对讲：表示该门禁具备可视对讲的能力
bluetooth	蓝牙：表示该门禁具备蓝牙通行的能力
face	人脸：表示该门禁具备人脸识别通行的能力
qrcode	二维码：表示该门禁具备二维码识别通行的能力
elevator_gateway	梯控网关：表示该设备为梯控网关
elevator_gateway_with_rule	支持联动规则的梯控网关：表示该设备为支持联动规则的梯控网关

3.4.3. 人行接口

3.4.3.1. 人员账号服务API

1. 概述

创建人员统一身份id（identifyID）

2. 接口定义

2.1 注销账号

路径	/user/account/unregister
版本号	1.0.1
协议	HTTPS
请求方法	POST

授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
identityId	String	是	需要注销的统一身份ID。

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 设置参数
request.putParam("identityId","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/user/account/unregister", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "id": "8d1a665e-ce92-45bf-add8-99d2e818****",
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": ""
}
```

异常返回示例：

```
{
  "id": "8d1a665e-ce92-45bf-add8-99d2e818****",
  "code": 460,
  "message": "identityId is blank!",
  "localizedMsg": null,
  "data": null
}
```

错误码：

错误码	错误信息	错误描述
460	identityId is blank	需要注销的账号不存在

2.2 账号注册

路径	/user/account/register
版本号	1.0.2
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
loginName	String	是	登录名
phone	String	否	手机号
email	String	否	邮件地址
nickName	String	是	用户昵称
password	String	否	密码
avatarUrl	String	否	头像url
creater	String	否	创建者
mobileLocationCode	String	否	手机国际区号

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

data	String	统一身份identityId，即为创建员工的accountId

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.2");
// 设置参数
request.putParam("loginName","value1");
request.putParam("phone","value2");
request.putParam("email","value3");
request.putParam("nickName","value4");
request.putParam("password","value5");
request.putParam("avatarUrl","value6");
request.putParam("creater","value7");
request.putParam("mobileLocationCode","value8");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/user/account/register", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "id": "f1ed2f40-a02e-4b20-a691-21bc3cb17707",
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": "505fop357f0d8299b92664fc91012ae88ed25269"
}
```

异常返回示例：

```
{
  "id": "f1ed2f40-a02e-4b20-a691-21bc3cb17707",
  "code": 2404,
  "message": "手机格式不正确",
  "localizedMsg": null,
  "data": null
}
```

错误码

错误码	错误信息	错误描述
2402	registerAccount failed	账号注册失败
2404	oa error message	手机号格式不正确

2.3 账号冻结

路径	/user/account/freeze
版本号	1.0.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
identityId	String	是	统一身份id

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 设置参数
request.putParam("identityId","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/user/account/freeze", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "id": "8d1a665e-ce92-45bf-add8-99d2e81814f3",
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": ""
}
```

异常返回示例：

```
{
  "id": "8d1a665e-ce92-45bf-add8-99d2e81814f3",
  "code": 460,
  "message": "identityId is blank!",
  "localizedMsg": null,
  "data": null
}
```

错误码

错误码	错误信息	错误描述
460	identityId is blank	冻结的账号不存在

2.4 账号解冻

路径	/user/account/unfreeze
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
identityId	String	是	统一身份id

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("identityId","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/user/account/unfreeze", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止"
}
```

错误码

错误码	错误信息	错误描述
403	request forbidden.	请求被禁止

2.5 根据手机号查找账号信息

路径	/user/account/querybyphone
版本号	1.0.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
phone	String	是	手机号

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
identityId	String	账号id
loginId	String	openaccount Id
loginName	String	登录名
phone	String	手机号
email	String	邮箱地址
nickName	String	昵称
creater	String	创建者
modifier	String	修改者
createAppKey	String	创建appkey
avatarUrl	String	头像地址
phoneLocationCode	String	手机号国家区号

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 设置参数
request.putParam("phone","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/user/account/querybyphone", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
        "createAppKey": "25214645",
        "creater": "郑士林",
        "identityId": "50b7op6575e3453gd345290b4aa7480174495",
        "loginId": "27921931",
        "nickName": "abc",
        "phone": "180274*****"
    }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

错误码

错误码	错误信息	错误描述
403	request forbidden.	请求被禁止

3.4.3.2. 传统门禁服务API

1. 概述

边缘网关门禁服务通过边缘方案集成本地门禁系统，把本地系统、老旧存量系统接入阿里云IoT云，实现设备、系统上云。

方案支持传统的卡式门禁、二维码门禁，为不同类型的门禁提供了标准数据模型和服务模型。这些标准模型可以由其它平台集成，实现系统控制（下行）和业务感知（数据上行）的业务闭环。

2. 接口定义

2.1通用对象

2.1.1 门禁对象DeviceInfo

字段	类型	是否必传	备注
gatewayProductKey	String	如不传gatewayIotId，此处必填	边缘网关设备的productKey
gatewayDeviceName	String	如不传gatewayIotId，此处必填	边缘网关设备的deviceName
gatewayIotId	String	如不传gatewayProductKey和gatewayDeviceName，此处必填	边缘网关的iotId，
iotId	String	是	设备门禁的iotId
direction	String	否	ENTRANCE，EXIT，如不填不更新
extInfo	String	否	设备扩展信息，用于数据模型透出，如不填不更新

2.1.2 人员信息UserDTO

字段	类型	备注
identityId	String	统一身份ID。
name	String	人员称谓。
extInfo	String	自定义信息。

2.1.3 门禁卡权限CardDTO

字段	类型	备注
iotId	String	门禁设备的iotId
expiryTime	String	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

2.1.4 蓝牙卡权限BLECardDTO

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

2.2人员信息管理

2.2.1添加或更新人员信息

路径	/entrance/paas/user/modify
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
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。

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("identityId","value1");
request.putParam("name","value2");
request.putParam("extInfo","value3");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/user/modify", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

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

异常返回示例：

```
{
 "id": "4a70179d-47d8-4fdf-9067-8a5eedf63483",
 "code": 500,
 "message": "system err",
 "localizedMsg": "系统错误",
 "data": null
}
```

2.2.2删除人员信息

路径	/entrance/paas/user/delete
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
identityId	String	是	统一身份ID。

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("identityId","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/user/delete", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

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

异常返回示例：

```
{
  "id":"4a70179d-47d8-4fdf-9067-8a5eedf63483",
  "code": 403,
  "message": "user does not exist",
  "localizedMsg": "用户不存在",
  "data":null
}
```

2.2.3 查询人员信息

路径	/entrance/paas/user/get
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
identityId	String	是	统一身份ID。

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
identityId	String	统一身份id
name	String	用户姓名
extInfo	String	用户扩展信息

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("identityId","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/user/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

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

异常返回示例：

```
{
 "id": "4a70179d-47d8-4fdf-9067-8a5eedf63483",
 "code": 500,
 "message": "system err",
 "localizedMsg": "系统错误",
 "data": null
}
```

2.2.4 人员列表

路径	/entrance/paas/user/query
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
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

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSONObject	分页数据对象PageDTO

分页数据对象PageDTO:

名称	类型	描述
pageSize	Integer	传入的pageSize
pageNo	Integer	传入的pageNo
total	Long	总记录数
data	JSONArray	数据列表

数据列表：

名称	类型	描述
identityId	String	统一身份id
name	String	用户姓名
extInfo	String	用户扩展信息

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("pageSize","value1");
request.putParam("pageNo","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/user/query", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
 "id": "4a70179d-47d8-4fdf-9067-8a5eedf63483",
 "code": 200,
 "message": null,
 "localizedMsg": null,
 "data": {
  "total": 1,
  "pageNo": 1,
  "pageSize": 10,
  "data": [{
   "identityId": "xxx",
   "name": "xxx",
   "extInfo": {}
  }]
 }
}
```

异常返回示例：

```
{
 "id": "4a70179d-47d8-4fdf-9067-8a5eedf63483",
 "code": 500,
 "message": "system err",
 "localizedMsg": "系统错误",
 "data": null
}
```

2.3 刷卡&蓝牙门禁权限管理

2.3.1 批量设置门禁卡权限

路径	/entrance/paas/perm/card/batchmodify
版本号	1.0.2
协议	HTTPS
请求方法	POST

授权类型	app签名
超时时间	25000

请求参数

名称	类型	是否必选	描述
cards	JSONArray	是	待创建门禁卡权限列表，上限200

待创建门禁卡权限列表 cards:

名称	类型	是否必选	描述
iotId	String	是	门禁设备的iotId
cardId	String	是	门禁卡id
identityId	String	是	统一身份ID
expiryTime	String	是	超时时间，时间格式 yyyy-MM-dd HH:mm:ss

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSONArray	响应结果。处理失败的记录，建议可以依照这个列表重新添加

响应结果 data

名称	类型	描述
iotId	String	门禁设备的iotId
cardId	String	门禁卡id
identityId	String	统一身份ID
code	Integer	错误码
message	String	错误信息

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.2");
// 如果需要登录，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("cards","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/perm/card/batchmodify", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
  "code": 200,
  "message": null,
  "localizedMsg": null,
    "data": [{
    "identityId": "50acopfd3a94900494085ed2699e052432c34452",
    "iotId":"p2OsgId2N2kXMWThRCQP000101",
    "cardId":"3d8deae9402d38adfji38",
    "code": 460,
    "message": "device offline"
  }]
}
```

异常返回示例：

```
{
    "code":403,
    "message":"request forbidden.",
    "localizedMsg":"请求被禁止"
}
```

2.3.2 删除刷卡门禁上某用户的权限

路径	/entrance/paas/perm/card/batchdelete
版本号	1.0.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	25000

请求参数

名称	类型	是否必选	描述
list	JSONArray	是	待删除对象列表 PermDTO

待删除对象列表PermDTO

名称	类型	是否必选	描述
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

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSONArray	删除失败的列表 ModifyResultDTO

删除失败的列表ModifyResultDTO

名称	类型	描述
iotId	String	门禁设备的iotId
cardId	String	门禁卡id
identityId	String	统一身份ID
code	Integer	错误码
message	String	错误信息

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 如果需要登录，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("list","valuel");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/perm/card/batchdelete", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "code": 200,
    "data": [
        {
            "cardId": "3d8deae9402d38adfji38",
            "identityId": "50acopfd3a94900494085ed2699e052432c34452",
            "iotId": "p2OsgId2N2kXMWThRCQP000101"
        }
    ],
    "message": "success"
}
```

异常返回示例：

```
{
    "code":403,
    "message":"request forbidden.",
    "localizedMsg":"无效的设备ID"
}
```

2.3.3 查询用户的门禁卡权限信息

路径	/entrance/paas/user/perm/query
版本号	1.0.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	20000

请求参数

名称	类型	是否必选	描述
permQueryList	JSONArray	是	需要查询的权限列表，由用户、卡、设备三元组构成

需要查询的权限列表 permQueryList

名称	类型	是否必选	描述
iotId	String	是	设备id
identityId	String	是	用户id
cardId	String	是	卡号

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSONObject	分页数据对象PermDTO

分页数据对象PermDTO：

名称	类型	描述
identityId	字符串	用户id
iotId	字符串	设备id
cardId	字符串	卡号
status	字符串	权限状态

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 设置参数
request.putParam("permQueryList","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/user/perm/query", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data":[{
    "iotId": "50acopfd3a94900494085ed2699e052432c34452",
    "identityId": "p2OsgId2N2kXMWThRCQP000101",
    "cardId":"3d8deae9402d38adfji38"
  }]
```

异常返回示例：

```
{
    "code":403,
    "message":"request forbidden.",
    "localizedMsg":"请求被禁止"
}
```

2.3.4 查询门禁设备上的权限信息

路径	/entrance/paas/device/perm/query
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述

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

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSONObject	分页数据对象PageDTO

分页数据对象PageDTO

名称	类型	描述
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

门禁卡权限对象列表：

名称	类型	描述
identityId	String	统一身份ID
iotId	String	设备ID
cardId	String	卡ID
expiryTime	String	过期时间

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("iotId","value1");
request.putParam("pageNo","value2");
request.putParam("pageSize","value3");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/device/perm/query", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
  "code": 200,
  "message": null,
  "localizedMsg": null,
    "data":{
        "total":1,
        "pageSize":10,
        "pageNo":1,
        "data":[
            {
                "identityId":"50acopfd3a94900494085ed2699e052432c34452",
                "iotId":"p2OsgId2N2kXMWThRCQP000101",
                "expiryDate":"2020-03-04 18:00:00",
                "cardId":"3d8deae9402d38adfji38"
            }
        ]
    }
}
```

异常返回示例：

```
{
    "code":403,
    "message":"request forbidden.",
    "localizedMsg":"请求被禁止"
}
```

2.4 二维码门禁通行方式

2.4.1生成二维码

路径	/homelink/common/qrcode/generate
版本号	1.0.1
协议	HTTPS

请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
identityId	String	是	申请二维码人员的统一身份ID
codeLength	整型	否	二维码字符串长度

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	String	二维码

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 设置参数
request.putParam("identityId","value1");
request.putParam("codeLength","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/homelink/common/qrcode/generate", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": "24DD9END8CD3"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.4.2 二维码权限配置

路径	/entrance/paas/perm/qrcode/config
版本号	1.0.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	20000

请求参数

名称	类型	是否必选	描述
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
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	最大刷码次数，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

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。

message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSONArray	权限配置结果

权限配置结果 data:

名称	类型	描述
code	int	返回码
message	String	错误消息
iotId	String	设备iotId

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 如果需要登录，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("qrCode","value1");
request.putParam("iotIds","value2");
request.putParam("identityId","value3");
request.putParam("expiryTime","value4");
request.putParam("effectiveTime","value5");
request.putParam("maxScanTimes","value6");
request.putParam("maxScanScope","value7");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/perm/qrcode/config", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": [
        {
          "iotId": "p2OsgId2N2kXMWThRCQP000101",
          "code": "200"
        }
      ]
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.4.3 二维码权限删除

路径	/entrance/paas/perm/qrcode/remove
版本号	1.0.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	20000

请求参数

名称	类型	是否必选	描述
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

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	设备结果列表

设备结果列表 data:

名称	类型	描述
code	Int	错误码，200为成功
message	String	错误信息
iotId	String	设备ID

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 如果需要登录，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("qrCode","value1");
request.putParam("iotIds","value2");
request.putParam("identityId","value3");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/perm/qrcode/remove", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": [
        {
            "code": 200,
            "iotId": "xxxx"
        }
        ]
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.5 远程控制开门

路径	/cloud/thing/service/invoke
版本号	1.0.2
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	15000

请求参数

名称	类型	是否必选	描述
iotId	String	是	门禁设备的iotId
identifier	String	是	固定值“remoteOpen”
args	JSNOObject	是	固定值{}
productKey	String	是	产品的Key，设备证书信息之一。
deviceName	String	是	设备的名称，设备证书信息之一。

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.2");
// 设置参数
request.putParam("iotId","p2OsgId2N2kXMWThRCQP000101");
request.putParam("identifier","remoteOpen");
request.putParam("args", {});
request.putParam("productKey","value4");
request.putParam("deviceName","value5");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/cloud/thing/service/invoke", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "code": 200,
  "data": null,
  "message": "success",
  "localizedMsg": ""
}
```

3. 数据模型定义

3.1 门禁通行事件

模型ID：iot_entrance_event_model

模型版本：1.1

属性列表：

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
通行类型	type	String	通行类型 - CARD门禁卡 - PASSWORD密码 - QRCODE二维码 - BleCARD蓝牙门禁卡	是

密码号码	number	String	密码号码	否
门禁卡号	cardId	String	门禁卡号，只有type是CARD的时候有效	否
通行方向	direction	String	通行方向，ENTRANCE 进, EXIT 出	否
用户扩展信息	userExtInfo	String	用户扩展信息	否
设备扩展信息	devExtInfo	String	设备扩展信息	否

事件扩展信息	eventExtInfo	String	事件扩展信息
			{
			"code": 0, //0为通行成功
			"message": "xxx",//通行失败原因
			"remainScanTimes":1//二维码剩余刷码次数
			}
				否

项目ID	projectId	String	项目ID，如为租户数据则为空	否

3.2 通行失败事件码说明

通行结果（code）	结果说明（message）	
0	成功	success
101	二维码未生效	QR code does not take effect
102	二维码已过期	QR code has expired
103	二维码不存在	QR code does not exist
104	二维码刷码次数用尽	QR code have be used up
201	传统卡号未生效	Traditional card number does not take effect
202	传统卡号已过期	Traditional card number has expired
203	传统卡号不存在	Traditional card number does not exist
301	蓝牙卡号未生效	Bluetooth card number does not take effect
302	蓝牙卡号已过期	Bluetooth card number has expired
303	蓝牙卡号不存在	Bluetooth card number does not exist

3.4.3.3. 人脸门禁服务API

1. 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。企业可以基于人脸门禁服务，结合阿里云IoT平台的空间管理功能，快速实现自己的SaaS平台。人脸门禁服务的云端API同时兼容边缘方案与端侧方案。

2. 接口定义

2.1 人脸图片管理

2.1.1 保存人脸图片

保存单个用户的人脸图片，支持不同用户体系（详见入参），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。

路径	/face/paas/image/save
版本号	1.2.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
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：用identityId作为userId：用手机号作为userId
userId	String	是	用户ID（identityId或手机号）
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
userRole	整型	否	用户扩展字段：用户角色
mobile	字符串	否	用户扩展字段：用户手机号

policy	String	否	人脸匹配后人员通行策略 PERMISSION（默认） - 允许 DENY - 禁止通行

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.2.0");
// 设置参数
request.putParam("userType","value1");
request.putParam("userId","value2");
request.putParam("imageUrl","value3");
request.putParam("imageBase64","value4");
request.putParam("faceExtInfo","value5");
request.putParam("userName","value6");
request.putParam("expiredTime","value7");
request.putParam("userExtInfo","value8");
request.putParam("userRole","value9");
request.putParam("mobile","value10");
request.putParam("policy","value11");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/face/paas/image/save", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.1.2 删除人脸图片

删除单个用户的人脸图片，支持不同用户体系（详见入参）。

路径	/face/paas/image/delete
版本号	1.2.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
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； 【YUNDUN】云盾自定义用户体系； 【PHONE】用手机号作为userId
userId	String	是	用户ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.2.0");
// 设置参数
request.putParam("userType","value1");
request.putParam("userId","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/face/paas/image/delete", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.1.3 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。

路径	/face/paas/image/get
版本号	1.2.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述

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； 【YUNDUN】云盾自定义用户体系； 【PHONE】用手机号作为userId
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

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
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
faceId	String	人脸ID
userId	String	用户ID
userType	String	用户类型
userName	String	用户名称
expireTime	String	超时时间
policy	String	通行策略

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.2.0");
// 设置参数
request.putParam("userType","value1");
request.putParam("userId","value2");
request.putParam("imageFormat","value3");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/face/paas/image/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
        "imageUrl":"http://yyy/1.jpg",
        "faceId":"xxx",
        "userId":"xxx",
        "userType":"PHONE",
        "userName":"xxx",
        "expireTime":"yyyy-MM-dd HH:mm:ss",
        "policy":"PERMISSION"
    }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2 人脸权限管理

2.2.1 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。支持从设备维度和空间维度进行下发。

路径	/face/paas/permission/add
版本号	1.2.0
协议	HTTPS

请求方法	POST
授权类型	app签名
超时时间	25000

请求参数

名称	类型	是否必选	描述
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； 【YUNDUN】云盾自定义用户体系； 【PHONE】用手机号作为userId
userIdList	JSONArray	是	用户ID列表 【单次下发： 最多支持10个用户】
scopeType	String	是	人脸信息下发的目标范围类型：：将人脸信息下发到空间下绑定的所有人脸识别设备：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

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.2.0");
// 设置参数
request.putParam("userType","value1");
request.putParam("scopeType","value2");
request.putParam("scopeIdList","value3");
request.putParam("userIdList","value4");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/face/paas/permission/add", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": "success"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2.2 删除人脸权限

将已保存的用户人脸权限从设备端删除(边缘端人脸删除，但云端维护的人脸图片不会删除)，使设备无权限识别对应的人脸用户。支持从设备维度和空间维度进行删除。

路径	/face/paas/permission/delete
版本号	1.2.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	25000

请求参数

名称	类型	是否必选	描述
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； 【YUNDUN】云盾自定义用户体系； 【PHONE】用手机号作为userId
userIdList	JSONArray	是	用户ID列表
scopeType	String	是	人脸信息下发的目标范围类型：：将人脸信息下发到空间下绑定的所有人脸识别设备：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

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.2.0");
// 设置参数
request.putParam("userType","value1");
request.putParam("userIdList","value2");
request.putParam("scopeType","value3");
request.putParam("scopeIdList","value4");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/face/paas/permission/delete", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2.3 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）。

路径	/face/paas/permission/querybyuser
版本号	1.0.4
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
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； 【YUNDUN】云盾自定义用户体系； 【PHONE】用手机号作为userId
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	JSON	否	权限状态

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

data	JSON	响应结果

响应结果 data:

名称	类型	描述
userType	String	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：用identityId作为userId ： 用手机号作为userId
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

设备列表 deviceList:

名称	类型	描述
iotId	字符串	设备id
syncTime	字符串	下发时间
status	字符串	下发状态: toBeTransferred-等待传输；transferring-传输中； transferred-传输完成； deviceOffline-设备离线或者设备异常； faceCheckError-人脸落库失败（可能照片不符合要求）； transferDeleted-人脸照片在底库中被删除； deleteFailed-删除照片失败
deviceNickName	字符串	设备昵称
cron	字符串	有效期表达式

示例

请求示例:

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.4");
// 设置参数
request.putParam("userType","value1");
request.putParam("userId","value2");
request.putParam("deviceListPageNo","value3");
request.putParam("deviceListPageSize","value4");
request.putParam("statusList","value5");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/face/paas/permission/querybyuser", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": "success",
    "localizedMsg": "success",
    "data": {
    "userType":"OPEN",
    "userId":"xxx",
    "deviceListTotal":1,
    "deviceListPageNo":1,
    "deviceListPageSize":20,
    "deviceList":
    [
        {
      "iotId":"zzz",
      "syncTime":"2019-02-28 19:00:00",
      "syncStatus":"transferred"
        }
    ]
  }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.3 人脸质量评估

2.3.1 人脸质量评估接口

路径	/face/quality/query

版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
faceUrl	String	是	人脸图片地址，Http
provider	String	是	算法提供商，枚举值： Alipay，IoT，Damo

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
requestId	string	request_id
faceNum	int	检测出来的人脸个数
faceRect	int数组	返回人脸矩形框，分别是[left, top, width, height], 如有多个人 脸，则依次顺延，返回矩形框。如有两个人脸则返回[left1, top1, width1, height1, left2, top2, width2, height2]
faceProb	float数组	返回人脸概率, 0-1之 间，如有多个人脸，则依次顺延。如有两个人脸则返回[face_prob1, face_prob2]
pose	float数组	返回人脸姿态[yaw, pitch, roll]， yaw为左右角度，取值[-90, 90]，pitch为上下角 度，取值[-90, 90]， roll为平面旋转角度， 取值[-180, 180]，如有 多个人脸，则依次顺延

landmarkNum	int	特征点数目，目前固定为5点(顺序：左眼，右眼，鼻子6点，左嘴角，右嘴角)
landmark	float数组	特征点定位结果，每个人脸返回一组特征点位置，表示方式为（x0, y0, x1, y1, ...）；如有多个人脸，则依次顺延，返回定位浮点数
gender	int数组	0：女性，1：男性，如有多个人脸，则依次返回性别
age	int数组	年龄0-100，如有多个人脸，依次返回年龄
expression	int数组	2种表情，0：中性，1：微笑
glass	int数组	是否佩戴眼镜，0：无眼镜，1：有眼镜

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登录，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("provider","value1");
request.putParam("faceUrl","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/face/quality/query", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
  "code": 200,
  "data": {
    "iris": null,
    "glass": [
      0
    ],
    "expression": [
      0
    ],
    "gender": [
      1
```

```
  ],
  "pose": [
    -1,
    -2,
    1
  ],
  "faceNum": 1,
  "faceProb": [
    1
  ],
  "requestId": "02393494-897F-478A-BF93-72A66340A0FC",
  "faceRect": [
    170,
    138,
    245,
    328
  ],
  "landmarkNum": 105,
  "landmark": [
    189.2341766357422,
    244.83270263671875,
    263.9117431640625,
    235.50816345214844,
    223.74264526367188,
    213.3116912841797,
    225.18899536132812,
    231.9640350341797,
    195.27978515625,
    229.09661865234375,
    207.72967529296875,
    217.909423828125,
    240.44154357910156,
    214.62469482421875,
    256.0074462890625,
    220.8974151611328,
    200.65699768066406,
    238.69119262695312,
    212.3941650390625,
    233.17869567871094,
    238.10061645507812,
    233.05096435546875,
    250.9724884033203,
    234.63412475585938,
    314.1351318359375,
    238.15829467773438,
    389.20306396484375,
    246.52294921875,
    355.11920166015625,
    216.89163208007812,
    353.19873046875,
    234.36343383789062,
    322.86474609375,
    224.12869262695312,
    338.559326171875,
    218.57135009765625,
    371.3887023925781,
    220.08120727539062,
    384.1372375488281,
    230.62918090820312,
    327.1981506347656,
    237.35391235351562,
    340.1736755371094,
    235.63998413085938,
    366.2593994140625,
    234.1768341064453,
    378.302001953125,
    239.27926635742188,
    211.5428466796875,
    266.6617126464844,
    254.77716064453125,
    268.15924072265625,
    216.42266845703125,
    262.96588134765625,
    221.82733154296875,
    260.094482421875,
    227.68597412109375,
    258.33538818359375,
    233.78591918945312,
    257.9173583984375,
```

```
239.88876342773438,
258.3575744628906,
245.7425994873047,
260.1201171875,
250.83941650390625,
263.48358154296875,
216.4285430908203,
269.462646484375,
221.8634033203125,
270.9305725097656,
227.40357971191406,
271.9388732910156,
233.01028442382812,
272.46429443359375,
238.6412811279297,
272.4317321777344,
244.25062561035156,
271.9342041015625,
249.73974609375,
270.6767272949219,
323.91925048828125,
269.5917663574219,
367.63409423828125,
268.4336853027344,
328.056640625,
264.9695739746094,
333.44500732421875,
261.895263671875,
339.3934631347656,
260.11883544921875,
345.53680419921875,
259.2233581542969,
351.7408142089844,
259.4993896484375,
357.7361755371094,
261.11572265625,
363.1594543457031,
264.13311767578125,
328.9476623535156,
272.4422607421875,
334.52117919921875,
273.97210693359375,
340.22454833984375,
274.9097595214844,
345.99761962890625,
275.195556640625,
351.76171875,
274.76519775390625,
357.43511962890625,
273.6596984863281,
362.8805847167969,
271.72186279296875,
288.5384826660156,
271.520751953125,
286.1998291015625,
331.32574462890625,
287.2810974121094,
301.2675476074219,
286.2149658203125,
347.75140380859375,
255.48033142089844,
344.3111572265625,
318.2540588378906,
347.0624084472656,
245.88827514648438,
392.71600341796875,
328.6979064941406,
392.91510009765625,
253.7222137451172,
392.2544250488281,
320.916259765625,
392.8489685058594,
287.09417724609375,
379.467529296875,
276.6300048828125,
376.14068603515625,
298.02484130859375,
376.5364685058594,
260.374755859375,
382.62030029296875,
```

```
      382.62030029298875,
      314.23968505859375,
      382.96148681640625,
      252.67446899414062,
      387.02081298828125,
      267.9718017578125,
      378.063720703125,
      306.6196594238281,
      378.546630859375,
      321.9701843261719,
      387.22088623046875,
      286.06634521484375,
      415.8150634765625,
      264.0054931640625,
      408.02960205078125,
      309.4367980957031,
      408.4892883300781,
      254.72647094726562,
      400.63348388671875,
      274.3739013671875,
      413.792724609375,
      298.3180847167969,
      413.9489440917969,
      319.3194580078125,
      401.0159912109375,
      286.95135498046875,
      392.3644714355469,
      287.06842041015625,
      394.86602783203125,
      270.33563232421875,
      389.91778564453125,
      270.4597473144531,
      392.7579345703125,
      303.978271484375,
      390.0447082519531,
      303.95709228515625,
      393.2314453125,
      261.9865417480469,
      390.7685241699219,
      262.09649658203125,
      392.370849609375,
      278.65557861328125,
      391.0590515136719,
      278.7314147949219,
      394.0658874511719,
      295.47149658203125,
      391.2536315917969,
      295.53363037109375,
      394.26373291015625,
      312.50531005859375,
      391.0933837890625,
      312.430419921875,
      392.76031494140625,
      167.90496826171875,
      258.4239501953125,
      414.7978515625,
      266.3011779785156,
      282.4410095214844,
      469.7729797363281,
      182.39126586914062,
      380.735107421875,
      392.3157653808594,
      389.7607116699219,
      170.78298950195312,
      319.3047790527344,
      407.664306640625,
      328.16729736328125,
      223.0463104248047,
      438.533203125,
      345.96502685546875,
      445.37750244140625
    ],
    "class": "com.alibaba.iot.tgc.base.dto.PicQualityDTO",
    "age": [
      28
    ]
  },
  "message": "damo face quality success",
  "localizedMsg": null,
  "class": "com.aliyun.iotx.common.base.service.IoTxResult"
```

```
    class : com.aliyun.iotx.common.base.service.IotXResult
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

3. 数据模型定义

3.1 刷脸记录

模型ID：iotx_face_record_model

模型版本：1.4

属性列表：

属性名称	属性标识符	数据类型	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_ID、OPEN_ID
设备扩展信息	deviceExtInfo	String	设备扩展信息，（边缘端上报的事件包含设备PK/DN。 示例：{"productKey":"a11q0MdjQ6d", "deviceName":"iot_face_dev"}）
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

通行事件信息	eventExtInfo	String	事件扩展信息
			{
			"code": 0, //0为通行成功
			"message": "xxx" //通行失败原因
			}

3.2 通行失败事件码说明

通行结果（code）	结果说明（message）	
0	成功	success
501	人脸信息未生效	Face auth does not take effect
502	人脸信息已过期	Face information has expired
503	人脸库为空	Face library is empty
504	人脸照片特征值提取失败	Face feature extraction failed
505	人脸匹配的相似度低于阈值	Face matching similarity is less than the threshold
506	黑名单用户	The user is in blacklist
507	人脸识别超时	Face recognition timeout
10000	未知错误	Unknown error

3.3 门禁服务业务数据推送

说明：门禁服务在处理具体业务时，一些过程信息对上层应用来说也是必要的，如人脸门禁的下发过程，涉及人脸状态的变更，从初始状态到完成，或异常，需要推送给业务平台，做后续的补发、重发处理。一些设备无响应状态，也透出给上层应用，判断是否重启设备。

模型ID: BizDataPushModel

模型版本：1.1

属性列表:

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
消息级别	eventLevel	String	64	是	消息的级别 • INFO • WARN • ERROR
消息描述信息	eventText	String	2048		推送文本，如当前消息的处理建议

下面分别说明业务类型bizType为FACE_SYNC_STATUS，DEVICE_SYNC_ERROR，FACE_DEVICE_PERMS时的data域或dataUrl的定义

3.3.1. 人脸同步状态

- bizType = FACE_SYNC_STATUS
- 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
人员ID	identityId	String	是
人脸ID	faceId	String	是
状态	status	String	是
前置状态	preStatus	String	

3.3.2. 设备同步异常

- bizType = DEVICE_SYNC_ERROR
- 当同步设备出现错误时，推送相关错误信息。

data域

属性名称	属性标识符	类型	非空
设备ID	iotId	String	是
错误结果	result	String	是
错误码	code	Integer	

错误信息	message	String	

3.3.3. 设备上报数据

- bizType = FACE_DEVICE_PERMS
- 当查询设备全量数据时，同步推送数据。

dataUrl域，设备上报数据

属性名称	属性标识符	类型	非空
成功列表	addFaceInfos	Array	是
	faceId		
失败列表	failedFaceInfos	Array	是
	faceId		

3.4.3.4. 可视对讲服务API

1. 概述

可视门禁对讲服务，临时访客可通过云对讲呼叫业主：业主可通过室内机、手机APP或电话与访客做双向语音对讲并开门。

2. 接口定义

2.1 SIP服务器管理

2.1.1 查询服务器列表

路径	/entrance/paas/sipserver/query
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数：无

名称	类型	是否必选	描述

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	SIP服务器列表

SIP服务器列表 data:

名称	类型	描述
serverId	String	服务器id
hostIp	String	服务器IP地址

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
// 无参数
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/sipserver/query", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
      [
      {
        "serverId": "xxx",
        "hostIp": "1.1.1.1"
      },
      {
        "serverId": "xxx",
        "hostIp": "1.1.1.2"
      }
      ]
    }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.1.2 绑定服务器

注：不同服务器下创建的sip账号不能互通

路径	/entrance/paas/sipserver/bind
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
serverId	String	是	服务器id

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("serverId","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/sipserver/bind", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2 SIP管理

2.2.1 创建SIP账号

需要指定type，1为门禁对讲机类型，用于门禁对讲机，当一路APP正在调看门禁对讲机时，若发起第二路APP调看，则会加入会话，可以实现多个APP同时调看同一台门禁对讲机；0为APP类型，用于APP，当SIP号在通话时，则无法接入。

路径	/entrance/paas/sipservice/sip/add
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
count	Integer	是	创建数量，最大100
type	Integer	是	类型，0 - APP， 1 - 门禁机

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
sipNumber	String	SIP号,长度不超过50个字符

sipPassword	String	SIP密码,长度不超过50个字符
sipHost	String	SIP服务器地址
sipPort	Integer	SIP服务器端口
type	Integer	类型，0 - APP， 1 - 门禁机
createTime	Long	创建时间戳（单位毫秒）

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("count","value1");
request.putParam("type","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/sipservice/sip/add", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": [
    {
      "sipNumber": "obog97v34g",
      "sipPassword": "o3yira",
      "sipHost": "sip.iot.aliyun.com",
      "sipPort": 6050,
      "type": 1,
      "createTime":1577085030000
    }
  ]
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2.2 删除SIP账号

路径	/entrance/paas/sipservice/sip/delete
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
sipNumberList	JASONArray<String>	是	待删除SIP号列表

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("sipNumberList","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/sipservice/sip/delete", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2.3 查询SIP账号详情

路径	/entrance/paas/sipservice/sip/detail/get
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述

sipNumber	String	是	SIP号

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
sipNumber	String	SIP号,长度不超过50个字符
sipPassword	String	SIP密码,长度不超过50个字符
sipHost	String	SIP服务器地址
sipPort	Integer	SIP服务器端口
type	Integer	类型，0 - APP， 1 - 门禁机
createTime	Long	创建时间戳（单位毫秒）

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("sipNumber","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/sipservice/sip/detail/get", request, true, headers)
;

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
    "sipNumber": "obog97v34g",
    "sipPassword": "o3yira",
    "sipHost": "sip.iot.aliyun.com",
    "sipPort": 6050,
    "type": 1,
    "createTime":1577085030000
  }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

若返回错误为SYS_0002，表示SIP号无效或不存在

2.2.4 更新SIP账号密码

目前一个SIP账号同时只允许在一个客户端登录，因此针对可能在多端同时登录的SIP账号，请在登录SIP账号前，调用该接口更新密码，并返回新密码，则之前的旧密码失效。

路径	/entrance/paas/sipservice/sip/password/update
版本号	1.0.0
协议	HTTPS

请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
sipNumber	String	是	SIP号

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
sipPassword	String	SIP密码,长度不超过50个字符

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("sipNumber","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/sipservice/sip/password/update", request, true, hea
ders);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
        "sipPassword": "12345678"
    }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2.5 创建SIP组

路径	/entrance/paas/sipservice/group/add
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
count	Integer	是	创建数量，最大100

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
groupID	String	SIP组ID，长度不超过50个字符
createTime	Long	创建时间戳（单位毫秒）

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("count","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/sipservice/group/add", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": [
      {
      "groupID": "obog97v34g",
      "createTime":1577085030000
      },
      {
      "groupID": "abcg97v34g",
      "createTime":1577085030000
      }
    ]
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2.6 删除SIP组

路径	/entrance/paas/sipservice/group/delete
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
groupList	JASONArray<String>	是	待删除SIP组号列表

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("groupList","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/sipservice/group/delete", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2.7 SIP账号分组

只有SIP账号type为0，即APP端使用的SIP账号，才能支持分组。

路径	/entrance/paas/sipservice/group/sip/add
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
groupID	String	是	SIP组ID
sipNumberList	JASONArray<String>	是	SIP号列表

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("groupID","value1");
request.putParam("sipNumberList","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/sipservice/group/sip/add", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2.8 SIP账号从组里移除

路径	/entrance/paas/sipservice/group/sip/delete
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
groupID	String	是	SIP组ID
sipNumberList	JASONArray<String>	是	SIP号列表

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("groupID","value1");
request.putParam("sipNumberList","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/sipservice/group/sip/delete", request, true, header
s);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.2.9 查看组下SIP账号列表

路径	/entrance/paas/sipservice/group/sip/list
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
groupID	String	是	SIP组ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
sipNumber	String	SIP号,长度不超过50个字符
sipPassword	String	SIP密码,长度不超过50个字符
sipHost	String	SIP服务器地址
sipPort	Integer	SIP服务器端口
type	Integer	类型，0 - APP， 1 - 门禁机
createTime	Long	创建时间戳（单位毫秒）

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("groupID","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/sipservice/group/sip/list", request, true, headers)
;

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": [{
      "sipNumber": "obog97v34g",
      "sipPassword": "o3yira",
      "sipHost": "sip.iot.aliyun.com",
      "serverPort": 6050,
      "type": 0,
      "createTime":1577085030000
    },
    {
      "sipNumber": "tb2yz0ca17",
      "sipPassword": "p14lvq",
      "sipHost": "sip.iot.aliyun.com",
      "serverPort": 6050,
      "type": 0,
      "createTime":1577085030000
    }]
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.3 门禁对讲机SIP同步

2.3.1 颁发门禁对讲机SIP Number

路径	/entrance/paas/videointercom/sipnumber/sync

版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
iotId	String	是	门禁对讲机iotId
sipNumber	String	是	SIP Number
password	String	是	SIP 密码
host	String	是	SIP 服务器地址
port	String	是	SIP 服务器端口

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("iotId","value1");
request.putParam("sipNumber","value2");
request.putParam("host","value3");
request.putParam("port","value4");
request.putParam("password","value5");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/videointercom/sipnumber/sync", request, true, heade
rs);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.3.2 查询门禁对讲机SIP Number

路径	/entrance/paas/videointercom/sipnumber/get
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
iotId	String	是	门禁对讲机iotId

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
sipNumber	String	SIP Number
host	String	SIP 服务器地址
port	String	SIP 服务器端口

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("iotId","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/videointercom/sipnumber/get", request, true, header
s);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
      "sipNumber": "xxx",
       "host": "x.x.x.x",
       "port": "8080"
 }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.4 SIP组与房间映射

2.4.1 添加SIP组与房间编号映射关系

路径	/entrance/paas/videointercom/map/add
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
iotId	String	是	门禁对讲机iotId
maps	JSONArray	是	映射关系列表

映射关系列表 maps:

名称	类型	是否必选	描述
roomId	String	是	房间ID 规则：期号-楼幢号-单元号-房号 例如： 1-9-3-1202 1-10-3A-1709 1-5-767-801 最大位数：1位-3位-4位-5位，规则数字+字母。 期号默认为1。
groupId	String	是	SIP组ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("iotId","value1");
request.putParam("maps","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/videointercom/map/add", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.4.2 删除SIP组与房间编号映射关系

路径	/entrance/paas/videointercom/map/delete

版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
iotId	String	是	门禁对讲机iotId
maps	JSONArray	是	映射关系列表

映射关系列表 maps:

名称	类型	是否必选	描述
roomId	String	是	房间ID 规则：期号-楼幢号-单元号-房号 例如： 1-9-3-1202 1-10-3A-1709 1-5-767-801 最大位数：1位-3位-4位-5位，规则数字+字母。 期号默认为1。
groupId	String	是	SIP组ID

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("iotId","value1");
request.putParam("maps","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/videointercom/map/delete", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.4.3 查询SIP组与房间编号映射关系

路径	/entrance/paas/videointercom/map/query
版本号	1.0.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
iotId	String	是	门禁对讲机iotId
roomId	String	否	房间ID 规则：期号-楼幢号-单元号-房号 例如： 1-9-3-1202 1-10-3A-1709 1-5-767-801 最大位数：1位-3位-4位-5位，规则数字+字母。 期号默认为1。
groupId	String	否	SIP组ID
pageNo	Integer	否	请求的页码，详细见“分页参数”定义。
pageSize	Integer	否	每页的记录数，详细见“分页参数”定义。

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data

名称	类型	描述
total	Integer	总数
pageNo	Integer	页码
pageSize	Integer	页码大小
data	JSONArray	映射关系列表

映射关系列表 data:

名称	类型	描述
roomId	String	房间ID
groupId	String	SIP组ID

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 如果需要登陆，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("iotId","value1");
request.putParam("roomId","value2");
request.putParam("groupId","value3");
request.putParam("pageNo","value4");
request.putParam("pageSize","value5");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/entrance/paas/videointercom/map/query", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data":{
        "total": 2,
        "pageNo": 1,
        "pageSize": 20,
     "data": [
            {
              "roomId": "room1",
              "groupId": "group1"
            },
            {
              "roomId": "room2",
              "groupId": "group2"
            }
            ]
    }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

2.5 呼叫事件

2.5.1 SIP呼叫事件通知订阅

呼叫事件topic

/broadcast/%s/vinotify

请在阿里云账号控制台中获取aliyun uid

https://account.console.aliyun.com/#/secure

数据格式

字段	类型	是否必传	备注
eventCode	String	是	事件编号，1：呼叫事件
eventTime	Long	是	事件发生时间戳(单位毫秒)
source	String	是	事件发起方，如呼叫事件，则为门禁机SIP号
targetType	String	是	目标类型，0 - SIP号；1 - SIP组
target	String	是	事件目标方，如呼叫事件，则为组ID

样例：

```
{
  "eventCode": "xxx",
  "eventTime": 1577085030000,
  "source": "22222",
  "targetType": 1,
  "target": "11111"
}
```

数据订阅请参考 第4节：业务数据订阅

3. 数据模型定义

模型ID：iot_entrance_videointercom_record_model

模型名称：可视对讲通行记录

模型版本：1.0

属性列表：

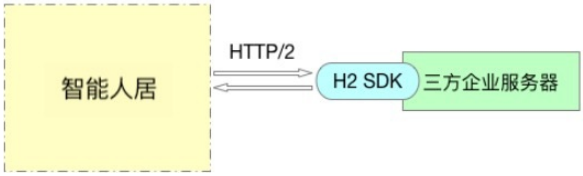
属性名称	属性标识符	数据类型	数据描述	必须
租户	tenantId	String	租户	是
项目	projectId	String	项目ID	否
事件时间	time	Date	事件时间	是
IoT设备标示	iotId	String	门禁对讲机IoT设备标示，如果为固话对讲则为空	否
productKey	productKey	String	productKey	否
deviceName	deviceName	String	deviceName	否
源类型	sourceType	String	0 - 云对讲 1 - 固话对讲	是

事件信息	eventInfo	String	类型为云对讲：
			{
			"sipNumber": "xxx",
			"groupId": "xxx",
			"roomId": "xxx"
			}
			类型为固话对讲：
			{
			"phone": "xxx"
			}
				是

4 业务数据订阅

除了设备数据可以由平台订阅以外，一些预定义好的业务数据也可以通过HTTP/2通道流转到您的服务器。

除了设备数据可以由平台订阅以外，一些预定义好的业务数据也可以通过HTTP/2通道流转到您的服务器。



配置HTTP/2服务的订阅可以参考下面第5章的设备数据订阅服务中的操作步骤。

通用定义：

topic

业务消息以模块维度发布，在topic上以**${subject}**体现数据分类，payload中是消息详情。

/broadcast/${uid}/homelink/${subject}

通用消息体

参数	类型	含义
id	String	消息ID
messageType	String	消息标题（类型）
gmtCreate	Long	数据消息产生时间，自1970-1-1起流逝的毫秒值
data	JSON	业务数据，见具体具体数据定义。

3.4.3.5. 可视对讲移动端SDK

3.4.3.5.1. 可视对讲移动端SDK接口说明-[iOS]

1. 概述

1.1 编写目的

本文是可视对讲设备云对讲和呼手机功能SDK接口说明。

1.2 名词解释

名词	解释
IoT	物联网

2. 介绍

通过集成对接SDK可以使门禁设备快速实现包括门禁机呼叫APP、APP调看门禁机、门禁机呼叫手机号等对讲功能。本文旨在描述该SDK的集成方式及主要业务代码。

限制条件：

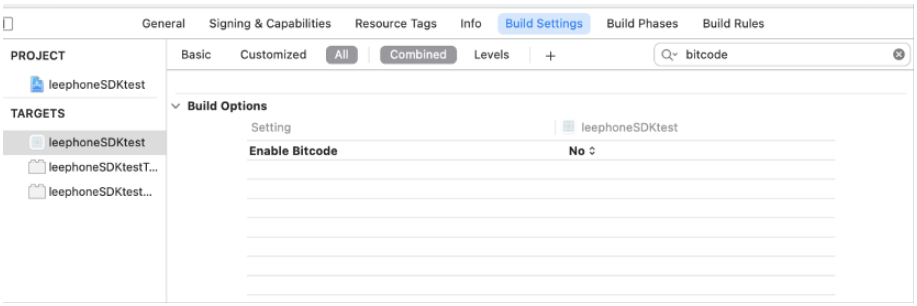
推荐使用Xcode版本：V12.5.1及以上

iOS版本：iOS9.0及以上

3. SDK集成方法

3.1 导入所需文件

1. 新建或打开已有工程
2. 导入EVVideoVoipSDK.framework和EVVideoVoipSDK.bundle
1. 将工程配置->TARGETS->Build Settings中的bitcode设置为NO



1. 在需要用到的类中导入头文件EVideoVoipSDK.h，调用各接口，使用SDK相关功能

```
#import <EVideoVoipSDK/EVideoVoipSDK.h>
```

3.2 权限要求

麦克风权限

相册权限（抓拍使用）

4. 类及接口列表

4.1 类说明

类名	说明
EVVoipManager	负责登录SIP
EVVoipAccount	SIP会话
EVVoipCall	负责SIP通话
EVVideoView	负责视频显示
EVVoipCallParams	通话参数

4.2 接口说明

4.2.1流程关键接口

名称	类名	方法名
初始化	EVVoipManager	- (void)setupWithCompletion:(void(^)())completion;
反初始化	EVVoipManager	- (void)teardown;
登录SIP服务器	EVVoipManager	- (EVVoipAccount *)loginWithSipNumber:(NSString*)sipnum displayName:(NSString*)displayName password:(NSString*)password domain:(NSString*)domain port:(NSInteger)port;

设置调看视频view	EVVoipManager	- (void)setVideoView:(UIView *)videoView;
安静模式，不响铃	EVVoipManager	- (void)enableSilentMode:(BOOL)silent;
呼叫APP	EVVoipManager	- (EVVoipCall *)callWithSipNumber:(NSString*)sipnum videoView:(UIView*)videoView error:(EVVoipError**)error;
来电监听	EVVoipManager	@property (nonatomic,copy) void (^onCallDidIncoming)(EVVoipCall *call);
账号状态监听	EVVoipAccount	@property (nonatomic,copy) void (^onStateDidChanged)(EVVoipAccountState state);
退出登录	EVVoipAccount	- (BOOL)logout;
移除账号	EVVoipAccount	- (void)remove;
消息通知	EVVoipManager	onMessageDidReceived
接听并挂断其它	EVVoipCall	- (EVVoipError *)acceptWithVideoView:(UIView*)videoView;
接听并暂停其它	EVVoipCall	- (EVVoipError*)acceptAndPauseOhterWithVideoView:(UIView*)videoView;
视频接听	EVVoipCall	- (EVVoipError *)acceptVideo;
接听并挂断其它通话	EVVoipCall	- (EVVoipError *)acceptWithVideoView:(UIView*)videoView;
接听并暂停其它通话	EVVoipCall	- (EVVoipError*)acceptAndPauseOhterWithVideoView:(UIView*)videoView;
设置媒体视图	EVVoipCall	- (void)setNativeVideoView:(UIView *)videoView; （和acceptWithVideoView的videoView有什么区别）
音频接听	EVVoipCall	- (EVVoipError *)acceptAudio;
暂停	EVVoipCall	- (void)pause;
恢复	EVVoipCall	- (void)resume;
挂断	EVVoipCall	- (EVVoipError*)hangup;
呼叫状态监听	EVVoipCall	@property (nonatomic, copy) void (^onStateDidChanged)(EVVoipCallState state, EVVoipCallEndReason reason);
视频状态改变	EVVoipCall	@property (nonatomic, copy) void (^onVideoStateDidChanged)(EVVoipVideoState state);

开锁结果回调	EVVoipCall	@property (nonatomic, copy) void (^onUnlockStateDidChanged)(EVVoipError *error, NSInteger unlockCode);
开锁	EVVoipCall	- (int)unlock;
截图	EVVoipCall	- (void)snapshot:(void(^)(UIImage* image, EVVoipError* error))completion;
抢麦	EVVoipCall	- (void)applyRemoteMicrophone:(void(^)(BOOL successed))completion; 抢麦，当有多端监视时，默认第一个监视可以说话，其他若要说话需要进行抢麦
释放抢麦	EVVoipCall	- (BOOL)cancelRemoteMicrophone;
是否静麦	EVVoipCall	- (BOOL)microphoneEnabled;
是否支持扬声器	EVVoipCall	- (BOOL)speakerEnabled;
是否关闭视频	EVVoipCall	- (BOOL)videoEnabled;
切换到扬声器	EVVoipCall	- (void)switchToSpeaker;
切换到听筒	EVVoipCall	- (void)switchToReceiver;

5. 主要业务流程

5.1 初始化

在进行对讲业务之前，需要先初始化该SDK，才能保证后续业务正常，后续的所有API调用都必须在SDK初始化成功的情况下执行。在应用的生命周期内，一般只进行一次初始化。

```
[[EVVoipManager defaultManager] setupWithCompletion:^{
 NSLog(@"初始化完成");
}];
```

当不再使用对讲业务时，可以调用deInit进行SDK反初始化。

```
// 反初始化对讲SDK
[[EVVoipManager defaultManager] teardown]
```

5.2 登入SIP服务器

在对讲SDK初始化成功之后，需要进行SIP服务器登入，登入成功后可以得到EVVoipAccount对象。

```
EVVoipAccount *account = [[EVVoipManager defaultManager]
         loginWithSipNumber:@"SIP账号"
            displayName:@"显示名"
            password:@"密码"
               domain:@"域名"
                 port:80//端口
               transport:@"传输方式，默认UDP"];
```

为了能够实时显示账号状态，本SDK提供 onStateDidChanged 接口实时监听当前账号状态。

```
account.onStateDidChanged = ^(EVVoipAccountState state){
    ...
    EVVoipAccountStateUnknown = 0,  /** 未知 */
    EVVoipAccountStateLogging,      /** 登入中 */
    EVVoipAccountStateLogouting,    /** 登出中 */
    EVVoipAccountStateOnline,       /** 在线 */
    EVVoipAccountStateOffline,      /** 离线 */
    EVVoipAccountStateLoggedout,    /** 已登出 */
    ...
};
```

当不再使用对讲业务，可以通过以下接口进行登出操作。

```
[account logout];
```

5.3 呼叫监听

为了使APP能够响应设备通话请求，需要注册一个来电回调监听接口。

```
[EVVoipManager defaultManager].onCallDidIncoming = ^(EVVoipCall * _Nonnull call) {
 // 新来电
 }
```

为了能够实时获取当前的通话状态，可以在EVVoipCall对象上设置一个通话状态回调接口。

```
call.onStateDidChanged = ^(EVVoipCallState state, EVVoipCallEndReason reason) {

}

/** 呼叫状态 */
typedef NS_ENUM(NSUInteger, EVVoipCallState) {
    EVVoipCallStateUnknown = 0,     /** 未知 */
    EVVoipCallStateIncoming,        /** 呼入中 */
    EVVoipCallStateOutgoing,        /** 呼出中 */
    EVVoipCallStateConnected,       /** 已接通 */
    EVVoipCallStatePaused,          /** 已挂起 */
    EVVoipCallStateEnd,             /** 结束 */
};

/** 电话结束原因 */
typedef NS_ENUM(NSUInteger, EVVoipCallEndReason) {
    EVVoipCallEndReasonNone          = 3000, /** 无 */
    EVVoipCallEndReasonHangupByUser = 3003, /** 挂断 */
    EVVoipCallEndReasonNotFound      = 3004, /** 未找到对方账号 */
    EVVoipCallEndReasonNotAnswered  = 3005, /** 未接听 */
    EVVoipCallEndReasonBusy          = 3006, /** 对方忙 */
};
```

5.4 通话接听

为了显示设备的对讲画面，需要提供一个UIView作为视频容器，如果不提供则为音频通话。

视频接听

```
[call acceptWithVideoView:self.playView];
```

音频接听

```
[call acceptWithVideoView:nil];
```

5.5 通话挂断

```
[call hangup];
```

5.6 开门

通话中可以发送开门指令，在发送开门指令前先设置监听器，通过监听获取开门指令执行结果。

```
[call unlock];
```

5.7 开关视频

打开视频

```
call.videoEnabled = YES;
[call setNativeVideoView:self.playView];
```

关闭视频

```
call.videoEnabled = NO;
[call setNativeVideoView:nil];
```

5.8 APP调看门禁机

设置调看UIView

```
[[EVVoipManager defaultManager] setVideoView:self.playView];
```

开始调看，可通过EVVoipError对象查看调看失败原因

```
self.currentCall =
[[EVVoipManager defaultManager] callWithSipNumber:self.sipNumber videoView:nil error:&error];
if (error) {
 [self showToastWithMsg:[NSString stringWithFormat:@"errorCode = %d",error.errorCode]];
    [self.navigationController popViewControllerAnimated:YES];
    return;
}
```

设置调看状态监听

```
self.currentCall.onStateDidChanged = ^(EVVoipCallState state, EVVoipCallEndReason reason) {

}
```

3.4.3.5.2. 可视对讲移动端SDK接口说明[Andriod]

1. 概述

1.1 编写目的

本文是可视对讲设备云对讲和呼手机功能SDK接口说明。

1.2 名词解释

名词	解释
IoT	物联网

2. 介绍

通过集成对接SDK可以使门禁设备快速实现包括门禁机呼叫APP、APP调看门禁机、门禁机呼叫手机号等对讲功能。本文旨在描述该SDK的集成方式及主要业务代码。

限制条件：

编译过程基于Android Studio 4.1.3

推荐使用GradlePlugin版本：4.1.2

推荐使用Gradle版本：6.5

推荐编译SDK版本：30

最小支持SDK版本：21

3. SDK集成方法

3.1 导入所需文件

新建或打开已有工程，拷贝leephone.aar包到app/libs目录下.

在app/build.gradle中添加aar包的引用，如图所示。

```
...
android {
  ...
    defaultConfig {
    ...
        ndk.abiFilters "armeabi-v7a", "arm64-v8a"
    }
}
...
repositories {
    flatDir {
        dirs 'libs' // aar dir
    }
}
...
dependencies {
 implementation(name:'leephone', ext:'aar')
  ...
}
```

3.2 在AndroidManifest.xml配置

支持的最低SDK版本为5.0:

<uses-sdk android:minSdkVersion="21" />

添加必要权限:

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

3.3 代码混淆配置

如果有做代码混淆，请确保以下类不加入混淆

```
-keep class com.evideo.voip.** { *; }
-keep class com.lee.phone.** { *; }
```

4. 类及接口列表

4.1 类说明

类名	说明
EVVoipManager	负责登录SIP
EVVoipAccount	SIP会话
EVVoipCall	负责SIP通话
EVVideoView	负责视频显示
EVVoipCallParams	通话参数

4.2 接口说明

4.2.1流程关键接口

名称	类名	方法名
初始化	EVVoipManager	init(Context context, EVVoipManager.OnInitCallback callback)
反初始化	EVVoipManager	deInit(Context context)
登录SIP服务器	EVVoipManager	EVVoipAccount login(String sipnum, String password, String displayName, String domain, int port)
账号状态监听	EVVoipAccount	setAccountStateCallback(EVVoipAccount.AccountStateCallbackaccountStateCallback)
来电监听	EVVoipManager	setIncomingCallback(EVVoipManager.IncomingCallbackincomingCallback)
呼叫APP	EdgeBoxManager	callRoomID(String roomID, EVVoipCallParams params, EdgeBoxManager.OnCallRoomCallback callback)
接听	EVVoipCall	accept(EVVideoView display)

挂断	EVVoipCall	hangup()
通话状态监听	EVVoipCall	setCallStateCallback(EVVoipCall.CallStateCallbackcallStateCallback)

5. 主要业务流程

5.1 初始化

在进行对讲业务之前，需要先初始化该SDK，才能保证后续业务正常，后续的所有API调用都必须在SDK初始化成功的情况下执行。在应用的生命周期内，一般只进行一次初始化。

```
public static void init(Context context, EVVoipManager.OnInitCallback callback)

// 初始化对讲SDK
EVVoipManager.init(getApplicationContext(), new EVVoipManager.OnInitCallback() {
    @Override
    public void complete() {
        // 对讲SDK初始化成功
    }
    @Override
    public void error(int code) {
        // 对讲SDK初始化失败，错误代码code
    }
});
```

当不再使用对讲业务时，可以调用deInit进行SDK反初始化。

```
// 反初始化对讲SDK
EVVoipManager.deInit(getApplicationContext());
```

5.2 登入SIP服务器

在对讲SDK初始化成功之后，需要进行SIP服务器登入，登入成功后可以得到EVVoipAccount对象。

```
EVVoipAccount EVVoipManager.login(String sipnum, String password, String displayName, String domain, int port)
```

为了能够实时显示账号状态，本SDK提供 AccountStateCallback 接口实时监听当前账号状态。

```
mEVVoipAccount.setAccountStateCallback(new EVVoipAccount.AccountStateCallback() {
    @Override
    public void onState(EVVoipAccount.AccountState state) {
        /* 获取该账号的状态
         * AccountState.ONLINE 在线
         * AccountState.OFFLINE 离线
         * AccountState.LOGINPROCESS 登入中
         * AccountState.NONE 无账号、未登入
         */
    }
});
```

当不再使用对讲业务，可以通过以下接口进行登出操作。

```
mEVVoipAccount.logoutEdgeBox();
```

5.3 呼叫监听

为了使APP能够响应设备通话请求，需要注册一个来电回调监听接口。

```
EVVoipManager.setIncomingCallback(new EVVoipManager.IncomingCallback() {
    @Override
    public void inComing(EVVoipCall evCall) {
        // evCall表示当前来电的通话
    }
});
```

为了能够实时获取当前的通话状态，可以在EVVoipCall对象上设置一个通话状态回调接口。

```
evCall.setCallStateCallback(new EVVoipCall.CallStateCallback() {
    @Override
    public void onState(CallState state, EndReason reason) {
        // 参考门禁机呼叫APP示例
    }
});
```

5.4 通话接听

为了显示设备的对讲画面，需要提供一个EVVideoView作为视频容器，如果不提供则为音频通话。

```
private EVVideoView eVideoView;
public int accept(EVVideoView display) throws EVVoipException;
// 初始化视频绘制容器，EVVideoView是一个android.app.Fragment，目前只支持在xml布局文件中配置
mEVVideoView= (EVVideoView) getFragmentManager().findFragmentById(R.id.display_view);
try {
    // 响应监控请求
    evCall.accept(mEVVideoView);
} catch (EVVoipException e) {
    e.printStackTrace();
}
```

5.5 通话挂断

```
evVoipCall.setCallStateCallback(null);
try {
 evVoipCall.hangup();
} catch (EVVoipException e) {
 e.printStackTrace();
}
```

5.6 开门

通话中可以发送开门指令，在发送开门指令前先设置监听器，通过监听获取开门指令执行结果。

EVVoipCall.unlock

```
evVoipCall.setUnlockCallback(new EVVoipCall.UnlockCallback() {
 @Override
 public void onSuccess() {
        //开门成功
 }

 @Override
 public void onFailure() {
  //开门失败
 }
});
int ret = call.unlock();//开门
```

5.7 通话中呼叫监听

```
EVVoipManager.setSwitchCallback
mOnSwitchCallback = new OnSwitchCallBack() {

 @Override
 public void onSwitch(EVVoipCall call) {

 }

 @Override
 public void onResume(EVVoipCall call) {

 }

 @Override
 public void onPause(EVVoipCall call) {

 }
};
EVVoipManager.setSwitchCallback(mOnSwitchCallback);
```

5.8 对话视频

```
evVoipCall.setOnRemoteVideoMuteListener(new OnRemoteVideoMuteListener() {
 @Override
 public void onRemoteVideoMuteAnswer(boolean mute) {
 }

 @Override
 public void onRemoteVideoMuteOffer(boolean mute) {

 }
});
```

5.9 开关视频

evVoipCall.enableVideo

5.10 通话协议监听

```
evVoipCall.setOnIceNatListener(new OnIceNatListener() {
 @Override
 public void OnAudioIceNatListener(int iceNet) {
  case IceNatType.ICE_NAT_TYPE_HOST:
  case IceNatType.ICE_NAT_TYPE_SRFLX:
  case IceNatType.ICE_NAT_TYPE_PRFLX:
  case IceNatType.ICE_NAT_TYPE_RELAY:
    }

    @Override
    public void OnVideoIceNatListener(int iceNet) {
  case IceNatType.ICE_NAT_TYPE_HOST:
  case IceNatType.ICE_NAT_TYPE_SRFLX:
  case IceNatType.ICE_NAT_TYPE_PRFLX:
  case IceNatType.ICE_NAT_TYPE_RELAY:
 }
});
evVoipCall.setOnMonitorListener(new OnMonitorListener() {
 @Override
 public void onMonitor() {

 }
});
```

问题1：setOnMonitorListener接口作用

5.11 APP调看门禁机

呼叫对讲设备

```
evVoipCall = VoipManager.getInstance().call(sipNum, evVoipCallParams);
```

设置通话状态监听

```
evVoipCall.setCallStateCallback(new CallStateCallback {
 @Override
 void onState(EVVoipCall.CallState var1, EVVoipCall.EndReason var2) {

    }
});
```

设置呼叫失败监听

```
evVoipCall.setCallFailureCallback(new CallFailureCallback() {
 @Override
 public void onFailure() {
    }
});
```

开启麦克风（本地还是远端）

```
evVoipCall.enableMicrophone(true); //true-关闭麦克风，false-开启麦克风
```

开启扬声器

```
evVoipCall.enableSpeaker(true);    //true-开启扬声器，false-关闭扬声器
```

设置远程视频禁止监听

```
evVoipCall.setOnRemoteVideoMuteListener(new OnRemoteVideoMuteListener() {
 @Override
 public void onRemoteVideoMuteAnswer(boolean mute) {

 }

 @Override
 public void onRemoteVideoMuteOffer(boolean mute) {

    }
});
```

6.调试方法

6.1日志抓取方法

6.2关键流程日志

6.3 启动功能选择Activity

```
adb shell am start com.lee.phone.demo/com.lee.phone.demo.ChooseActivity
```

6.4 强制关闭Demo

```
adb shell am force-stop com.lee.phone.demo
```

7.FAQ

7.1 Debug版本工作正常，Release工作启动异常

```
2021-10-12 15:13:51.270 30733-30733/? A/DEBUG: Abort message: 'JNI DETECTED ERROR IN APPLICATION: mid == null
        in call to CallStaticIntMethod
        from void com.lee.phone.jni.LeeJni.leeInit(java.lang.String, java.lang.String, int, int, boolean, java.lang.String, java.lang.St
ring, java.lang.String)'
2021-10-12 15:13:51.270 30733-30733/? A/DEBUG:     x0  0000000000000000  x1  00000000000077db  x2  0000000000000006  x3  0000007ffe2299b
0
2021-10-12 15:13:51.270 30733-30733/? A/DEBUG:     x4  fefeff71731e1f97  x5  fefeff71731e1f97  x6  fefeff71731e1f97  x7  7f7f7f7f7f7f7ff
f
2021-10-12 15:13:51.270 30733-30733/? A/DEBUG:     x8  00000000000000f0  x9  156478bff1817c09  x10 0000000000000001  x11 000000000000000
0
2021-10-12 15:13:51.270 30733-30733/? A/DEBUG:     x12 fffffff0fffffbdf  x13 ffffffffffffffff  x14 0000000000000004  x15 ffffffffffffff
f
2021-10-12 15:13:51.270 30733-30733/? A/DEBUG:     x16 00000072780c3958  x17 000000727809fa70  x18 00000072795bc000  x19 00000000000077d
b
2021-10-12 15:13:51.270 30733-30733/? A/DEBUG:     x20 00000000000077db  x21 00000000ffffffff  x22 00000071e1f50300  x23 00000071f256594
5
2021-10-12 15:13:51.270 30733-30733/? A/DEBUG:     x24 00000071f258754b  x25 0000000000000001  x26 00000071f2bbc258  x27 0000007278bca9a
0
2021-10-12 15:13:51.270 30733-30733/? A/DEBUG:     x28 00000071f2a91338  x29 0000007ffe229a50
2021-10-12 15:13:51.270 30733-30733/? A/DEBUG:     sp  0000007ffe229990  lr  000000727804f704  pc  000000727804f730
2021-10-12 15:13:51.650 30733-30733/? A/DEBUG: backtrace:
2021-10-12 15:13:51.650 30733-30733/? A/DEBUG:       #00 pc 0000000000083730  /apex/com.android.runtime/lib64/bionic/libc.so (abort+160)
(BuildId: 34c26ab262e93abd89e0bece8eedd768)
2021-10-12 15:13:51.650 30733-30733/? A/DEBUG:       #01 pc 00000000004b9f68  /apex/com.android.runtime/lib64/libart.so (art::Runtime::A
bort(char const*)+2280) (BuildId: 0e606603700990d6d4117e187ef17d14)
2021-10-12 15:13:51.650 30733-30733/? A/DEBUG:       #02 pc 000000000000b458  /system/lib64/libbase.so (android::base::LogMessage::~LogM
essage()+580) (BuildId: 34ca25523d4cef6dbfb222bd871ca0ed)
2021-10-12 15:13:51.650 30733-30733/? A/DEBUG:       #03 pc 000000000037885c  /apex/com.android.runtime/lib64/libart.so (art::JavaVMExt:
:JniAbort(char const*, char const*)+1584) (BuildId: 0e606603700990d6d4117e187ef17d14)
2021-10-12 15:13:51.650 30733-30733/? A/DEBUG:       #04 pc 00000000003b605c  /apex/com.android.runtime/lib64/libart.so (art::JNI::CallS
taticIntMethod(_JNIEnv*, _jclass*, _jmethodID*, ...)+1464) (BuildId: 0e606603700990d6d4117e187ef17d14)
2021-10-12 15:13:51.650 30733-30733/? A/DEBUG:       #05 pc 0000000000290da0  /data/app/com.mqsz.life-JDTtNTsP89Vr90VYRjaCTQ==/lib/arm64
/libleephone.so (lee_phone_detect_cameras+132) (BuildId: d1fde14fc5dc0807eb669a513e0e682a303e5ada)
2021-10-12 15:13:51.650 30733-30733/? A/DEBUG:       #06 pc 000000000045e644  /data/app/com.mqsz.life-JDTtNTsP89Vr90VYRjaCTQ==/lib/arm64
/libleephone.so (lee_android_camera_init+120) (BuildId: d1fde14fc5dc0807eb669a513e0e682a303e5ada)
```

检查Release模式下混淆配置是否正确，参考《3.3 代码混淆配置》

7.2 leeMuteSpeaker和enableSpeaker区别

leeMuteSpeaker是sdk中把收到的音频直接扔掉，不通过扬声器播放这一步

enableSpeaker是android系统标准接口，切换扬声器和听筒

3.4.3.6. 梯控联动服务API

本文档提供的接口用于人员通行时触发梯控设备的自动呼梯。

使用说明

- 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）

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。

人梯联动事件数据模型定义

数据模型名称：人梯联动数据模型（AutoElevatorCallRecord）

数据模型版本：1.0

名称	标识符	类型	说明
记录标示	eventId	String	记录的唯一标识
上报时间	eventTime	Date	上报时间
用户ID	identityID	String	用户IDENTITY_ID

通行方式	mediaType	String	枚举值：FACE、QRCODE、CARD
通行ID	mediaID	String	对应通行方式中人员具备的唯一ID，例如人脸ID、二维码，或者卡号
门禁产品名	entranceProductKey	String	无
门禁设备名	entranceDeviceName	String	无
梯控产品名	elevatorProductKey	String	无
梯控设备名	elevatorDeviceName	String	无
出发楼层信息	startFloor	String	无
出发电梯门信息	StartRoomNumber	String	无
目的楼层信息	targetList	String	无
联动结果返回码	callResultCode	int	200表示成功
联动结果信息	callResultMsg	String	联动结果， "success" 或者 "failed + 错误详情"
设备扩展信息	deviceExtInfo	String	设备扩展信息，保留字段
用户信息扩展信息	userExtInfo	String	用户扩展信息，保留字段
事件扩展信息	eventExtInfo	String	事件扩展信息，保留字段；

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datald

```
{
  "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datald

```
{
  "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

3.4.4. 安防接口

3.4.4.1. 视频监控服务API

1接口定义

1.1 请求视频直播

路径	/vision/stream/query
版本号	2.1.1
协议	HTTP, HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	是	设备ID
streamType	Int	否	码流类型
encrypted	Boolean	否	是否加密
encryptType	Int	否	加密类型，I帧加密为0
forcelFrame	Boolean	否	强制I帧
scheme	String	否	协议类型，rtmp/hls

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
path	String	播放地址

supportRelayPrePlay	BOOL	是否支持链路预热
decryptKey	JSON	解密密钥

解密密钥 decryptKey

名称	类型	描述
iv	String	向量base64编码结果
key	String	密钥base64编码结果

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("2.1.1");
// 设置参数
request.putParam("iotId","value1");
request.putParam("streamType","value2");
request.putParam("encrypted","value3");
request.putParam("encryptType","value4");
request.putParam("forceIFrame","value5");
request.putParam("scheme","value6");
request.putParam("cacheDuration","value7");
request.putParam("enablePrePlay","value8");
request.putParam("needDomainName","value9");
request.putParam("playUnlimited","value10");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/vision/stream/query", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": "success",
    "localizedMsg": "success",
    "data": {
        "path": "rtmp://47.100.XX.XX:8000/live?token=8da2&session=b52d&cacheDuration=1/5RnS_0",
        "decryptKey": {
            "iv": "aa",
            "key": "bb"
        }
    }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

1.2 停止直播推流

路径	/vision/stream/stop
版本号	1.0.4
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	是	设备ID
streamType	Int	否	码流类型

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.4");
// 如果需要登录，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("iotId","value1");
request.putParam("streamType","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/vision/stream/stop", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

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

1.3 按时间请求本地录像播放地址

路径	/vision/vod/localfile/getbytime
版本号	2.0.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	是	设备ID
beginTime	Int	是	录像开始时间，1970年1月1日开始的秒数
endTime	Int	是	录像结束时间，1970年1月1日开始的秒数

encrypted	Boolean	否	是否加密
encryptType	Int	否	加密类型
seekTime	Int	否	相对于beginTime的偏移量，单位秒
scheme	字符串	否	播放协议，输入内容包括：rtmp、flv。
recordType	整型	否	录像类型

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
vodUrl	String	播放地址
decryptKey	JSON	解密密钥

解密密钥 decryptKey:

名称	类型	描述
iv	String	向量base64编码结果
key	String	密钥base64编码结果

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("2.0.1");
// 设置参数
request.putParam("iotId","value1");
request.putParam("beginTime","value2");
request.putParam("endTime","value3");
request.putParam("encrypted","value4");
request.putParam("encryptType","value5");
request.putParam("seekTime","value6");
request.putParam("scheme","value7");
request.putParam("recordType","value8");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/vision/vod/localfile/getbytime", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
        "vodUrl":"rtmp://47.100.XX.XX:8000/vod?token=c0cf&session=b350/1602863385"
    }
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止"
}
```

1.4 按文件名请求本地录像播放地址

路径	/vision/vod/localfile/getbyname
版本号	2.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名

超时时间	3000

请求参数

名称	类型	是否必选	描述
iotId	String	是	设备ID
fileName	String	是	文件名
encrypted	Boolean	否	是否加密
encryptType	Int	否	加密类型
scheme	String	否	协议类型，支持 rtmp/hls
seekTime	整型	否	相对于beginTime的偏移量，单位秒

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
vodUrl	String	播放地址
decryptKey	JSON	解密密钥

解密密钥 decryptKey:

名称	类型	描述
iv	String	向量base64编码结果
key	String	密钥base64编码结果

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("2.0.0");
// 如果需要登录，设置当前的会话的token，token通过登录api获取
request.setIotToken("你的<IoTToken>");
// 设置参数
request.putParam("iotId","value1");
request.putParam("fileName","value2");
request.putParam("encrypted","value3");
request.putParam("encryptType","value4");
request.putParam("scheme","value5");
request.putParam("seekTime","value6");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/vision/vod/localfile/getbyname", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": "请补充具体返回业务参数！"
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
    "code": 403,
    "message": "request forbidden.",
    "localizedMsg": "请求被禁止",
    "data": null
}
```

3.4.4.2. 智能安防服务API

高容量网关云端接口API

1. 概述

2. API

2.1 人脸管理

2.1.1 新增人脸库

接入方式	服务模型
模型ID	smartSecurityService

模型版本号	1.0.4
接口名称	faceGroupCreate
接口版本号	1.0.0
请求方法	POST
接口描述	新增人脸库
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
productKey	是	String	产品名
deviceName	是	String	设备名
groupName	是	String	人脸库名称（最大字符长度63）
groupType	否	Integer	人脸库类型 1：黑名单 2：白名单 3：员工 4：访客
extInfo	否	String	扩展信息

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
groupId	String	人脸库ID

示例

请求参数示例：

```
{
  "productKey":"aa****cc",
  "deviceName":"test_name",
  "groupName":"test_name"
}
```

正常返回示例：

```
{
  "code":200,
  "message":"success",
  "data":{
    "groupId":"12"
  }
}
```

2.1.2 删除人脸库

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	faceGroupRemove
接口版本号	1.0.0
请求方法	POST
接口描述	删除人脸库
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
productKey	必填	String	产品名
deviceName	必填	String	设备名
groupId	必填	String	人脸库ID（最大字符长度63）

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求参数示例：

```
{
  "productKey":"aa****cc",
  "deviceName":"test_name",
  "groupId":"12"
}
```

正常返回示例：

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

2.1.3 修改人脸库

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	faceGroupModify
接口版本号	1.0.0
请求方法	POST
接口描述	修改人脸库
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
productKey	必填	String	产品名
deviceName	必填	String	设备名
groupId	必填	String	人脸库ID（最大字符长度63）
groupName	必填	String	人脸库名称（最大字符长度63）
groupType	可选	Integer	人脸库类型 1：黑名单 2：白名单 3：员工 4：访客
extInfo	可选	String	扩展信息

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求参数示例：

```
{
  "productKey":"aa****cc",
  "deviceName":"test_name",
  "groupId":"12",
  "groupName":"test_name2"
}
```

正常返回示例：

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

2.1.4 查询人脸库

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	faceGroupDetail
接口版本号	1.0.0
请求方法	POST
接口描述	查询人脸库
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
productKey	必填	String	产品名
deviceName	必填	String	设备名
groupId	必填	String	人脸库Id

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述

groupId	String	人脸库ID
groupName	String	人脸库名称
groupFaceNum	Integer	人脸库入库数量
extInfo	String	扩展信息

示例

请求参数示例：

```
{
  "productKey":"aa****cc",
  "deviceName":"test_name",
  "groupId":"12"
}
```

正常返回示例：

```
{
  "code":200,
  "message":"success",
  "data":{
    "groupId":"12",
    "groupName":"test_name",
    "groupFaceNum":100
  }
}
```

2.1.5 查询所有人脸库

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	getAllFaceGroup
接口版本号	1.0.0
请求方法	POST
接口描述	查询所有人脸库
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
productKey	必填	String	产品名
deviceName	必填	String	设备名

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。

message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
groups	JsonArry	

groups:

名称	类型	描述
groupId	String	人脸库ID（最大字符长度63）
groupName	String	人脸库名称（最大字符长度63）
groupFaceNum	Integer	人脸库入库数量
extInfo	String	扩展信息

示例

请求参数示例：

```
{
  "productKey":"aa****cc",
  "deviceName":"test_name"
}
```

正常返回示例：

```
{
  "code":200,
  "message":"success",
  "data":[{
    "groupId":"12",
    "groupName":"test_name",
    "groupFaceNum":10
  },{
    "groupId":"13",
    "groupName":"test_name2",
    "groupFaceNum":10
  }]
}
```

2.1.6 新增人脸

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	faceCreate
接口版本号	1.0.0
请求方法	POST

接口描述	创建人脸
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
productKey	必填	String	产品名
deviceName	必填	String	设备名
groupId	必填	String	人脸库ID（最大字符长度63）
facePicUrl	必填	String	人脸图片url
name	可选	String	姓名（最大字符长度63）
extInfo	可选	String	扩展信息

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
faceId	String	人脸ID（最大字符长度63）

示例

请求参数示例：

```
{
  "productKey":"aa****cc",
  "deviceName":"test_name",
  "groupId":"12",
  "name":"test_name",
  "facePicUrl":"****",
  "facePicType":"JPG"
}
```

正常返回示例：

```
{
  "code":200,
  "message":"success",
  "data":{
    "faceId":"12"
  }
}
```

2.1.7 删除人脸

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	faceRemove
接口版本号	1.0.0
请求方法	POST
接口描述	删除人脸
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
productKey	必填	String	产品名
deviceName	必填	String	设备名
groupId	必填	String	人脸库ID（最大字符长度63）
faceId	必填	String	人脸Id（最大字符长度31）

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求参数示例：

```
{
  "productKey":"aa****cc",
  "deviceName":"test_name",
  "groupId":"12",
  "faceId":"12"
}
```

正常返回示例：

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

2.1.8 修改人脸

接入方式	服务模型

模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	faceModify
接口版本号	1.0.0
请求方法	POST
接口描述	修改人脸
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
productKey	必填	String	产品名
deviceName	必填	String	设备名
groupId	必填	String	人脸库ID（最大字符长度63）
faceId	必填	String	人脸Id（最大字符长度31）
facePicUrl	可选	String	人脸图片url
name	可选	String	姓名（最大字符长度63）
extInfo	可选	String	扩展信息

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求参数示例：

```
{
  "productKey":"aa****cc",
  "deviceName":"test_name",
  "groupId":"12",
  "faceId",12,
  "name":"test_name2",
  "facePicUrl":"****"
}
```

正常返回示例：

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

2.1.9 搜索人脸

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	faceSearch
接口版本号	1.0.0
请求方法	POST
接口描述	查询人脸
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
productKey	必填	String	产品名
deviceName	必填	String	设备名
groupId	必填	String	人脸库ID
name	可选	String	姓名（最大字符长度63）
extInfo	可选	String	扩展信息
pageNo	可选	Integer	分页查询，页码，从1开始 默认为1
pageSize	可选	Integer	分页查询，页尺寸 默认为10

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
faces	JsonArray	人脸信息

人脸信息 faces:

名称	类型	描述
groupId	String	人脸库ID（最大字符长度63）
groupName	String	人脸库名称（最大字符长度127）
faceId	String	人脸Id（最大字符长度31）
name	String	姓名（最大字符长度63）
facePicBase64	String	人脸图片base64编码后的数据
facePicType	String	图片类型 JPG PNG BMP
extInfo	String	扩展信息

示例

请求参数示例：

```
{
  "productKey":"aa****cc",
  "deviceName":"test_name",
  "groupId":"12",
  "name":"test_name2",
  "pageNo":1,
  "getCount":10
}
```

正常返回示例：

```
{
  "code":200,
  "message":"success",
  "data":{
    "faces":[{
      "groupId":"12",
      "groupName":"test_grp_name",
      "faceId":"12",
      "name":"test_name",
      "facePicBase64":"AABB****1122",
      "facePicType":"JPG"
    },{
      "groupId":"12",
      "groupName":"test_grp_name",
      "faceId":"13",
      "name":"test_name2",
      "facePicBase64":"AABB****1122",
      "facePicType":"JPG"
    }]
  }
}
```

2.2 事件布防

2.2.1 设置布防规则

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	setAlarmRule
接口版本号	1.0.0
请求方法	POST
接口描述	设置布防规则
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
productKey	必填	String	产品名
deviceName	必填	String	设备名
eventType	必填	Integer	事件类型 1：人员识别 2：区域入侵 3：电动车进电梯 4：垃圾桶满溢 5：高空抛物 6：消防通道占用 7：热成像-温度异常 8：热成像-人体测温 100：其他
alarmRule	必填	JsonArray	布防计划

布防计划 alarmRule:

名称	是否必选	类型	描述
weekDay	必填	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
timeSlot	必填	JSONArray	时间段
bindGroup	必填	JSONArray	人脸底库绑定 eventType== 1 时有效

extInfo	可选	String	规则扩展信息

时间段 timeSlot：

名称	是否必选	类型	描述
time	必填	String	时间范围 0:00-2:00
interval	必填	Integer	时间间隔，单位秒
timeExtInfo	可选	String	时间段扩展信息

人脸底库绑定 bindGroup：

名称	是否必选	类型	描述
groupId	必填	String	eventType == 1 时有效

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求参数示例：

```
{
  "productKey":"aa****cc",
  "deviceName":"test_name",
  "eventType":1,
  "alarmRule":[{
    "weekDay":"SUNDAY",
    "timeSlot":[{
      "time":"00:00-2:00",
      "interval":600
    },{
      "time":"02:00-4:00",
      "interval":600
    }]
  },{
    "weekDay":"MONDAY",
    "timeSlot":[{
      "time":"00:00-23:59",
      "interval":600
    }]
  }]
}
```

正常返回示例：

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

2.2.2 获取布防规则

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	getAlarmRule
接口版本号	1.0.0
请求方法	POST
接口描述	获取布防规则
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
productKey	必填	String	产品名
deviceName	必填	String	设备名
eventType	必填	Integer	事件类型 1：人员识别 2：区域入侵 3：电动车进电梯 4：垃圾桶满溢 5：高空抛物 6：消防通道占用 7：热成像-温度异常 8：热成像-人体测温 100：其他

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述

weekDay	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
timeSlot	JsonArray	时间段
bindGroup	JSONArray	人脸底库绑定 deviceType == 1 时有效
extInfo	String	规则扩展信息

时间段 timeSlot：

名称	是否必选	类型	描述
time	必填	String	时间范围 0:00-2:00
interval	必填	Integer	时间间隔，单位秒
timeExtInfo	可选	String	时间段扩展信息

人脸底库绑定 bindGroup：

名称	是否必选	类型	描述
groupId	必填	String	eventType == 1 时有效

示例

请求参数示例：

```
{
  "productKey":"aa****cc",
  "deviceName":"test_name",
  "eventType":1
}
```

正常返回示例：

```
{
  "code":200,
  "message":"success",
  "data":[{
    "weekDay":"SUNDAY",
    "timeSlot":[{
      "time":"00:00-2:00",
      "interval":600
    },{
      "time":"02:00-4:00",
      "interval":600
    }]
  },{
    "weekDay":"MONDAY",
    "timeSlot":[{
      "time":"00:00-23:59",
      "interval":600
    }]
  }]
}
```

2.2.3 告警事件上报

接入方式	数据模型
模型ID	securityAlarmEvent
模型版本号	1.2
模型描述	边缘安防服务告警事件数据模型

属性列表

参数	是否必选	类型	描述
productKey	必填	String	产品名称
deviceName	必填	String	设备名称
eventType	必填	String	事件类型 1：人员识别 2：区域入侵 3：电动车进电梯 4：垃圾桶满溢 5：高空抛物 6：消防通道占用 7：热成像-温度异常 8：热成像-人体测温 0xFF：其他
eventTime	必填	String	事件的触发时间 格式：2010-01-01 01:01:11
eventId	必填	String	事件ID，事件的唯一标识
eventStatus	可选	String	事件状态 0：瞬时 1：开始 2：停止

confidence	可选	Integer	事件置信度，0~255，值越大置信度越高
picSceneUrl	可选	String	全景图片url
objectInfo	可选	String(JsonArray)	目标INFO
extInfo	可选	String(Json)	其他扩展信息

目标INFO：objectInfo

参数	是否必选	类型	描述
objPicUrl	可选	String	目标图片url
letfTopX	可选	String	左上角x坐标(0, 8192)
leftTopY	可选	Integer	左上角y坐标(0, 8192)
rightBottomX	可选	String	右下角x坐标(0, 8192)
rightBottomY	可选	String	右下角y坐标(0, 8192)
objectExtInfo	可选	String(Json)	目标自定义扩展信息

ExtInfo：当eventType = 1 人员识别时：

参数	是否必选	类型	描述
similarity	必填	Float	相似度 (0, 1)
groupId	必填	String	人脸所在底库的GroupId
faceId	必填	String	底库人脸Id
gender	可选	Integer	性别 1：男 2：女 0xFF：未知
glass	可选	Integer	是否戴眼镜 0：未戴眼镜 1：戴眼镜 0xFF：未知
mask	可选	Integer	是否戴口罩 0：未戴口罩 1：戴口罩 0xFF：未知
pitch	可选	Integer	俯仰角 [-90,90]
yow	可选	Integer	偏航角 [-90,90]
roll	可选	Integer	翻滚角 [-90,90]

faceQuality	可选	Integer	人脸质量 [0-100]
facePicUrl	可选	String	底库人脸图片url

ExtInfo：当eventType = 2 区域入侵时：

参数	是否必选	类型	描述
targetType	必填	Integer	目标类型 1：人 2：车 0xFF：其他

ExtInfo：当eventType = 6 消防通道占用时：

参数	是否必选	类型	描述
plateColor	必填	Integer	车牌颜色 0：未知颜色 1：蓝色 2：黄色 3：白色 4：黑色 5：绿色 6：黄绿色 0xFF：其他颜色
plateNum	可选	String	车牌号
vehicleColor	必填	Integer	车身颜色 1：红色 2：绿色 3：蓝色4：黄色 5：白色 6：灰色 7：黑色 8：紫色 9：棕色 10：粉色 11：银色 12：深蓝 13：深灰 14：青色 0xFF：其他颜色

ExtInfo：当eventType = 7 热成像-温度异常时：

参数	是否必选	类型	描述
alarmContion	必填	Integer	报警条件 1：大于 2：等于 3：小于
tempValue	必填	Float	温度值

tempUnit	必填	Integer	温度单位 1：摄氏度 2：华氏度

ExtInfo：当eventType = 8 热成像-人体测温时：

参数	是否必选	类型	描述
tempMax	必填	Float	最高温度
tempUnit	必填	Integer	温度单位 1：摄氏度 2：华氏度

2.3 抓图计划

2.3.1 设置抓图计划

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	setPicCaptureTask
接口版本号	1.0.3
请求方法	POST
接口描述	设置抓图计划
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
taskId	必填	String	任务的唯一标识
deviceList	必填	JSONArray	关联的监控设备
startDate	必填	String	计划开始时间 格式：YYYY-MM-DD
endDate	必填	String	计划结束时间 格式：YYYY-MM-DD
captureWeekDay	可选	JsonArray	抓图周期 captureWeekDay和cron二选一必填
cron	可选	JsonArray	cron表达式 captureWeekDay和cron二选一必填

status	必填	Integer	启用状态 1：启用 0：禁用
extInfo	可选	String	扩展字段

设备列表：deviceList：

名称	是否必选	类型	描述
productKey	必填	String	产品名称
deviceName	必填	String	设备名称

抓图周期：captureWeekDay

名称	是否必选	类型	描述
weekDay	必填	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
captureTime	必填	JsonArray	抓图时分 24时制，格式： [hh:mm,hh2:mm2]

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求参数示例：

```
{
  "taskId":"123",
  "deviceList":[{
    "productKey":"aa****cc",
    "deviceName":"test_name"
  },{
    "productKey":"aa****cc",
    "deviceName":"test_name2"
  }],
  "startDate":"2010-01-01",
  "endDate":"2010-04-01",
  "captureWeekDay":[{
    "weekDay":"SUNDAY",
    "captureTime":["10:00","12:00"]
  },{
    "weekDay":"MONDAY",0
    "captureTime":["10:00","12:00"]
  }],
  "status":1
}
```

正常返回示例：

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

2.3.2 获取抓图计划

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	getPicCaptureTask
接口版本号	1.0.1
请求方法	POST
接口描述	获取抓图计划
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
taskId	必填	String	任务的唯一标识

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

data	JSON	响应结果

响应结果 data:

名称	类型	描述	
deviceList	必填	JSONArray	关联的监控设备
startDate	必填	String	计划开始时间 格式：YYYY-MM-DD
endDate	必填	String	计划结束时间 格式：YYYY-MM-DD
captureWeekDay	可选	JsonArray	抓图周期 captureWeekDay和cron二选一必填
cron	可选	List	cron表达式 captureWeekDay和cron二选一必填
status	必填	Integer	启用状态 1：启用 0：禁用
extInfo	可选	String	扩展字段

设备列表：deviceList：

名称	是否必选	类型	描述
productKey	必填	String	产品名称
deviceName	必填	String	设备名称

抓图周期：captureWeekDay

名称	是否必选	类型	描述
weekDay	必填	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
captureTime	必填	JsonArray	抓图时分 24时制，格式： [hh:mm,hh2:mm2]

示例

请求参数示例：

```
{
  "taskId":"123"
}
```

正常返回示例：

```
{
  "code":200,
  "message":"success",
  "data":{
    "deviceList":[{
      "productKey":"aa****cc",
      "deviceName":"test_name"
    },{
      "productKey":"aa****cc",
      "deviceName":"test_name2"
    }],
    "startDate":"2010-01-01",
    "endDate":"2010-04-01",
    "captureWeekDay":[{
      "weekDay":"SUNDAY",
      "captureTime":"10:00,12:00"
    },{
      "weekDay":"MONDAY",
      "captureTime":"10:00,12:00"
    }],
    "status":1
  }
}
```

2.3.3 删除抓图计划

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	removeCaptureTask
接口版本号	1.0.1
请求方法	POST
接口描述	删除抓图计划
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
taskId	必填	String	任务的唯一标识

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求参数示例：

```
{
  "taskId":"123"
}
```

正常返回示例：

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

2.3.4 抓图记录上报

接入方式	数据模型
模型ID	securityCapture
模型版本号	1.0
模型描述	智能安防抓图记录

属性列表

参数	是否必选	类型	描述
productKey	必填	String	产品名称
deviceName	必填	String	设备名称
taskId	可选	String	任务ID
captureTime	必填	Long	抓图时间，UTC时间，精确到s
picUrl	必填	String	抓拍图片的url

2.4 抓拍记录查询

2.4.1 抓拍记录查询

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	getPicRecord
接口版本号	1.0.1
请求方法	POST
接口描述	抓拍记录查询
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
picBase64	必填	String	图片Base64
picType	必填	String	图片类型 JPG PNG BMP
recordLimit	必填	Integer	获取记录数量限制
startTime	必填	Long	UTC时间，精确到s
endTime	必填	Long	UTC时间，精确到s endTime与startTime相差<=24h
minSimilarity	必填	Float	最小相似度(0,1)
maxSimilarity	必填	Float	最大相似度(0,1) maxSimilarity > minSimilarity
pageNo	可选	Integer	分页查询，页码，从1开始 默认为1
pageSize	可选	Integer	分页查询，页尺寸，pageSize>1&pageSize<=20 默认为10

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
recordList	JsonArray	记录列表

记录列表：recordList:

名称	类型	描述
productKey	String	产品名称
deviceName	String	设备名称

smallPicBase64	String	小图base64
smallPicType	String	图片类型 JPG PNG BMP
scenePicId	String	场景图id
similarity	Float	相似度(0,1)
captureTime	Long	UTC时间，精确到s
extinfo	JsonArray	眼镜，性别，年龄等

示例

请求参数示例：

```
{
  "picBase64":"AABB****1122",
  "picType":"JPG",
  "recordLimit":10,
  "startTime":1631794685,
  "endTime":1631798285,
  "minSimilarity":0.55,
  "maxSimilarity":0.65
}
```

正常返回示例：

```
{
  "code":200,
  "message":"success",
  "data":{
    "recordList":[{
      "productKey":"aa****cc",
      "deviceName":"test_name",
      "smallPicBase64":"AABB****1122",
      "smallPicType":"JPG",
      "scenePicId":"pic1234",
      "similarity":0.6,
      "captureTime":1631795685
    },{
      "productKey":"aa****cc",
      "deviceName":"test_name",
      "smallPicBase64":"AABB****1122",
      "smallPicType":"JPG",
      "scenePicId":"pic1235",
      "similarity":0.6,
      "captureTime":1631795685
    }]
  }
}
```

2.4.2 抓拍记录场景图获取

接入方式	服务模型
模型ID	smartSecurityService
模型版本号	1.0.4
接口名称	scenePicRecordUpload

接口版本号	1.0.1
请求方法	POST
接口描述	抓拍记录场景图获取
Content-Type	application/json

请求参数

名称	是否必选	类型	描述
scenePicId	必填	String	场景图ID
uploadUrl	必填	String	upload地址

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求参数示例：

```
{
  "scenePicId":"pic1234",
  "uploadUrl":"https://xxxxx"
}
```

正常返回示例：

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

3. 附录

3.1 错误码

code	message	说明
200	success	成功
804	param check error: /'%s/'	参数校验失败，返回失败的参数

3.4.4.3. 室内安防数据定义

1 室内安防品类信息

室内安防的品类归属为：**智慧社区**

查看方式：可登录物联网应用服务平台，点击**边缘接入-品类管理**，选择**智慧社区-室内安防**，进入查看，如下图所示。



2 室内安防设备物模型

2.1 震动探测器

模型编码：vibrator

报警事件：

功能类型	名称	标识符	事件类型	说明
事件	报警	Alarm	告警	设备上报的报警事件

布防状态属性：

功能类型	名称	标识符	数据类型	说明
属性	布防状态	AlertState	枚举型： 0：未知 1：撤防 2：布防	设置布防或撤防状态

2.2 水浸探测器

模型编码：water

报警事件：

功能类型	名称	标识符	事件类型	说明
事件	报警	Alarm	告警	设备上报的报警事件

布防状态属性：

功能类型	名称	标识符	数据类型	说明
属性	布防状态	AlertState	枚举型： 0：未知 1：撤防 2：布防	设置布防或撤防状态

2.3 烟雾探测器

模型编码：smoke

报警事件：

功能类型	名称	标识符	事件类型	说明
事件	报警	Alarm	告警	设备上报的报警事件

布防状态属性：

功能类型	名称	标识符	数据类型	说明
属性	布防状态	AlertState	枚举型： 0：未知 1：撤防 2：布防	设置布防或撤防状态

2.4 瓦斯探测器

模型编码：gas

报警事件：

功能类型	名称	标识符	事件类型	说明
事件	报警	Alarm	告警	设备上报的报警事件

布防状态属性：

功能类型	名称	标识符	数据类型	说明
属性	布防状态	AlertState	枚举型： 0：未知 1：撤防 2：布防	设置布防或撤防状态

2.5 门窗磁探测器

模型编码：magnetic

报警事件：

功能类型	名称	标识符	事件类型	说明
事件	报警	Alarm	告警	设备上报的报警事件

布防状态属性：

功能类型	名称	标识符	数据类型	说明
属性	布防状态	AlertState	枚举型： 0：未知 1：撤防 2：布防	设置布防或撤防状态

2.6 红外探测器

模型编码：infrared

报警事件：

功能类型	名称	标识符	事件类型	说明
事件	报警	Alarm	告警	设备上报的报警事件

布防状态属性：

功能类型	名称	标识符	数据类型	说明

属性	布防状态	AlertState	枚举型： 0：未知 1：撤防 2：布防	设置布防或撤防状态

2.7 SOS紧急按钮

模型编码：SOS_button

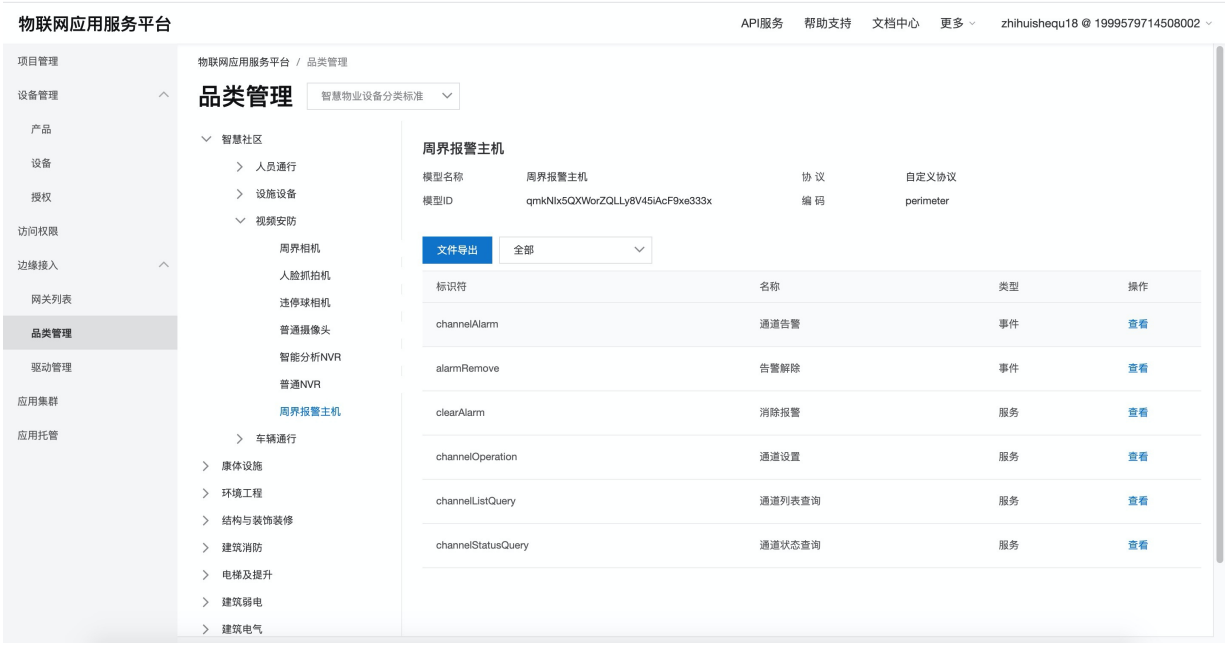
功能类型	名称	标识符	事件类型	说明
事件	报警	Alarm	告警	设备上报的报警事件

3.4.4.4. 周界报警主机数据定义

1 周界报警主机品类信息

人脸门禁的品类归属为：**智慧社区-视频安防-周界报警主机**

查看方式：可登录物联网应用服务平台，点击**边缘接入-品类管理**进入查看，如下图所示。



2 周界报警主机物模型

2.1 事件

2.1.1 通道告警事件

功能类型	名称	标识符	事件类型
事件	通道告警	channelAlarm	告警

输出参数：

名称	标识符	数据类型	说明
通道ID	channelID	text (字符串)	数据长度：2048

告警类型	alarmType	enum (枚举型)	0 其他告警 1 入侵告警 2 触网告警 3 短路告警 4 断路告警 5 防拆告警 6 故障告警

2.1.2 告警解除事件

功能类型	名称	标识符	事件类型
事件	告警解除	alarmRemove	信息

输入参数：无

输出参数：

名称	标识符	数据类型	说明
通道ID	channelID	text (字符串)	数据长度：2048
告警类型	alarmType	enum (枚举型)	0 其他告警 1 入侵告警 2 触网告警 3 短路告警 4 断路告警 5 防拆告警 6 故障告警

2.2 服务

2.2.1 消除报警服务

功能类型	名称	标识符	调用方式
服务	消除报警	clearAlarm	同步

输入参数：

名称	标识符	数据类型	数据长度
通道ID	channelID	text (字符串)	2048

输出参数：无

2.2.2 通道设置服务

功能类型	名称	标识符	调用方式
服务	通道设置	channelOperation	同步

输入参数：

名称	标识符	数据类型	说明
通道ID	channelID	text (字符串)	数据长度：2048

操作	operation	enum (枚举型)	0 未知 1 布防 2 撤防

输出参数：无

2.2.3 通道列表查询服务

功能类型	名称	标识符	调用方式
服务	通道列表查询	channelListQuery	同步

输入参数：无

输出参数：

名称	标识符	数据类型	元素个数
通道列表	channelList	array (数组)	512

2.2.4 通道状态查询服务

功能类型	名称	标识符	调用方式
服务	通道状态查询	channelStatusQuery	同步

输入参数：

名称	标识符	数据类型	说明
通道ID	channelID	text (字符串)	数据长度：2048

输出参数：

名称	标识符	数据类型	枚举项
状态	status	enum (枚举型)	0 未知 1 布防 2 撤防

3.4.5. EBA接口

3.4.5.1. 设备告警开放服务API

1. 概述

告警开放服务是应用服务平台告警功能的API版本，除继承原有告警功能外，还扩展支持告警边缘端部署运行，不同类型设备联合告警，时间窗口延迟触发告警等功能。

2. 接口定义

2.1 新增或更新告警规则

路径	/alarm/rule/addorupdate
版本号	1.0.0
协议	HTTPS
请求方法	POST

授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
alarmName	String	必填	告警规则名称
alarmLevel	Integer	必填	告警等级，0严重，1高，2中，3低
auxiliaryLevel	Integer	非必填	辅助等级，0~999，客户自定义
alarmContent	String	必填	告警内容
productKey	String	必填	设备PK
sceneConfig	String	必填	阈值告警规则配置（限值告警高限值配置） 详细参考附录
extConfig	字符串	非必填	扩展内容，用于回填
receivers	JSONArray	非必填	告警消息接收者
alarmType	Integer	必填	告警类型，1运行告警，2故障告警
alarmMode	Integer	必填	告警模式，1阈值告警，2限值告警
timeWindow	Integer	非必填	时间窗口，告警满足条件持续一段时间后触发，单位：秒
runEnv	String	必填	运行环境，cloud云端，edge边缘端
multiDevice	Integer	是	是否多设备
alarmRuleId	String	非必填	告警规则id，更新时必填
lowSceneConfig	String	非必填	限值告警的低限值规则配置，限制告警必填 详细参考附录

告警消息接收者receivers：

名称	类型	是否必选	描述
type	String	必填	接收类型，EMAIL邮箱，receivers不为空时必填
email	String	必填	邮箱地址，receivers不为空时必填

返回数据

名称	类型	描述

code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	String	新增或更新的告警记录id

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("alarmName","value1");
request.putParam("alarmLevel","value2");
request.putParam("auxiliaryLevel","value3");
request.putParam("alarmContent","value4");
request.putParam("productKey","value5");
request.putParam("sceneConfig","value6");
request.putParam("extConfig","value7");
request.putParam("receivers","value8");
request.putParam("alarmType","value9");
request.putParam("alarmMode","value10");
request.putParam("timeWindow","value11");
request.putParam("runEnv","value12");
request.putParam("multiDevice","value13");
request.putParam("alarmRuleId","value14");
request.putParam("lowSceneConfig","value15");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/alarm/rule/addorupdate", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

请求入参示例：

```
{
   "alarmName": "新增告警规则",
   "alarmLevel": 0,
   "auxiliaryLevel": 10,
   "alarmContent": "设备产生告警",
   "productKey": "algEQeIkPn5",
   "sceneConfig": "{/* 见附录1 */}",
   "receivers": [
       {
            "type": "EMAIL",
            "email": "test@163.com"
       }
   ],
   "alarmType": 1,
   "alarmMode": 1,
   "timeWindow": 300,
   "runEnv": "cloud",
   "multiDevice": 0
}
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": ""
}
```

异常返回示例：

```
{
    "id": "37f7e5fa-d6a5-4efe-8abf-5bf23dca6284",
     "code": 28403,
     "message": "The project not exist",
     "localizedMsg": "该项目不存在",
     "data": null
}
```

2.2 批量删除告警规则

路径	/alarm/rule/delete
版本号	1.0.1
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
alarmRuleIds	JSONArray<String>	必填	告警规则id的集合
clusterId	String	非必填	集群id，告警运行在边缘端时必填

返回数据

名称	类型	描述

code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON array<String>	删除失败的规则ID列表

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.1");
// 设置参数
request.putParam("alarmRuleIds","value1");
request.putParam("clusterId","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/alarm/rule/delete", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

请求入参示例：

```
{
   ["8e2e8d41f4604878b413fc588333f068", "df152383d40842b0814346cc88cfd1b4"],
   "pVxFeNmZ8axogEnWCD9E"
}
```

正常返回示例：

```
{
   "code": 200,
   "message": "success",
   "data": ["df152383d40842b0814346cc88cfd1b4"]
}
```

异常返回示例：

```
{
   "code": 28420,
   "message": "No access to alarm rule",
   "localizedMsg": "无权访问该告警规则"
}
```

2.3 分页查询告警规则列表

路径	/alarm/rule/query

版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
nameLike	String	非必填	告警规则名称，模糊匹配
alarmLevel	Integer	非必填	告警等级，0严重，1高，2中，3低
status	Integer	非必填	启用状态，1启用，0停止
alarmType	Integer	非必填	告警类型，1运行告警，2故障告警
alarmMode	Integer	非必填	告警模式，1阈值告警，2限值告警
runEnv	String	非必填	运行环境，cloud云端，edge边缘端
pageNo	Integer	必填	当前页码，默认1
pageSize	Integer	必填	每页条数，默认20

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON array	响应结果

响应结果 data:

名称	类型	描述
total	Integer	总条数
pageNo	Integer	当前页码
pageSize	Integer	每页条数
data	Integer	告警规则列表

告警规则列表 data:

名称	类型	描述
alarmRuleId	String	告警规则id
alarmName	String	告警规则名称
status	Integer	启用状态，1启用，0停止
alarmLevel	Integer	告警等级，0严重，1高，2中，3低
auxiliaryLevel	Integer	辅助等级，0~999，用户自定义
content	String	告警内容
productKey	String	设备PK
alarmType	Integer	告警类型，1运行告警，2故障告警
runEnv	String	运行环境，cloud云端，edge边缘端
multiDevice	Integer	是否多设备，0单设备，1多设备
alarmMode	Integer	告警模式，1阈值告警，2限值告警
timeWindow	Integer	时间窗口，告警满足条件持续一段时间后触发，单位：秒

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("nameLike","value1");
request.putParam("alarmLevel","value2");
request.putParam("status","value3");
request.putParam("alarmType","value4");
request.putParam("alarmMode","value5");
request.putParam("runEnv","value6");
request.putParam("pageNo","value7");
request.putParam("pageSize","value8");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/alarm/rule/query", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

请求入参示例：

```
{
   "nameLike": "告警",
   "alarmLevel": 0,
   "alarmType": 1,
   "status": 1,
   "alarmMode": "2",
   "runEnv": "edge",
   "pageNo": 1,
   "pageSize": 10
}
```

正常返回示例：

```
{
   "code": 200,
   "message": "success",
   "data": {
       "total": 100,
       "pageNo": 1,
       "pageSize": 10,
       "data": [
            {
                "alarmRuleId": "df152383d40842b0814346cc88cfd1b4",
                "alarmName": "设备告警",
                "status": 1,
                "alarmLevel": 0,
                "auxiliaryLevel": 5,
                "content": "设备电压超标",
                "productKey": "a1gEQeIkPn5",
                "alarmType": 1,
                "runEnv": "edge",
                "multiDevice": 0,
                "alarmMode": 2,
                "timeWindow": 600
            }
       ]
   }
}
```


异常返回示例：


```
{
   "code": 28403,
   "message": "The project not exist",
   "localizedMsg": "该项目不存在"
}
```


2.4 查询单个告警规则的详情

路径	/alarm/rule/detail/get
版本号	1.0.0
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
alarmRuleId	String	必填	告警规则id

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

data	JSON	响应结果

响应结果 data:

名称	类型	描述
alarmRuleId	String	告警规则id
alarmName	String	告警规则名称
status	Integer	启用状态，1启用，0停止
sceneConfig		阈值告警规则配置（限值告警高限值配置）
extConfig	String	扩展配置
alarmLevel	Integer	告警等级，0严重，1高，2中，3低
auxiliaryLevel	Integer	辅助等级，0~999，用户自定义
content	String	告警内容
productKey	String	设备PK
receivers	List	告警消息接收者
alarmType	Integer	告警类型，1运行告警，2故障告警
lowSceneConfig	String	限值告警的低限值规则配置
runEnv	String	运行环境，cloud云端，edge边缘端
multiDevice	Integer	是否多设备，0单设备，1多设备
alarmMode	Integer	告警模式，1阈值告警，2限值告警
timeWindow	Integer	时间窗口，告警满足条件持续一段时间后触发，单位：秒

告警消息接收者receivers

名称	类型	描述
type	String	接收类型，EMAIL邮箱
email	String	邮箱地址

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("alarmRuleId","value1");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/alarm/rule/detail/get", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

请求入参示例：

```
"alarmRuleId": "df152383d40842b0814346cc88cfd1b4"
```

正常返回示例：

```
{
   "code": 200,
   "message": "success",
   "data": {
       "alarmRuleId": "df152383d40842b0814346cc88cfd1b4",
       "alarmName": "设备告警",
       "status": 1,
       "alarmLevel": 0,
       "auxiliaryLevel": 5,
       "content": "设备电压超标",
       "productKey": "a1gEQeIkPn5",
       "sceneConfig": "{}",
       "receivers": [
            {
                "type": "EMAIL",
                "email": "test@163.com"
            }
       ],
       "alarmType": 1,
       "lowSceneConfig": "{}",
       "runEnv": "edge",
       "multiDevice": 0,
       "alarmMode": 2,
       "timeWindow": 600
   }
}
```

异常返回示例：

```
{
   "code": 28403,
   "message": "The project not exist",
   "localizedMsg": "该项目不存在"
}
```

2.5 批量操作告警规则启停

路径	/alarm/rule/status/update
版本号	1.0.2
协议	HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
alarmRuleIds	JSONArray<String>	必填	告警规则id
operType	Integer	必填	操作类型，1启动，0停止
clusterId	String	非必填	集群id，告警运行在边缘端时必填

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	JSON	响应结果

响应结果 data:

名称	类型	描述
deployId	字符串	部署单id
failAlarmIds	JSONArray<String>	失败告警id列表

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.2");
// 设置参数
request.putParam("alarmRuleIds","value1");
request.putParam("operType","value2");
request.putParam("clusterId","value3");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/alarm/rule/status/update", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

请求入参示例：

```
{
   ["8e2e8d41f4604878b413fc588333f068", "df152383d40842b0814346cc88cfd1b4"],
   "operType": 0,
   "pVxFeNmZ8axogEnWCD9E"
}
```

正常返回示例：

```
{
   "code": 200,
   "message": "success",
   "data": ["df152383d40842b0814346cc88cfd1b4"]
}
```

异常返回示例：

```
{
   "code": 28420,
   "message": "No access to alarm rule",
   "localizedMsg": "无权访问该告警规则"
}
```

2.6 批量更新告警记录确认状态

路径	/alarm/record/confirm
版本号	1.0.0
协议	HTTP, HTTPS
请求方法	POST
授权类型	app签名
超时时间	10000

请求参数

名称	类型	是否必选	描述
alarmRecordIds	JSONArray<Long>	必填	告警记录id
confirm	Integer	必填	确认状态，0未确认，1已确认

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	BOOL	响应结果：True or False

示例

请求示例：

```
// https://github.com/aliyun/iotx-api-gateway-client
IoTApiClientBuilderParams ioTApiClientBuilderParams = new IoTApiClientBuilderParams();

ioTApiClientBuilderParams.setAppKey("你的<AppKey>");
ioTApiClientBuilderParams.setAppSecret("你的<AppSecret>");

SyncApiClient syncApiClient = new SyncApiClient(ioTApiClientBuilderParams);

IoTApiRequest request = new IoTApiRequest();

// 设置请求ID
String uuid = UUID.randomUUID().toString();
String id = uuid.replace("-", "");
request.setId(id);
// 设置API版本号
request.setApiVer("1.0.0");
// 设置参数
request.putParam("alarmRecordIds","value1");
request.putParam("confirm","value2");
// 如果需要，设置headers
Map<String, String> headers = new HashMap<String, String>(8);
// headers.put("你的<header", "你的<value>");

// 设置请求参数域名、path、request , isHttps, headers
ApiResponse response = syncApiClient.postBody("api.link.aliyun.com", "/alarm/record/confirm", request, true, headers);

System.out.println(
    "response code = " + response.getCode()
        + " response = " + new String(response.getBody(), "UTF-8")
        + " headers = " + response.getHeaders().toString()
);
```

请求入参示例：

```
{
   "alarmRecordIds": [1000000001, 1000000002],
   "confirm": 1
}
```

正常返回示例：

```
{
    "id": "4de2c367-c1db-417c-aa15-8c585e595d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": true
}
```

异常返回示例：

```
{
   "code": 28403,
   "message": "The project not exist",
   "localizedMsg": "该项目不存在"
}
```

3. 数据模型定义

接入方式	数据模型
模型ID	AlarmRecord
模型版本号	1.1
模型描述	告警规则产生的记录

属性列表

参数	是否必选	类型	描述
alarmName	非必选	String	告警规则名称
deviceName	非必选	String	设备deviceName
productKey	非必选	String	设备productKey
productName	非必选	String	产品名称
alarmLevel	非必选	Integer	告警等级，0严重，1高，2中，3低
auxiliaryLevel	非必选	Integer	辅助等级，客户自定义
recovery	非必选	Integer	是否恢复，0未恢复，1已恢复
confirm	非必选	Integer	是否确认，0未确认，1已确认
recoveryTime	非必选	Date	恢复时间，毫秒级时间戳
confirmTime	非必选	Date	确认时间，毫秒级时间戳
alarmContent	非必选	String	告警内容描述
alarmType	非必选	Integer	告警类型，1运行告警，2故障告警
alarmMode	非必选	Integer	告警模式，1阈值告警，2限值告警
restrictAlarmType	非必选	Integer	限值告警类型，1高限值告警，2低限值告警

extConfig	非必选	String	扩展配置，用于回填
time	非必选	Date	告警产生时间，毫秒级时间戳
operator	非必选	String	操作人员
sceneId	非必选	String	场景id
alarmId	非必选	String	告警记录id
confirmType	非必选	Integer	确认类型。1故障，2误报，3报警
confirmComment	非必选	String	确认告警的备注信息

4. 附录

sceneConfig和lowSceneConfig格式样例

1. 水位从<=80%变化到>80%告警，从>80%变化到<=80%取消告警

```
{
"type": "IFTTT",
"trigger": {
"uri": "trigger/device/property",
"params": {
"productKey": "MyProduct",
"deviceName": "MyDevice",
"propertyName": "level",
"compareType": ">",
"compareValue": 80
}
},
"action": [
{
"uri": "action/mqtt/pub/sceneInfo",
"params": {
"env": "edge"
}
}
],
"features": {
"supportPauseRule": {
"pauseType": "time",
"pauseParams": {},
"scope": "device",
"recoverCondition": "event.level.value <= 80", // Must be the opposite of the condition in trigger.
"notifyTypeOnRecover": "MQTT",
"notifyTopicOnRecover": "Ignored"
}
}
}
```

2.温度>80度持续15分钟告警，温度<=80度取消告警

```
{
"type": "IFTTT",
"trigger": {
"uri": "trigger/device/property",
"params": {
"productKey": "MyProduct",
"deviceName": "MyDevice",
"propertyName": "temp",
"compareType": ">",
"compareValue": 80
}
},
"action": [
{
"uri": "action/mqtt/pub/sceneInfo",
"params": {
"env": "edge"
}
}
],
"features": {
"supportPauseRule": {
"pauseType": "time",
"pauseParams": {
"stateDuration": 900, //the continuous time of a state, 1s <= time <= 3600s
},
"scope": "device",
"recoverCondition": "event.temp.value <= 80", // Must be the opposite of the condition in trigger.
"notifyTypeOnRecover": "MQTT",
"notifyTopicOnRecover": "Ignored"
}
}
}
```

3.水位从<=20cm变化到>20cm告警，从>20cm变化到<=20cm取消告警

```
{
"type": "IFTTT",
"trigger": {
"uri": "trigger/device/property",
"params": {
"productKey": "MyProduct",
"deviceName": "MyDevice",
"propertyName": "level",
"compareType": ">",
"compareValue": 20
}
},
"action": [
{
"uri": "action/mqtt/pub/sceneInfo",
"params": {
"env": "edge"
}
}
],
"features": {
"supportPauseRule": {
"pauseType": "time",
"pauseParams": {},
"scope": "device",
"recoverCondition": "event.level.value <= 20", // Must be the opposite of the condition in trigger.
"notifyTypeOnRecover": "MQTT",
"notifyTopicOnRecover": "Ignored"
}
}
}
```

4.消防点位状态从非动作信号变化为动作信号告警，从动作信号变为非动作信号取消告警

```
{
"type": "IFTTT",
"trigger": {
"uri": "trigger/device/property",
"params": {
"productKey": "MyProduct",
"deviceName": "MyDevice",
"propertyName": "state",
"compareType": ">",
"compareValue": 1 // 1 for action signal, others for non action signal.
}
},
"action": [
{
"uri": "action/mqtt/pub/sceneInfo",
"params": {
"env": "edge"
}
}
],
"features": {
"supportPauseRule": {
"pauseType": "time",
"pauseParams": {},
"scope": "device",
"recoverCondition": "event.level.value != 1", // Must be the opposite of the condition in trigger.
"notifyTypeOnRecover": "MQTT",
"notifyTopicOnRecover": "Ignored"
}
}
}
```

5.管网压力持续5分钟在[0.9, 1.2]之外告警，管网压力在[0.9, 1.2]之内取消告警

```
{
"type": "IFTTT",
"trigger": {
"uri": "trigger/device/property",
"params": {
"productKey": "MyProduct",
"deviceName": "MyDevice",
"propertyName": "pressure",
"compareType": "beyond",
"compareValue": [0.9, 1.2]
}
},
"action": [
{
"uri": "action/mqtt/pub/sceneInfo",
"params": {
"env": "edge"
}
}
],
"features": {
"supportPauseRule": {
"pauseType": "time",
"pauseParams": {
"stateDuration": 300, //the continuous time of a state, 1s <= time <= 3600s
},
"scope": "device",
"recoverCondition": "event.pressure.value >= 0.9 and event.pressure.value <= 1.2", // Must be the opposite of the condition in trigger.
"notifyTypeOnRecover": "MQTT",
"notifyTopicOnRecover": "Ignored"
}
}
}
```

6.水浸点>80连续5次告警，水浸点<=80度取消告警

```
{
"type": "IFTTT",
"trigger": {
"uri": "trigger/device/property",
"params": {
"productKey": "MyProduct",
"deviceName": "MyDevice",
"propertyName": "level",
"compareType": ">",
"compareValue": 80
}
},
"action": [
{
"uri": "action/mqtt/pub/sceneInfo",
"params": {
"env": "edge"
}
}
],
"features": {
"supportPauseRule": {
"pauseType": "count",
"pauseParams": {
"windowInterval": 60, // Set to maximum window interval. This is the minimum sensitivity.
"conditionMatchNum": 5,
"continuousMatch": true,
"executeOnce": true
},
"scope": "device",
"recoverCondition": "event.temp.value <= 80", // Must be the opposite of the condition in trigger.
"notifyTypeOnRecover": "MQTT",
"notifyTopicOnRecover": "Ignored"
}
}
}
```

7.夜间10点至次日早6点，门禁触发开门则告警

```
{
"type": "IFTTT",
"trigger": {
"uri": "trigger/device/property",
"params": {
"productKey": "MyProduct",
"deviceName": "MyDevice",
"propertyName": "door",
"compareType": "==",
"compareValue": 1
}
},
"condition": {
"uri": "condition/timeRange",
"params": {
"beginDate": "22:00:00",
"endDate": "06:00:00",
"format": "HH:mm:ss",
"timezoneID": "Asia/Shanghai"
}
},
"action": [
{
"uri": "action/mqtt/pub/sceneInfo,
"params": {
"env": "edge"
}
}
],
"features": {
"supportPauseRule": {
"pauseType": "time",
"pauseParams": {},
"scope": "device",
"recoverCondition": "event.door.value != 1", // Must be the opposite of the condition in trigger.
"notifyTypeOnRecover": "MQTT",
"notifyTopicOnRecover": "Ignored"
}
}
}
```

8.1分钟内温度>30度连续10次告警，温度<=30度取消告警

```
{
"type": "IFTTT",
"trigger": {
"uri": "trigger/device/property",
"params": {
"productKey": "MyProduct",
"deviceName": "MyDevice",
"propertyName": "temp",
"compareType": ">",
"compareValue": 30
}
},
"action": [
{
"uri": "action/mqtt/pub/sceneInfo",
"params": {
"env": "edge"
}
}
],
"features": {
"supportPauseRule": {
"pauseType": "count",
"pauseParams": {
"windowInterval": 1,
"conditionMatchNum": 10,
"continuousMatch": true,
"executeOnce": true
},
"scope": "device",
"recoverCondition": "event.temp.value <= 30", // Must be the opposite of the condition in trigger.
"notifyTypeOnRecover": "MQTT",
"notifyTopicOnRecover": "Ignored"
}
}
}
```

9.设备下线告警，设备上线恢复告警

```
{
"type": "IFTTT",
"trigger": {
"uri": "trigger/device/statusChange",
"params": {
"productKey": "MyProduct",
"deviceName": "MyDevice",
"status": 3
}
},
"action": [
{
"uri": "action/mqtt/pub/sceneInfo",
"params": {
"env": "edge"
}
}
],
"features": {
"supportPauseRule": {
"pauseType": "time",
"pauseParams": {},
"scope": "device",
"recoverCondition": "event.status.value != 3", // Must be the opposite of the condition in trigger.
"notifyTypeOnRecover": "MQTT",
"notifyTopicOnRecover": "Ignored"
}
}
}
```

3.4.6. 车行接口

3.4.6.1. 软硬分离方案

3.4.6.1.1. 服务总线开发指引

请参考阿里云文档：服务总线

3.4.6.1.2. 边缘文件服务接口

1.概述

边缘文件服务，使用HTTP restful API。

主机地址-根据实际情况配置

端口- 9102

2. 接口定义（HTTP）

主机地址-根据实际情况配置

端口- 9102

2.1 文件增加

路径	/home/paas/file/add
版本号	N/A
Content-Type	form-data

协议	HTTP
请求方法	POST

请求参数

名称	类型	是否必选	描述
filePath	Text	是	文件路径
file	File	是	文件流

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。
message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。
data	String	文件下载url 文件超出1小时会定期删除

示例

正常返回示例：

```
{
  "code": 200,
  "message": "success",
  "data": "http://x.x.x.x:9012/xxx/xxx.jpg"
}
```

2.2 文件删除

路径	/home/paas/file/delete
版本号	N/A
Content-Type	json
协议	HTTP
请求方法	POST

请求参数

名称	类型	是否必选	描述
filePath	Text	是	文件路径

返回数据

名称	类型	描述
code	Int	接口返回码。200表示成功。

message	String	调用失败时，返回的出错信息。
localizedMsg	String	本地语言的错误消息。

示例

请求入参示例：

```
{
"filePath":"aaa/bbb/ccc/jpg"
}
```

正常返回示例：

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

2.3 文件获取

接口定义：

路径	文件路径，如：/aaa/bbb/ccc.json
协议	HTTP
请求方法	GET

返回数据：

文件流

示例

请求示例：

```
http://x.x.x.x/aaa/bbb/ccc.json
```

3.4.6.1.3. 边缘设备接口

请参考阿里云文档：边缘应用 Link IoT Edge对接

3.4.6.2. 系统对接方案

3.4.6.2.1. 停车场系统领域服务-服务模型定义

服务模型相关说明 ：

服务模型相关说明	
服务模型名称	停车场系统领域模型
模型标识符	iotx_parking_service_model
发布模型版本	1.0

1. 停车场

1.1 查询停车场信息

停车场信息包含停车场名称、停车场id、车位总数量、车位空位数量

定义

path	版本	是否需要登录

/parkingLotInfoGet	1.0.0	否

入参

无参数

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	objectList	停车场信息，可能是多个停车场信息

ParkingLotInfoDTO对象

字段	类型	备注
name	String	停车场名称
parkingLotId	String	停车场id
totalCount	Integer	总车位数
unParkedCount	Integer	空余车位数

示例

```
{
  "code": 200,
  "data": [
    {
      "name":"户外停车场",
     "parkingLotId":"a34d9b31d931b",
     "totalCount":677,
     "unParkedCount":302
    },
  ],
  "message": null
}
```

1.2 查询停车场一级区域ID

定义

path	版本	是否需要登录
/parkingLotFirstAreaIdsGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id

示例

```
{
   "parkingLotId": "a34d9b31d931b"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	Array	停车场区域编码列表

示例

```
{
  "code": 200,
  "data":[
        "area_id_1",
        "area_id_2",
        ...
  ],
  "message": null
}
```

1.3 查询停车场道闸信息

本接口就提供停车场道闸的查询功能

定义

path	版本	是否需要登录
/parkingLotBarriersInfoGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id

示例

```
{
   "parkingLotId": "a34d9b31d931b"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	List	道闸的dto

BarrierDTO

barrierId	String	道闸id
barrierName	String	道闸名称
direction	String	IN表示进场道闸，OUT表示出场道闸

示例

```
{
  "code": 200,
  "data": [
    {
    "barrierId":"道闸id",
    "barrierName":"道闸名称",
    "direction":"IN"
  },
    ],
  "message": null
}
```

1.4 校正停车系统通行记录上的车牌号

如果车辆通过的时候，系统识别出来的车牌号和实际车牌号不符，调用该接口校正对应的车牌号。这个接口校正的是本地系统上一次通行记录中对应的车牌号。这个接口是为了解决本地系统上，通行记录里的错误车牌号会影响收费计算的问题。

定义

path	版本	是否需要登录
/parkingLotPlateNumberUpdate	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号
recordId	String	是	车辆通行上报的通行事件id。对应车辆通行数据模型 iot_park_pass_record 的recordId字段。

示例

```
{
  "parkingLotId":"xxxxxxxx",
  "plateNumber":"浙X XXXXX",
  "recordId":"a389FIUHDK"
}
```

出参

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

示例

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

2. 停车区域

停车区域是存在层级结构的，区域下可以挂区域，根区域挂在停车场下。

2.1 查询停车区域信息

停车区域信息包含停车区域名称、区域id、上层区域的id、下一层区域id的列表、空余车位数、总车位数。

定义

path	版本	是否需要登录
/parkingAreaInfoGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
areaId	String	是	停车区域id

示例

```
{
   "parkingLotId":"停车场id",
    "areaId": "d4d984jdlafjek"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	Object	停车场信息

AreaDTO

name	String	停车区域名称
areaId	String	区域id
parentAreaId	String	上层区域id
subAreaList	List	下一层区域id列表
unParkedCount	Integer	空余车位数
totalCount	Integer	总车位数

示例

```
{
  "code": 200,
  "data": {
    "name":"区域a",
    "areaId":"area1",
    "parentAreaId":"a34d9b31d931b",
    "SubAreaList":[
     "SubAreaId_1",
     "SubAreaId_2",
     "SubAreaId_3",
     ...
     ],
    "unParkedCount":30,
    "totalCount":67
  },
  "message": null
}
```

2.2 查询区域下的道闸列表

返回的是代表道闸的设备信息

定义

path	版本	是否需要登录
/parkingAreaBarriersInfoGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	车场id
areaId	String	是	停车区域id

示例

```
{
   "parkingLotId": "10",
   "areaId": "d4d984jdlafjek"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	List	返回的数据对象列表

BarrierDTO

barrierId	String	道闸id
barriereName	String	道闸名称
direction	String	IN表示进场道闸，OUT表示出场道闸。不填不会更新这个属性。

示例

```
{
  "code": 200,
  "data": [
    {
     "barrierId":"4a70179d",
      "barriereName":"xxxx",
      "direction":"IN"
   },
   ],
  "message": null
}
```

2.3 查询区域车位列表

这是一个全量列表，返回的是车位在本地系统中的编码，这个编码用于同步停车位

定义

path	版本	是否需要登录

/parkingAreaParkspacesGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	车场id
areaId	String	是	停车区域id

示例

```
{
   "parkingLotId":"d4d984jdlaf",
   "areaId": "d4d984jdlafjek"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	Array	返回的都是车位的id,放入数组里

示例

```
{
  "code": 200,
  "data": [
          "parkingSpaceId_1",
          "parkingSpaceId_2",
            ...
  ],
  "message": null
}
```

3. 道闸

3.1 查询道闸信息

定义

path	版本	是否需要登录
/parkingBarrierInfoGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
barrierId	String	是	道闸id

示例

```
{
  "parkingLotId":"xxxx",
  "barrierId":"xxx"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	Object	道闸的信息

BarrierDTO

areaId	String	道闸所属区域id
barrierId	String	道闸id
status	String	open道闸打开，close道闸关闭
direction	String	in表示进场道闸，out表示出场道闸。不填不会更新这个属性。
barrierName	String	道闸名称

示例

```
{
  "code": 200,
  "data": {
     "areaIdBarrierId":"4a70179d",
      "barrierId":"xxxx",
      "status":"open",
      "direction":"in",
      "barrierName":"道闸名称"
  },
  "message": null
}
```

3.2 手动打开道闸栏杆

定义

path	版本	是否需要登录
/parkingBarrierOperate	1.0.0	否

入参

字段	类型	是否必传	备注
barrierId	String	是	道闸Id
command	String	是	选项： open -- 打开 close -- 关闭 keepOpen -- 长开

示例

```
{
  "barrierId":"道闸唯一标识",
  "command":"open"
}
```

出参

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

示例

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

4. 停车位

4.1 更新停车位

定义

path	版本	是否需要登录
/parkingAreaParkspaceUpdate	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
areaId	String	是	区域id
parkingSpaceId	String	是	车位id
userName	String	否	用户姓名
parkingSpaceType	String	否	车位类型： private私人车位 public公共车位 consume消费车位 guest访客车位 selfdefined自定义 不填不更新
isSold	String	否	true 已售 false未售
property	String	否	guard人防 noguard非人防 不填不更新
ownerPhone	String	否	产权人手机号 不填不更新
ownerName	String	否	产权人姓名长度64字符 不填不更新

ownerId	String	否	产权人证件号 不填不更新

示例

```
{
  "parkingLotId": "ad8sdlfd",
  "parkingSpaceId": "ad8sdlfd",
  "userName":"用户姓名",
  "type": "A位",
  "isSold":"true/false",
  "property":"GUARD",
  "ownerPhone":"18967533306",
  "ownerName":"产权人姓名",
  "ownerId":"122345666"
}
```

出参

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

示例

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

4.2 查询车位信息

定义

path	版本	是否需要登录
/parkingAreaParkspaceGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
areaId	String	是	区域id
parkingSpaceId	String	是	车位id

示例

```
{
  "parkingLotId":"28384949",
  "areaId":"294949540",
  "parkingSpaceId": "vnidalwed746aosd"
}
```

出参

字段	类型	备注
code	Integer	返回码

message	String	返回信息
data	Object	车位信息

ParkingSpaceDTO

parkingSpaceId	String	车位的id
name	String	车位名称
type	String	车位类型： private私人车位 public公共车位 consume消费车位 guest访客车位 selfdefined自定义
isSold	Boolean	true 已售 false未售
property	String	guard人防 noguard非人防
ownerPhone	String	产权人手机号
ownerName	String	产权人姓名长度64字符
ownerId	String	产权人证件号
isParking	Boolean	true有车在停泊，false无车
plateNumber	String	停泊车的车牌号

示例

```
{
  "code": 200,
  "data": {
    "parkingSpaceId":"vnidalwed746aosd",
    "parkingLotId":"a34d9b31d93lb",
    "type":"a34d9b31d93lb",
    "isSold":"true",
    "property":"PRIVATE",
    "ownerPhone":"13939495959",
    "ownerName":"老王",
    "ownerId":"2839494",
    "isParking":"true",
    "plateNumber":"京N TR669"
  },
  "message": null
}
```

5. 车辆管理

5.1 添加车辆

定义

path	版本	是否需要登录
/parkingLotVehicleAdd	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号
userName	String	否	用于显示的车主名字
userPhone	String	否	车主联系电话
userId	String	否	车主证件号

示例

```
{
  "parkingLotId": "d4d984jdlafjek",
  "plateNumber":"京N TR666",
  "userName": "王生",
  "userPhone":"13888888888",
  "userId":"10xxxxxxxxxxx9876X"
}
```

出参

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

示例

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

5.2 删除车辆

定义

path	版本	是否需要登录
/parkingLotVehicleDelete	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号

示例

```
{
  "parkingLotId": "d4d984jdlafjek",
  "licenseNumber":"京N TR666"
}
```

出参

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

示例

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

5.3 更新车辆信息

定义

path	版本	是否需要登录
/parkingLotVehicleUpdate	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场的id，创建停车场时的id
plateNumber	String	是	车牌号
userName	String	否	车主名字，不传不修改
userPhone	String	否	车主联系电话，不传不修改
userId	String	否	车主证件号，不传不修改

示例

```
{
  "parkingLotId": "d4d984jdlafjek",
  "plateNumber":"京N TR666",
  "userName": "王生",
  "userPhone":"13888888888",
  "userId":"10xxxxxxxxxxx9876X"
}
```

出参

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

示例

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

5.4 停车场内的车辆列表

返回车牌号的列表

定义

path	版本	是否需要登录
/parkingLotVehicleListInfoGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场Id
pageSize	Integer	是	分页查询,每页显示条数上限100
pageNum	Integer	是	分页查询，当前页数从1开始

示例

```
{
  "parkingLotId": "d4d984jdlafjek",
  "pageSize":20,
  "pageNum":20
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	List	车牌列表

车牌列表

plateNumberList	List	车牌列表

示例

```
{
  "code": 200,
  "data": [
        "浙A5566",
        "浙A5567",
        "浙A5568",
       ...
 ],
  "message": null
}
```

5.5查询停车场的车辆信息

定义

path	版本	是否需要登录
/parkingLotVehicleInfoGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号

示例

```
{
  "parkingLotId": "d4d984jdlafjek",
  "plateNumber":"京N TR666"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	Object	车辆信息

VehicleDTO

字段	类型	备注
parkingLotId	String	停车场的id，创建车位的时候获取
userName	String	车主名字，不传不修改
userPhone	String	车主联系电话，不传不修改
userId	String	车主证件号，不传不修改
plateNumber	String	车牌号
parkingSpaceId	String	停泊的车位id

示例

```
{
  "code": 200,
  "data":{
      "parkingLotId": "d4d984jdlafjek",
      "userName": "王生",
      "userPhone":"13888888888",
      "userId":"10xxxxxxxxxxx9876X",
      "plateNumber":"京N TR666",
      "parkingSpaceId":"509x480"
  },
  "message": null
}
```

5.6 锁定车辆

定义

path	版本	是否需要登录
/parkingLotVehicleLock	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场的id
plateNumber	String	是	车牌号
lockEffectiveDate	String	是	锁车的开始时间，时间格式 yyyy-MM-dd HH:mm:ss"
lockExpiredDate	String	是	锁车的结束时间，时间格式 yyyy-MM-dd HH:mm:ss"

示例

```
{
   "parkingLotId": "d4d984jdlafjek",
   "plateNumber":"京N TR666",
   "lockEffectiveDate":"2019-01-01 00:00:00",
   "lockExpiredDate":"2021-12-31 23:59:59"
}
```

出参

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

示例

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

5.7 查询车场中锁定车辆的信息

定义

path	版本	是否需要登录
/parkingLotVehicleLockGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号

示例

```
{
  "parkingLotId":"停车场id",
  "plateNumber":"京A 5938"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	Object	锁定车辆信息

VehicleDTO

字段	类型	备注
lockEffectiveDate	String	锁车的开始时间，时间格式 yyyy-MM-dd HH:mm:ss"
lockExpiredDate	String	锁车的结束时间，时间格式 yyyy-MM-dd HH:mm:ss"

```
{
  "code": 200,
  "data":{
    "lockEffectiveDate":"2019-01-01 00:00:00",
    "lockExpiredDate":"2021-12-31 23:59:59"
  },
  "message": null
}
```

5.8 取消锁定车辆

定义

path	版本	是否需要登录
/parkingLotVehicleUnlock	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号

示例

```
{
  "parkingLotId":"停车场id",
  "plateNumber":"京A 5938"
}
```

出参

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

示例

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

5.9 查询车辆入场时间

path	版本	是否需要登录
/parkingLotVehicleInTime	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	车场id
plateNumber	String	是	车牌号

示例

```
{
 "parkingLotId":"349494590",
  "plateNumber":"京A384859"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	Object	VehicleDTO

VehicleInTimeDTO

字段	类型	备注
vehicleInTime	String	车辆入场时间，时间格式*yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss*

```
{
  "code": 200,
  "data":{
      "2019-06-03 17:53:04"
  },
  "message": null
}
```

6 车辆权限

6.1 车辆对道闸授权

说明：

如权限已存在，则更新权限

定义

path	版本	是否需要登录

/parkingVehiclePermissionAdd	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
effectiveDate	String	是	*车牌信息有效期的开始yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
expiryDate	String	是	*车牌信息有效期的终止yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
plateNumber	String	是	车牌号
barrierList	List	是	道闸列表

示例

```
{
  "parkingLotId":"73849593",
  "effectiveDate": "2019-01-01 00:00:00",
  "expiryDate": "2019-01-01 00:00:00",
  "plateNumber": "京N12345",
  "barrierList": [
      "xxx"
  ]
}
```

出参

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

示例

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

6.2 车辆对道闸权限查询

定义

path	版本	是否需要登录
/vehicleBarrierPermissionGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id

plateNumber	String	是	车牌号

示例

```
{
  "parkingLotId":"37484949",
  "plateNumber": "京N12345"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	List<VehiclePermissionDTO>	车辆权限信息

VehiclePermissionDTO

字段	类型	备注
plateNumber	String	车牌号
effectiveDate	String	*车牌信息有效期的开始时间yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
expiryDate	String	*车牌信息有效期的终止时yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
barrierId	String	道闸Id

示例

```
{
  "code": 200,
  "message": null,
  "data": {
    "plateNumber": "浙A54321",
    "expiryDate": "2019-06-03 17:53:04",
    "effectiveDate": "2019-06-03 17:53:04",
    "barrierId": "xxx"
  }
}
```

6.3 车辆对道闸权限修改

说明：

车辆对哪些道闸具有权限

定义

path	版本	是否需要登录
/vehicleBarrierPermissionUpdate	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id

plateNumber	String	是	车牌号
effectiveDate	String	是	*车牌信息有效期的开始yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
expiryDate	String	是	*车牌信息有效期的终止yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*

示例

```
{
  "parkingLotId":"23849499",
  "plateNumber": "京N12345",
  "effectiveDate": "2019-01-01 00:00:00",
  "expiryDate": "2020-01-01 00:00:00"
}
```

出参

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

示例

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

6.4 车辆对道闸权限删除

定义

path	版本	是否需要登录
/vehicleBarrierPermissionDelete	1.0.0	是

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号
barrierId	String	是	道闸Id

示例

```
{
  "parkingLotId":"26374898",
  "plateNumber": "京N12345",
  "barrierId": "xxx"
}
```

出参

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

示例

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

6.5 车辆对区域授权

1、对区域下所有设备增加权限

2、权限已存在则更新

3、设备为空返回失败

定义

path	版本	是否需要登录
/vehicleAreaPermissionAdd	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
effectiveDate	String	是	*车牌信息有效期的开始yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
expiryDate	String	是	*车牌信息有效期的终止yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
plateNumber	String	是	车牌号
areaIds	List	是	停车区域的列表

示例

```
{
  "parkingLotId":"6748488",
  "effectiveDate": "2019-01-01 00:00:00",
  "expiryDate": "2020-01-01 00:00:00",
  "plateNumber": "京N12345",
  "areaIds": [
      "xxx",
  ]
}
```

出参

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

示例

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

6.6 车辆对区域权限查询

定义

path	版本	是否需要登录
/vehicleAreaPermissionGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号

示例

```
{
  "parkingLotId":"6748488",
  "plateNumber": "京N12345"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	VehiclePermissionDTO	车辆权限信息

VehiclePermissionDTO

字段	类型	备注
plateNumber	String	车牌号
effectiveDate	String	*车牌信息有效期的开始时间yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
expiryDate	String	*车牌信息有效期的终止时yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
areaId	String	区域Id

示例

```
{
  "code": 200,
  "message": null,
  "data": {
    "plateNumber": "浙A54321",
    "expiryDate": "2019-06-03 17:53:04",
    "effectiveDate": "2019-06-03 17:53:04",
    "areaId": "xxx"
  }
}
```

6.7 车辆对区域权限修改

定义

path	版本	是否需要登录
/vehicleAreaPermissionUpdate	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号
effectiveDate	String	是	*车牌信息有效期的开始yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
expiryDate	String	是	*车牌信息有效期的终止yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*

示例

```
{
  "parkingLotId":"6748488",
  "plateNumber": "京N12345",
  "effectiveDate": "2019-01-01 00:00:00",
  "expiryDate": "2020-01-01 00:00:00"
}
```

出参

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

示例

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

6.8 车辆对区域权限删除

定义

path	版本	是否需要登录

/vehicleAreaPermissionDelete	1.0.0	是

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号
areaId	String	是	区域Id

示例

```
{
  "parkingLotId":"6748488",
  "plateNumber": "京N12345",
  "areaId": "xxx"
}
```

出参

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

示例

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

6.9 停车区域车辆权限列表查询

定义

path	版本	是否需要登录
/vehicleListAreaPermissionGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场Id
areaId	String	否	区域Id，不传代表全部区域
pageNum	Integer	是	分页查询，当前页数从1开始
pageSize	Integer	是	分页查询，每页显示条数20条

示例

```
{
   "parkingLotId": "xxx",
    "areaId": "xxx",
    "pageNum":20,
   "pageSize":20
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	List	车辆权限信息

VehiclePermissionDTO

字段	类型	备注
plateNumber	String	车牌号
effectiveDate	String	*车牌信息有效期的开始时间yyyy-MM-dd HH:mm:ss 或yyyy/MM/dd HH:mm:ss 等标准事件格式*
expiryDate	String	*车牌信息有效期的终止时间yyyy-MM-dd HH:mm:ss 或yyyy/MM/dd HH:mm:ss 等标准事件格式*
parkingLotId	String	车场Id
areaId	String	区域Id

示例

```
{
  "code": 200,
  "message": null,
  "data": [{
    "expiryDate": "2019-06-03 17:53:04",
    "effectiveDate": "2019-06-03 17:53:04",
    "plateNumber": "浙A54321",
    "parkingLotId": "xxx",
    "areaId":"xxx"
  },{
    "expiryDate": "2019-06-03 17:53:04",
    "effectiveDate": "2020-06-03 17:53:04",
    "plateNumber": "京A54321",
    "parkingLotId": "xxx",
    "areaId":"xxx"
  }]
}
```

6.10 停车道闸车辆权限列表查询

定义

path	版本	是否需要登录
/vehicleBarrierPermissionInfoGet	1.0.0	否

入参

字段	类型	是否必传	备注

parkingLotId	String	是	停车场Id
barrierId	String	否	道闸Id，不传代表全部区域
pageNum	Integer	是	分页查询，当前页数从1开始
pageSize	Integer	是	分页查询，每页显示条数20条

示例

```
{
   "parkingLotId": "xxx",
    "barrierId": "xxx",
    "pageNum":20,
   "pageSize":20
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	List	车辆权限信息

VehiclePermissionDTO

字段	类型	备注
plateNumber	String	车牌号
effectiveDate	String	*车牌信息有效期的开始时间yyyy-MM-dd HH:mm:ss 或yyyy/MM/dd HH:mm:ss 等标准事件格式*
expiryDate	String	*车牌信息有效期的终止时间yyyy-MM-dd HH:mm:ss 或yyyy/MM/dd HH:mm:ss 等标准事件格式*
parkingLotId	String	车场Id
barrierId	String	道闸Id

示例

```
{
  "code": 200,
  "message": null,
  "data": [{
    "expiryDate": "2019-06-03 17:53:04",
    "effectiveDate": "2019-06-03 17:53:04",
    "plateNumber": "浙A54321",
    "parkingLotId": "xxx",
    "barrierId":"xxx"
  },{
    "expiryDate": "2019-06-03 17:53:04",
    "effectiveDate": "2020-06-03 17:53:04",
    "plateNumber": "京A54321",
    "parkingLotId": "xxx",
    "barrierId":"xxx"
  }]
}
```

7 黑白名单

7.1 车辆加入安全黑/白名单

定义

path	版本	是否需要登录
/parkingLotVehicleListAdd	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	List	是	车牌号
areaId	List	否	区域Id
type	String	是	WHITELIST/BLACKLIST 白/黑名单
effectiveDate	String	否	*黑白名单有效期的开始时间yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
expiryDate	String	否	*黑白名单有效期的终止时间yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*

示例

```
{
  "parkingLotId":"6748488",
  "plateNumber": [
    plateNumber1
  ],
  "areaId": [
    "xxx"
  ],
  "type": "WHITELIST",
  "effectiveDate":"2019-01-01 00:00:00",
  "expiryDate":"2020-01-01 00:00:00"
}
```

出参

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

示例

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

7.2 车辆从黑/白名单移除

定义

path	版本	是否需要登录

/parkingLotVehicleListRemove	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号
areaId	List	否	区域Id

示例

```
{
  "parkingLotId":"6748488",
  "plateNumber": "京N12345",
  "areaId": [
    "xxx",
  ],
}
```

出参

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

示例

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

7.3 车辆属于黑/白名单查询

定义

path	版本	是否需要登录
/parkingLotVehicleInListGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号
areaId	String	否	区域Id

示例

```
{
  "parkingLotId":"6748488",
  "plateNumber": "京N12345",
  "areaId": "xxx"
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	VehicleSecurityDTO	车辆权限信息

VehiclePermissionDTO

字段	类型	备注
plateNumber	String	车牌号
areaId	String	区域Id
type	String	whitelist 白名单 blacklist 黑名单
effectiveDate	String	*黑白名单有效期的开始时间yyyy-MM-dd HH:mm:ss 或yyyy/MM/dd HH:mm:ss 等标准事件格式*
expiryDate	String	*黑白名单有效期的终止时间yyyy-MM-dd HH:mm:ss 或yyyy/MM/dd HH:mm:ss 等标准事件格式*

示例

```
{
  "code": 200,
  "message": null,
  "data": {
    "plateNumber": "浙A54321",
    "areaId": "xxx",
    "type": "whitelist",
    "effectiveDate":"2019-01-01 00:00:00",
    "expiryDate":"2021-12-31 23:59:59"
  }
}
```

7.4 分页查询黑/白名单车辆

定义

path	版本	是否需要登录
/parkingLotVehicleListGet	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
pageNum	Integer	是	分页查询，当前页数从1开始
pageSize	Integer	是	分页查询，每页显示条数20条
areaId	String	否	区域Id

type	String	否	whitelist 白名单 blacklist 黑名单

示例

```
{
  "parkingLotId":"6748488",
  "areaId": "xxx",
  "type": "blacklist",
  "pageNum":20,
  "pageSize":20
}
```

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	List	车辆权限信息

VehicleSecurityDTO

字段	类型	备注
plateNumber	String	车牌号
areaId	String	区域空间Id
type	String	whitelist 白名单 blacklist 黑名单
effectiveDate	String	*黑白名单有效期的开始时间yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*
expiryDate	String	*黑白名单有效期的终止时间yyyy-MM-dd HH:mm:ss 或 yyyy/MM/dd HH:mm:ss 等标准事件格式*

示例

```
{
  "code": 200,
  "message": null,
  "localizedMsg": null,
  "data": [{
    "plateNumber": "浙A54321",
    "type": "whitelist",
    "areaId": "xxx",
    "effectiveDate":"2019-01-01 00:00:00",
    "expiryDate":"2021-12-31 23:59:59"
  },{
    "plateNumber": "京A54321",
    "type": "blacklist",
    "areaId": "xxx",
    "effectiveDate":"2019-01-01 00:00:00",
    "expiryDate":"2021-12-31 23:59:59"
  }]
}
```

8 订单系统

8.1 新增车辆订单

定义

path	版本	是否需要登录
/parkingLotVehicleOrderAdd	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
plateNumber	String	是	车牌号
inBarrierGateId	String	是	入场道闸Id
inTime	String-utc	是	入场时间
payStatus	Integer	是	支付结果： 0 - 支付完成 4 - 未支付
outBarrierGateId	String	否	出场道闸Id，场内支付则为空
outTime	String-utc	否	出场时间，场内支付则为空

示例

```
{
  "parkingLotId":"6748488",
  "plateNumber": "京N12345",
  "outBarrierGateId": "xxx",
  "inTime": "2020-01-02 00:00:00",
  "outBarrierGateId": "xxx",
  "outTIme": "2020-01-02 00:00:00",
  "orderStatus": 0
}
```

出参

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

示例

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

8.2 确认车辆订单

定义

path	版本	是否需要登录
/parkingLotVehicleOrderCompleted	1.0.0	否

入参

字段	类型	是否必传	备注
parkingLotId	String	是	停车场id
areaId	String	否	停车区域Id
orderNumber	String	是	订单号
payStatus	Integer	是	支付结果：0 - 支付完成1 - 免费2 - 本地支付3 - 取消订单

示例

```
{
  "parkingLotId":"6748488",
  "orderNumber": "O4jRKfKLzpGWX4p4dcpZ000101",
  "areaId": "xxx",
  "payStatus": 1
}
```

出参

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

示例

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

9 错误码定义

结果码	结果说明	
0	成功	success
101	不存在停车场	There is no ParkingLot
102	停车场ID不存在	ParkingLot ID does not exist
103	订单号不存在	Order number does not exist
200	成功	success
200	车辆加入名单成功	Vehicle added to list successfully
200	车辆预约成功	Vehicle reservation successful
201	不存在停车区域	There is no Parking Area
202	停车区域ID不存在	Parking Area ID does not exist
301	不存在停车道闸	There is no Parking Barrier

302	停车道闸ID不存在	Parking Barrier ID does not exist
403	操作道闸失败	Barrier operation failed
401	不存在停车位	There is no Parking space
402	停车位ID不存在	Parking space ID does not exist
501	不存在车辆列表	There is no plate number list
502	车牌号不存在	Plate Number does not exist
503	车辆未锁定	The vehicle is not locked
504	停车场已存在该车辆	There is this vehicle in the parking lot already
601	通行事件ID不存在	Park Recorder ID does not exist
602	用户名不存在	User name does not exist
603	车辆权限不存在	There is no permission for this vehicle
604	该区域没有授权车辆列表	There is no vehicle permission list in the area
605	该道闸没有授权车辆列表	There is no vehicle permission list in the parking barrier
606	时间格式不正确	Date format error
607	起始时间晚于结束时间	EffectiveDate is later than expiryDate
701	车辆不在黑白名单里	There is no this vehicle in whitelist or blacklist
702	车辆已存在黑白名单中	There is this vehicle in whitelist or blacklist already
1000	未知错误	Unknown error
1001	请求忙	Timeout error
1002	请求参数无效	Input parameter is invalid
1003	道闸状态异常	Parking barrier is in abnormal status.
1004	设备不在线	Device is offline

3.4.6.2.2. 停车场系统领域服务-数据模型定义

1 车辆通行

当车辆通过停车区域的时候，会上报车辆通行的数据

模型名称：iot_park_pass_record

版本 1.0

字段	数据类型	字段含义

direction	Integer	"0":"进场" "1":"出场" "2":"未知"
openType	Integer	"0":"正常抬杆" "1":"异常抬杆" "2":"未知"
plateNumber	String	车牌号
plateNumberImage	String	车牌号照片文件名 <需要上传文件 类型:图片>
typePermission	Integer	"0":"未知类型" "1":"普通车辆" "2":"临时车辆" "3":"白名单"
plateColor	Integer	"0": "未知车牌颜色", "1": "蓝色", "2": "黄色", "3": "白色", "4": "黑色", "5": "绿色", "6": "黄绿色"
plateType	Integer	"0": "未知车牌类型", "1": "蓝牌车", "2": "单层黄牌车", "4": "警车", "8": "武警车辆", "16": "双层黄牌", "32": "摩托车", "64": "教练车", "128": "军车", "256": "个性化车", "512": "港澳车", "1024": "使馆车", "2048": "老式车牌", "4096": "民航车牌", "8192": "新能源车牌-小型", "9001": "香港进出中国大陆车牌", "9002": "农用车牌", "9003": "澳门进出中国大陆车牌", "9004": "无牌车"

vehicleColor	Integer	"0": "未知颜色", "1": "红色", "2": "绿色", "3": "蓝色", "4": "黄色", "5": "白色", "6": "灰色", "7": "黑色", "8": "紫色", "9": "棕色", "10": "粉色"
vehicleType	Integer	"0": "未知", "1": "轿车", "2": "面包车", "3": "大型客车", "4": "中型客车", "5": "皮卡", "6": "非机动车", "7": "SUV", "8": "MPV", "9": "微型货车", "10": "轻型货车", "11": "中型货车", "12": "重型货车"
barrierId	String	车闸的唯一标识车闸id
barrierName	String	车闸的名称
parkingLotId	String	车场id，车辆进出的停车场id
areaId	String	停车区域id，车辆进出的停车区域id
orderNumber	String	订单编号
recordId	String	本地系统上中车辆通行的记录号，用于可能需要的车牌校正功能
eventTime	String	本地系统产生通行事件的时间，时间格式为yyyy-MM-dd HH:mm:ss

2 收费事件

车辆在车闸处产生收费事件之后推送的数据

模型名称：iot_park_fee

版本 1.0

字段	数据类型	字段含义
outBarrierName	String	出场道闸名称
outBarrierId	String	出场道闸id
inBarrierName	String	进场道闸名称

inBarrierId	String	进场道闸id
payStatus	Integer	"0":"支付完成", "4":"未支付"
orderNumber	String	订单号
amount	Float	订单金额。单位元。最大99999.00，最小0.00，分辨率0.01。
outTime	String	出场时间
inTime	String	进场时间
plateNumber	String	车牌号
outImage	String	出场车牌照片文件名 <需要上传文件 类型:图片>
inImage	String	入场车牌照片文件名 <需要上传文件 类型:图片>
areaId	String	产生收费的区域id
parkingLotId	String	产生收费的车场id
eventTime	String	收费事件发生的时间，时间格式为yyyy-MM-dd HH:mm:ss

3 锁定车辆识别

当一个车辆被标记成锁定车辆通过了一个车闸的时候，上报该数据

模型名称：iot_park_locked_detect

版本 1.0

字段	数据类型	字段含义
direction	Integer	"0":"进场","1":"出场"
plateNumberImage	String	车牌号照片文件名 <需要上传文件 类型:图片>
plateNumber	String	车牌号
barrierId	String	通过的道闸id
barrierName	String	通过的道闸名称
parkingLotId	String	车场id，车辆进出的停车场id
areaId	String	停车区域id，车辆进出的停车区域id
eventTime	String	锁定车辆识别事件发生的时间，时间格式是yyyy-MM-dd HH:mm:ss

4 黑名单车辆识别

当一个被加入黑名单的车辆通过了一个车闸的时候，上报该数据

模型名称：iot_park_blacklist_detect

版本 1.0

字段	数据类型	字段含义
direction	Integer	"0":"进场","1":"出场"
plateNumberImage	String	车牌号照片文件名 <需要上传文件 类型:图片>
plateNumber	String	车牌号
barrierId	String	通过的道闸id
barrierName	String	通过的道闸名称
parkingLotId	String	车场id，车辆进出的停车场id
areaId	String	停车区域id，车辆进出的停车区域id
eventTime	String	黑名单车辆识别事件发生的时间，时间格式为yyyy-MM-dd HH:mm:ss

5 车位状态变化

当车位有车停泊，或者停泊的车离开的时候，上报该数据

模型名称：iot_park_parkinglot_update

版本 1.0

字段	数据类型	字段含义
stateParked	Integer	状态变成 "0": "无车","1": "有车"
parkedNumber	String	驻车车牌
parkingSpace	String	车位id
areaId	String	车位所属停车区域的id
parkingLotId	String	车位所属停车场的id
eventTime	String	车位状态变化发生的时间，时间格式为yyyy-MM-dd HH:mm:ss

6 停车场车辆信息变化

当停车场内的车辆信息发生变化的时候，上报该数据

模型名称：iot_park_vehicle_update

版本 1.0

字段	数据类型	字段含义
userName	String	车主称谓
plateNumber	String	车牌号

typeOperation	Integer	"0":"新增","1":"修改","2":"删除"
userId	String	车主的身份识别码
stateLocked	Integer	锁定状态 "0":"没有锁定","1":"锁定"
accessBarrierListFile	String	车辆可通行的车闸列表文件名 <需要上传文件 类型:文本>
parkingLotId	String	车辆所属停车场的id
eventTime	String	停车场车辆信息变化发生的时间，时间格式为yyyy-MM-dd HH:mm:ss

*车辆权限列表格式

```
{
  "PermissionList":[
    {"barrierId": "道闸id", "barrierName": "道闸名称","StartExpireTime":"起始时间","EndExpireTime": "截止时间"},
    {"barrierId": "道闸id", "barrierName": "道闸名称","StartExpireTime":"起始时间","EndExpireTime": "截止时间"},
    {"barrierId": "道闸id", "barrierName": "道闸名称","StartExpireTime":"起始时间","EndExpireTime": "截止时间"},
      ...
   ]
 }
```

7 停车场车辆锁定信息

当停车场内有车辆被锁定的时候，上报该数据

模型名称：iot_park_vehicle_lock_update

版本 1.0

字段	数据类型	字段含义
lockStartExpireTime	String	锁车开始时间
lockEndExpireTime	String	锁车过期时间
plateNumber	String	车牌号
parkingLotId	String	车场id
eventTime	String	停车场车辆锁定发生的时间，时间格式为yyyy-MM-dd HH:mm:ss

8 黑名单车辆更新

当本地停车系统更新了黑名单车辆之后，会上报这个数据

模型名称：iot_park_blacklist_update

版本 1.0

字段	数据类型	字段含义
plateNumber	String	车牌号
type	Integer	"0":"新增","1":"删除"
areaId	String	车位所属停车区域id

parkingLotId	String	车辆所属停车场id
eventTime	String	黑名单车辆更新时间，时间格式为yyyy-MM-dd HH:mm:ss

9 白名单车辆更新

当本地停车系统更新了白名单车辆之后，会上报这个数据

模型名称：iot_park_whitelist_update

版本 1.0

字段	数据类型	字段含义
plateNumber	String	车牌号
type	Integer	"0":"新增","1":"删除"
areaId	String	车位所属停车区域的id
parkingLotId	String	车辆所属停车场的id
eventTime	String	白名单车辆更新时间，时间格式为yyyy-MM-dd HH:mm:ss

3.4.7. 边缘托管应用开发指南

3.4.7.1. 应用开发手册

修改记录及版本信息

1 基本概念说明

本文提到的边缘应用，是指部署在物业管理一体机的边缘应用，是基于IoT的领域服务对接方案实现的边缘应用。领域服务对接方案里有服务模型定义和数据模型定义，下面介绍一些相关的基本概念：

服务模型：

是一组可提供完整业务功能的HTTP/HTTPS接口。边缘应用开发者可以根据对应的场景需求和业务需求进行这套接口定义。

服务提供方：

服务提供方即是可以提供服务模型里定义的服务功能的应用。服务提供方即可以是云端应用，也可以是边缘端应用

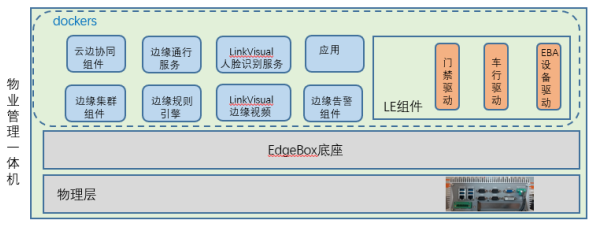
服务依赖方：

服务依赖方即是使用服务模型里定义的服务功能的应用。服务依赖方也可以是云端应用或边缘端应用

数据模型：

- 使用场景：本地系统/设备上报的消息可以基于数据模型进行定义，例如人脸通行事件
- 通过数据模型和IoT数据总线机制，可实现应用间的数据信息流转
- (数据提供方)应用可以通过数据添加的API接口，将本地系统或设备的消息送入数据总线。
- (数据消费方)应用可以通过查询数据模型的API接口获取消息，也可以通过数据变更消息订阅方式获取消息。
- 数据模型支持图片文件上传，例如人脸通行事件的人脸照片

物业管理一体机框架：



物业管理一体机是基于K8S框架实现的，底层是托管底座，之上是基于docker的各种应用和服务。

LE组件是基于docker的服务程序，里面包含了支持各种设备接入驱动，例如门禁驱动，车行驱动，EBA设备驱动等

边缘应用也是基于docker的应用程序。边缘应用可通过编译出的jar包，打包成镜像，然后通过IoT云端平台将应用下发到指定的物业管理一体机。边缘应用的启动入口，可以通过应用jar包里的docker file指定。

2 整体架构

2.1 核心模块功能说明

云端应用：即SaaS应用，一般由ISV提供，是服务模型依赖方，负责服务模型的调用和数据订阅。

物联网云平台：即IoT云端平台。在领域服务对接方案里，与边缘端核心服务一起提供服务总线和数据总线框架服务。

边缘托管应用：即运行在物业管理一体机的边缘应用，是服务模型的提供方。该应用一方面接受云端应用通过IoT平台的服务模型调用，然后将调用转换成本地系统支持的接口调用，另外一方面该应用处理本地的事件上报，然后通过边缘数据总线，将消息安装数据模型格式要求，上传到IoT云端平台

2.2 核心流程说明

基于边缘适配器应用的领域服务对接方案，核心流程包括两个：至上而下的服务模型调用，至下而上的数据上报。下面分别说明。下面的序号，与框架图中的序号一一对应。

服务调用：

(1) 云端SaaS应用调用服务模型提供的HTTP服务。 发起服务调用时，需提供项目Appkey, 路径名称(path)里需包含服务模型ID+接口方法名称。

(2) 边缘托管应用，侦听到对应的服务调用后，进行业务处理

数据上报：

(5) 边缘托管应用处理事务后，可以将事件内容转化成数据模型要求的格式，然后根据(边缘)IoT平台提供的数据插入接口，上报数据到边缘数据总线，然后内部流转到云端数据总线。

(6) 云端SaaS应用，可以通过IoT平台提供的查询数据模型的API接口获取数据，也可以通过数据变更消息订阅方式获取数据

3 开发指导

这里介绍的边缘应用，是服务提供方应用

3.1 对接服务模型

边缘应用侦听到对应的服务调用后，进行适当的适配转换，再调用本地系统提供的Http服务。

边缘应用对接服务模型，可以参考IoT公开链接：服务总线：服务提供的开发示例：

https://help.aliyun.com/document_detail/114863.html?spm=a2c4g.11186623.6.579.30243aa12KgeDk#h2-7-2-8

3.2 对接数据模型

具体对接可参考IoT公开链接：边缘应用数据总线对接：https://help.aliyun.com/document_detail/145523.html?spm=a2c4g.11186623.6.595.81bc5ee4wgidUt

这里只补充说明一些需要关注的内容：

通过本地系统环境变量获取appkey and appSecrect：

1. public static final String appkey =System.getenv("iot.hosting.appKey");
2. public static final String appSecret =System.getenv("iot.hosting.appSecret");
3. **边缘端数据模型服务路由**

private static final String DATA_EDGE_PATH ="api.link.aliyun.com";

请求参数：

请求参数里的modelId，就是对应的数据模型Id。

1. request.putParam("modelId","value1");

对于边缘端应用，下面两个请求参数可以忽略掉

1. request.putParam("scopeId","value2");
2. request.putParam("appId","value3");

上传文件数据模型接口说明：

下面链接文档提供的接口只是获得了需要上传的文件名称和URL:

https://help.aliyun.com/document_detail/145523.html?spm=a2c4g.11186623.6.595.56c2610adevJLp#h2-2-5-5

如返回接口示例：

```
{
"id":"6fr2c332-c1db-417c-aa15-8c5trg3r5d92",
"code":200,
"message":null,
"localizedMsg":null,
"data":{
"fileName":"5269712352e5.jpg",
"url":"https://xxxxx.xxx.xx.com/xxx/file/5269712352e5.jpg?Expires=1557902379&OSSAccessKeyId=uyedjYLHDH3DF&Signature=sotMFFIq4RP%2BWJSDScE8SxvOlvo%3D"
}
```

开发者还需将真正需要上传的文件，上传到接口返回值里指定的fileName 和 url，示例代码如下：

```
// response为上传文件数据模型接口的返回
result = new String(response.getBody(), "UTF-8");
UploadResult uploadResult = JSON.parseObject(result, UploadResult.class);
data nameAndPath = uploadResult.getData();
String url = nameAndPath.getUrl();
HttpClient httpClient = HttpClients.createDefault();
HttpResponse response;
HttpPut put = new HttpPut(url);
// byte[] fileBytes: 为准备要上传的图片文件
HttpEntity reqEntity = EntityBuilder.create().setBinary(fileByte).build();
put.setEntity(reqEntity);
response = httpClient.execute(put);
```

4 应用自测

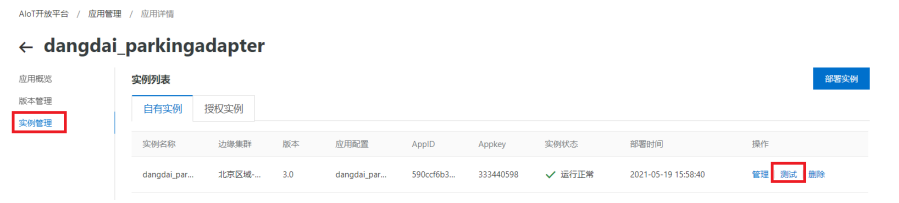
4.1 物联网应用服务平台应用调试接口

应用通过AIoT开放平台创建部署成功后，可以基于平台的应用调试接口，进行服务接口的调用验证：

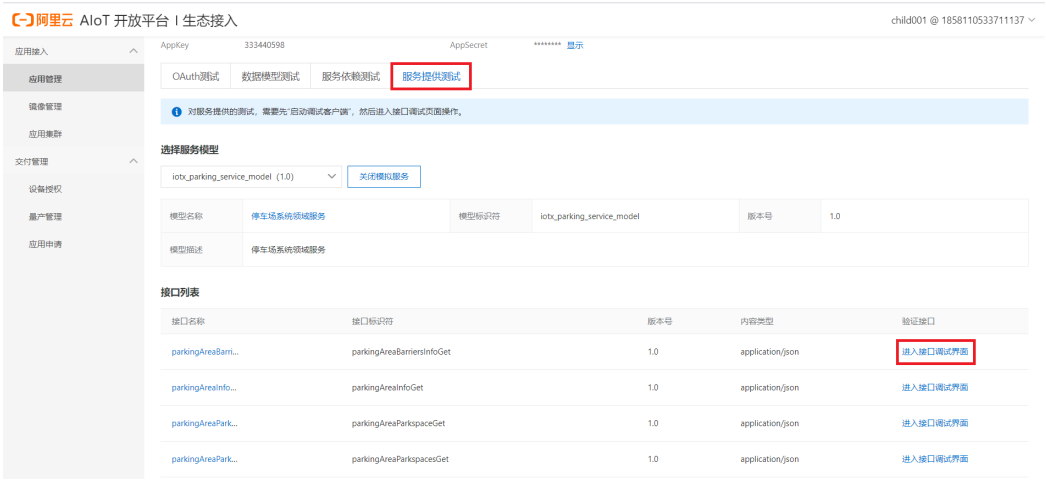
进入应用管理，点击需调试的应用。注意这个应用需要处于已发布状态，如下图所示



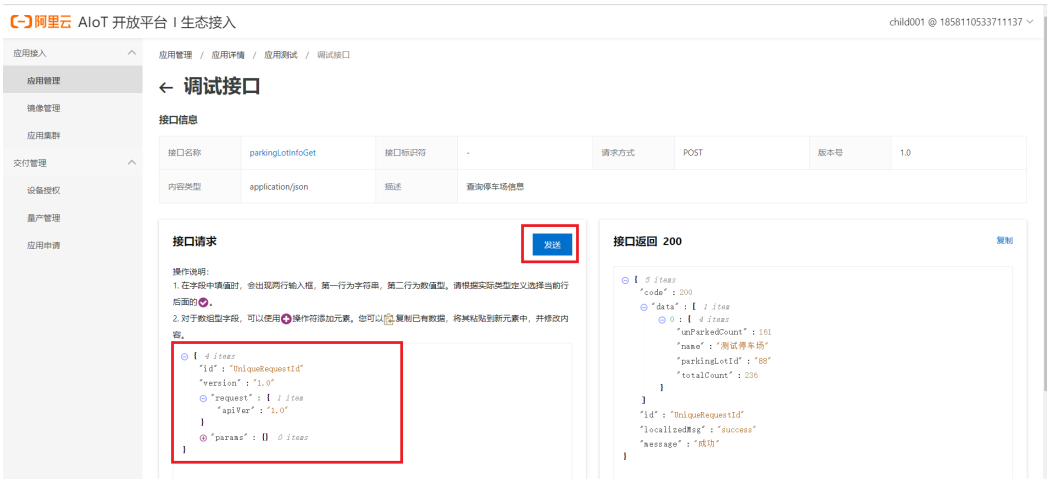
点击**实例管理-测试**：



点击**服务提供测试**，选择一个要调试验证的接口，点击右侧的**进入接口调试界面**



接口请求里，可以根据输入参数要求补充相关参数，然后点击**发送**，然后查看**接口返回**信息，如下图所示：



4.2 模拟本地系统事件上报：Postman

前置条件：

- PC机安装了Postman软件
- 物业管理一体机部署了适配器应用
- PC机需要与物业管理一体机在相同的局域网内

模拟事件上报：

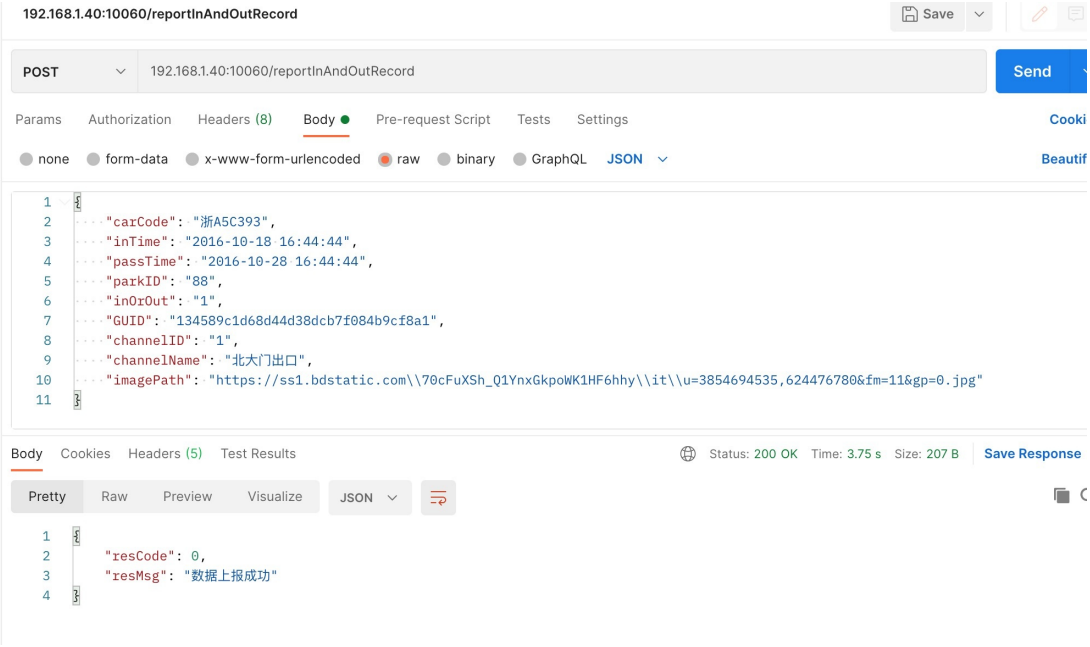
下面示例，以《停车场系统领域模型V3.1-数据模型定义》-车辆通行为例

路径：物业一体机ip:port/具体路径

请求Body内容示例如下：

```
{
    "carCode": "浙A5C393",
    "inTime": "2016-10-18 16:44:44",
    "passTime": "2016-10-28 16:44:44",
    "parkID": "88",
    "inOrOut": "1",
    "GUID": "134589c1d68d44d38dcb7f084b9cf8a1",
    "channelID": "1",
    "channelName": "北大门出口",
    "imagePath": "https://ss1.bdstatic.com\\70cFuXSh_Q1YnxGkpoWK1HF6hhy\\it\\u=3854694535,624476780&fm=11&gp=0.jpg"
}
```

Postman示例截图如下。其中192.168.1.40是物业一体机的内网IP地址，需根据实际情况填写，10060是固定的端口号。



4.3 模拟本地服务调用：Postman

前置条件：

- PC机安装了Postman软件
- 物业管理一体机部署了适配器应用
- PC机需要与物业管理一体机在相同的局域网内
- 本地部署了相应的系统，例如部署了立方停车系统

模拟服务调用：

下面示例，以《停车场系统领域服务V3.1-服务模型定义》-1.1 查询停车场信息为例

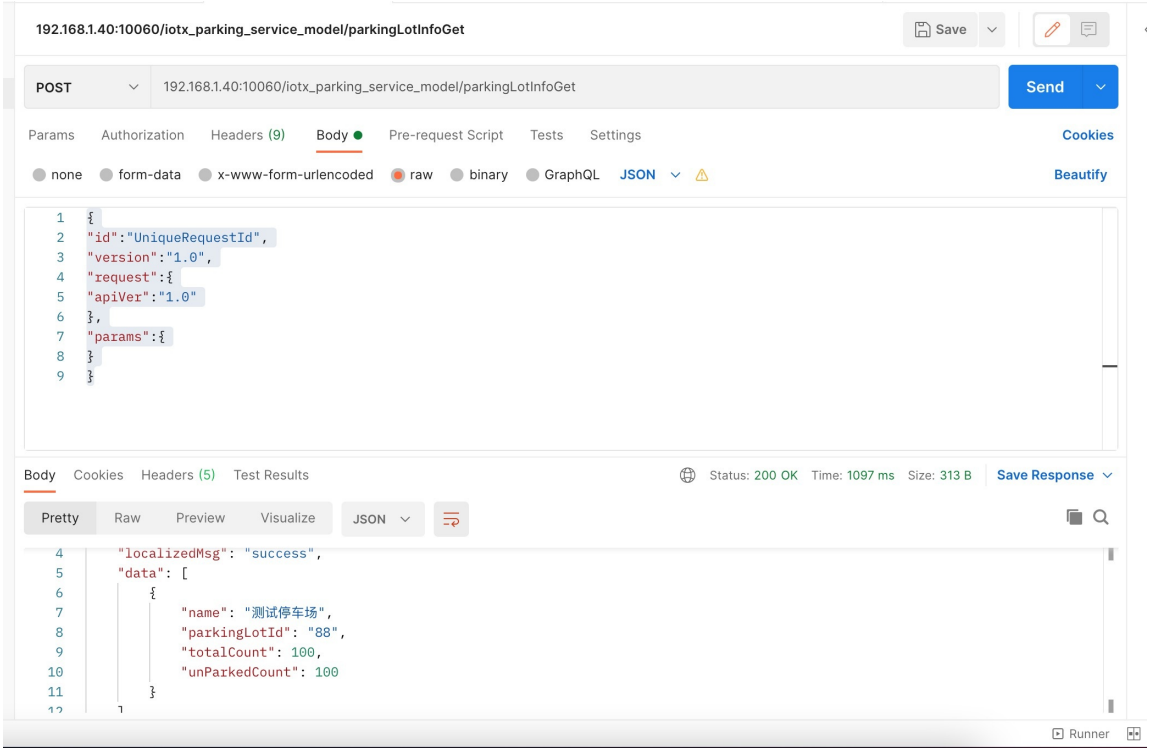
路径： 物业一体机IP:port/服务模型ID/服务接口path

特别注意Header内容：定义 Content-Type 为 application/octet-stream

请求Body内容示例如下：

```
{
 "id":"UniqueRequestId",
 "version":"1.0",
 "request":{
  "apiVer":"1.0"
  },
 "params":{
  }
}
```

Postman示例截图如下。其中192.168.1.40是物业一体机的内网IP地址，需根据实际情况填写，10060是固定的端口号。



4.4 如何查看边缘应用日志

应用开发完成并部署到物业一体机进行调试，可以登录到物业一体机，查看对应的应用日志：

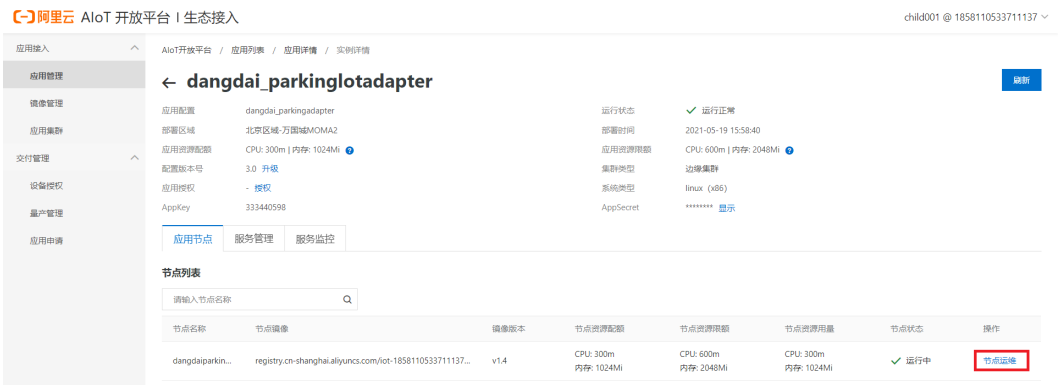
进入应用管理，点击需调试的适配器应用。注意这个应用需要处于已发布状态，如下图所示



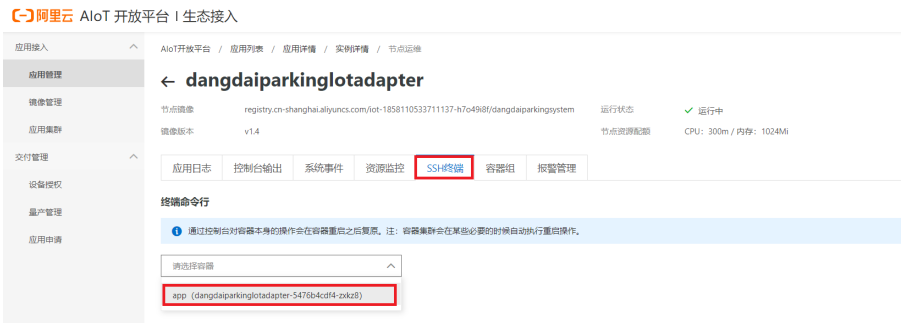
选择**实例管理-管理**



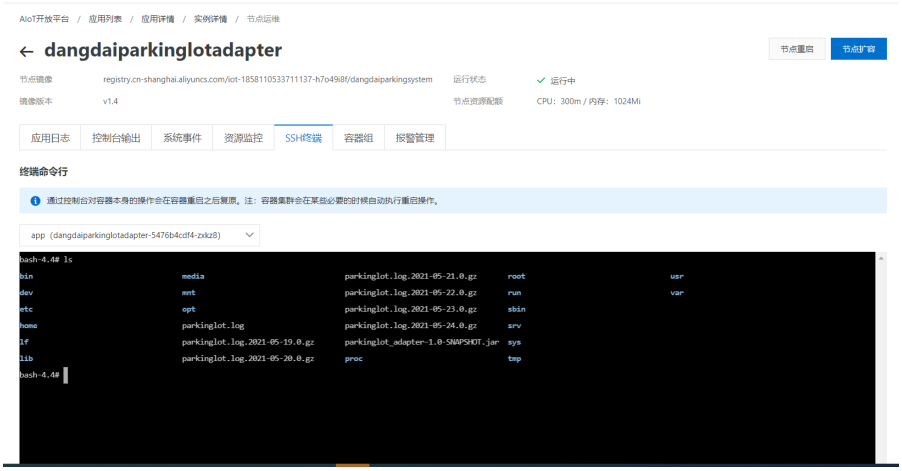
选择**节点运维**



点击SSH终端，选择容器（一般只有一个，点击选择即可）：



选择完容器，将自动登录到物业一体机。当前目录即保持有该应用的日志



5 示例代码

5.1 对接服务模型

以**车辆加入安全黑/白名单**服务为例

1、入口层

```
package com.aliyun.iotx.parkinglot.adapter.web.servicemodelcontroller;

import com.alibaba.fastjson.JSON;
import com.aliyun.iotx.common.base.service.IoTxResult;
import com.aliyun.iotx.parkinglot.adapter.enums.ParkingLotAdapterEnum;
import com.aliyun.iotx.parkinglot.adapter.service.BlackWhiteListService;
import com.aliyun.iotx.parkinglot.adapter.service.dto.VehiclePermissionalDTO;
import com.aliyun.iotx.parkinglot.adapter.service.serviceImpl.logicjudge.LogicJudge;
import com.aliyun.iotx.parkinglot.adapter.utils.IoTxResultUtils;
import com.aliyun.iotx.parkinglot.adapter.vo.*;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import javax.servlet.http.HttpServletRequest;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;

/**
 * 黑白名单
 */
@Slf4j
@RestController
@RequestMapping(value = "/iotx_parking_service_model", method = RequestMethod.POST)
public class BlackWhiteListController {

    @Autowired
    private BlackWhiteListService blackWhiteListService;

    /**
     * 7.1车辆加入安全黑/白名单
     * @param request
     * @return
     */
    @RequestMapping(value = "/parkingLotVehicleListAdd")
    public IoTxResult vehicleAddList(HttpServletRequest request) throws Exception {

        String json="";
        json= new String(readInputStream(request.getInputStream()),"UTF-8");

        //将json数据解析成vo对象接收
        BWListVo bwListVo = JSON.parseObject(json, BWListVo.class);
        BlackWhiteListVo blackWhiteList = bwListVo.getParams();

        //自己业务的处理逻辑
```

```
        blackWhiteListService.vehicleAddList(blackWhiteList);

        IoTxResult<Object> ioTxResult = new IoTxResult<>();
        ioTxResult.setData(null);
        IoTxResultUtils.ioTxResultSet(ParkingLotAdapterEnum.SUCCDESS_VEHICLEADDLIST, ioTxResult);
        return ioTxResult;
    }
}

/**
依赖类:
*/

@NoArgsConstructor
@AllArgsConstructor
@Data
class BWListVo implements Serializable {

    private static final long serialVersionUID = -8067179280515471493L;

    /**
     * request里的全局唯一id透传
     */
    private String id;

    /**
     * 请求协议版本
     */
    private String version;

    private Map<String, Object> request;

    private BlackWhiteListVo params;

}


/**
 * 车辆加入安全
 * 黑白名单入参
 */
@NoArgsConstructor
@AllArgsConstructor
@Data
@ValidateBean
class BlackWhiteListVo implements Serializable {

    private static final long serialVersionUID = 8317834756309144547L;

    @NotBlank(message = "parkingLotId不能为空")
    private String parkingLotId;

    @JsonFormat(with = ACCEPT_SINGLE_VALUE_AS_ARRAY)
    @EachValidate(constraint = NotBlank.class,message = "车牌号不能为空,不能有空字符串")
    private List<String> plateNumber;

    @JsonFormat(with = ACCEPT_SINGLE_VALUE_AS_ARRAY)
    private List<String> areaId;

    @NotBlank(message = "type不能为空")
    private String type;

    private String effectiveDate;
    private String expiryDate;
}
```

5.2 对接数据模型

以**车辆通行**数据模型为例:

车辆通行接口入口

```
package com.aliyun.iotx.parkinglot.adapter.web.datamodelcontroller;

import com.alibaba.cloudapi.sdk.model.ApiResponse;
import com.alibaba.fastjson.JSON;
import com.aliyun.iotx.parkinglot.adapter.service.datamodelservice.IoTParkPassRecordService;
import com.aliyun.iotx.parkinglot.adapter.utils.UrlFormatUtil;
import com.aliyun.iotx.parkinglot.adapter.vo.InOutRecordVo;
import com.aliyun.iotx.parkinglot.adapter.vo.LFResultVo;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

/**
 * 数据模型
 * 车辆通行
 */
@Slf4j
@RestController
public class IoTParkPassRecordController {

    @Autowired
    private IoTParkPassRecordService parkPassRecordService;

    /**
     * 上传进出记录
     * 当代项目的图片地址格式是
     * @param inOutRecordVo
     * @return
     */
    @RequestMapping(value = "/reportInAndOutRecord", method = RequestMethod.POST)
    public LFResultVo inOutRecordReport(@RequestBody InOutRecordVo inOutRecordVo) {

        String newUrl = UrlFormatUtil.change(inOutRecordVo.getImagePath());
        inOutRecordVo.setImagePath(newUrl);

        //把记录转换为标准数据模型后上传，需要引用后一段通用的DOP上传代码片段。参考下面的边缘应用数据总线参考使用：
        ApiResponse apiResponse = parkPassRecordService.inOutRecordReport(inOutRecordVo);

        LFResultVo lfResultVo = new LFResultVo();
        if (apiResponse.getCode() == 200) {
            lfResultVo.setResCode(0);
            lfResultVo.setResMsg("数据上报成功");
        } else {
            //立方上报数据失败返回数字1
            lfResultVo.setResCode(1);
            lfResultVo.setResMsg("数据上报失败");
        }
        return lfResultVo;
    }
}
```

边缘应用数据总线参考使用：

```
public class BlackWhiteList {
    /**
     * 车辆通行上传
     * @param request
     * @return
     */
    public static void main(String[] args) {
        IoTApiRequest ioTApiRequest = new IoTApiRequest();
        String uuid = UUID.randomUUID().toString();
        String uuidOne = uuid.replace("-", "");

        ioTApiRequest.setId(uuidOne);
        ioTApiRequest.setApiVer("1.0");
        ioTApiRequest.putParam("modelId", "iot_park_pass_record");

        JSONObject properties = new JSONObject();
        properties.put("direction", "获取入参的对应的值");
        properties.put("openType", "获取入参的对应的值");
        properties.put("plateNumber","获取入参的对应的值");

        //首先获取到要上传的文件名和上传的路径
```

```
        ApiResponse response = getUploadFileNameAndPath();
        String result = null;
        try {
            result = new String(response.getBody(), "UTF-8");
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }

        //将JSON字符串转化成对象
        UploadResult uploadResult = JSON.parseObject(result, UploadResult.class);
        data nameAndPath = uploadResult.getData();

        properties.put("plateNumberImage", nameAndPath.getFileName());

        //根据底层停车系统上报的文件路径下载文件
        String lfFileUrl = inOutRecordVo.getImagePath();
        //下载后文件的名字
        String name = nameAndPath.getFileName();
        //文件保存路径
        String savePath = fsp.getSavePathOne();

        if(!StringUtil.isEmpty(lfFileUrl)){
            String url = FileUtil.getURL(lfFileUrl);
            FileUtil.downLoadFromUrl(url,name,savePath);
        }

        //根据dop接口获取到的path将文件上传
        //上传文件的路径
        String url = nameAndPath.getUrl();
        //要上传的文件
        File file = new File(savePath + name);
        //判断上传是否成功
        //首先判断文件是否存在,存在才上传文件
        if(file.exists()){
            byte[] fileBytes = FileUtil.getFileBytes(file);
            boolean b1 = FileUtil.uploadFile(url, fileBytes);
            if (!b1) {
                log.info("文件上传失败");
                throw new ParkingLotAdapterException(502, "文件上传失败", "File upload failed");
            }
        }
        //文件上传成功后,清除本地缓存的文件
        FileUtil.deleteFile(savePath, name);

        properties.put("typePermission", "获取入参的对应的值");
        properties.put("plateColor", "获取入参的对应的值");
        properties.put("plateType", "获取入参的对应的值");
        properties.put("vehicleColor", "获取入参的对应的值");
        properties.put("vehicleType", "获取入参的对应的值");
        properties.put("barrierId", "获取入参的对应的值");

        if (StringUtils.isNotBlank(inOutRecordVo.getChannelName())) {
            properties.put("barrierName", "获取入参的对应的值");
        }

        properties.put("parkingLotId","获取入参的对应的值");
        properties.put("areaId", "未知");
        properties.put("orderNumber", "获取入参的对应的值");
        properties.put("recordId", "未知");

        //数据上报的时间
        Date date = new Date();
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        String eventTime = dateFormat.format(date);
        properties.put("eventTime",eventTime);  //应该产生一个时间上报

        ioTApiRequest.putParam("properties", properties);
        ApiResponse apiResponse = syncApiClient.postBody(host, path, ioTApiRequest, "https".equalsIgnoreCase(schema));

    }

    public static ApiResponse getUploadFileNameAndPath() {

        IoTApiRequest request = new IoTApiRequest();

        //设置api的版本
        request.setApiVer("0.0.1");

        // 接口参数
```

```
        String uuid = UUID.randomUUID().toString();
        request.setId(uuid.replace("-", ""));

        JSONObject param = new JSONObject();
        param.put("appId",  "应用id");
        param.put("version", "应用版本");

        request.putParam("properties", param);

        //这个参数对应于数据模型中需要上传文件的字段
        request.putParam("attrName", "plateNumberImage");

        //这个参数对应于数据模型的模型id
        request.putParam("modelId", "iot_park_pass_record");

        request.putParam("fileType", "文件类型");
        request.putParam("version", "版本");
        request.putParam("fileSize", "文件大小");

        try {
            ApiResponse apiResponse = syncApiClient.postBody(DATA_EDGE_PATH,
                    "/data/model/data/upload", request, false);
            return apiResponse;
        } catch (IOException e) {
            log.info("上传文件获取文件名和上传路径接口出现异常:{}", e.getMessage());
        }
        return null;
    }

}
```

pom依赖：

```
        <dependency>
            <groupId>com.aliyun.iotx</groupId>
            <artifactId>iotx-api-gateway-client</artifactId>
        </dependency>

        <dependency>
            <groupId>com.aliyun.iotx</groupId>
            <artifactId>common-base</artifactId>
        </dependency>
```

3.4.7.2. 边缘应用发布

修改记录及版本信息

序号	版本号	修改内容	修改时间
1	Rev1.0	文档创建	2022-01-21

1. 整体流程

一个“边缘托管应用”的部署整体流程，包括：应用开发、创建应用、应用配置、发布版本、部署实例。

2. 应用开发

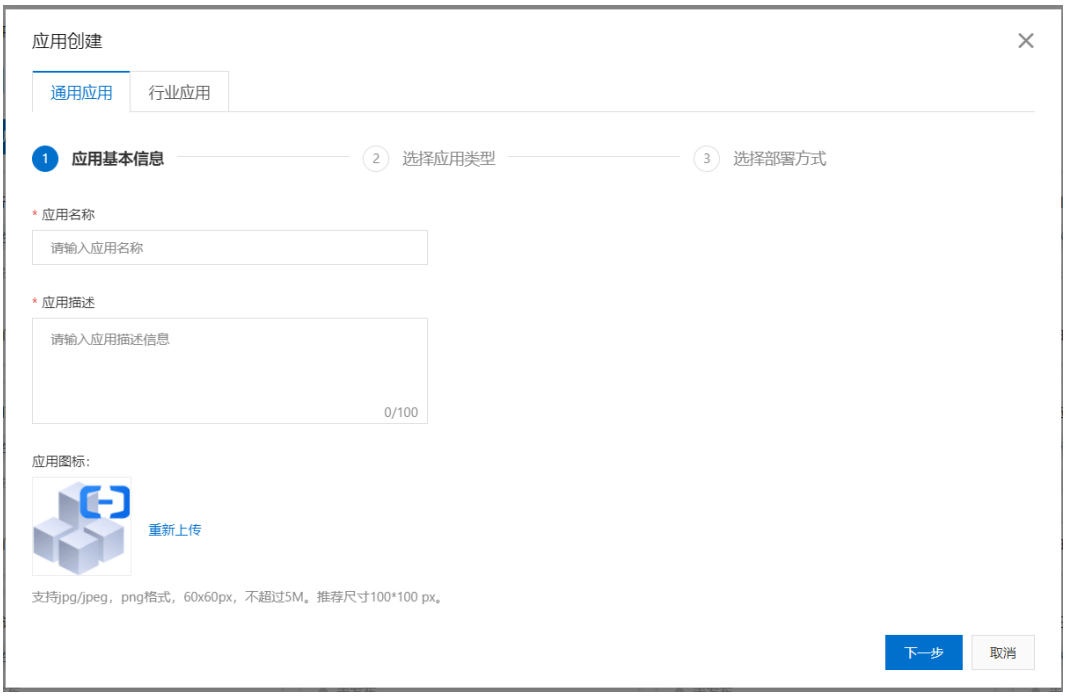
应用功能开发

3. 创建应用

登录AIoT开放平台， 在 **应用接入 > 应用管理 > 创建应用**



填写应用基本信息，如图所示：



选择应用类型为**实例分发**，如图所示：



根据应用实际情况选择部署方式与系统类型，如图所示：



- 部署方式：

云端托管部署：平台根据应用提供者的配置，自动分配资源，并部署应用到云端。边缘托管部署：平台根据应用提供者的配置，自动分配资源，并部署应用到边缘。

4. 应用配置

创建应用完成后点击刚才新创建的应用，**版本管理 > 初始化应用**，填写版本说明，如下图所示



4.1 可视化编排

版本管理 > 可视化编排，进入可视化编排



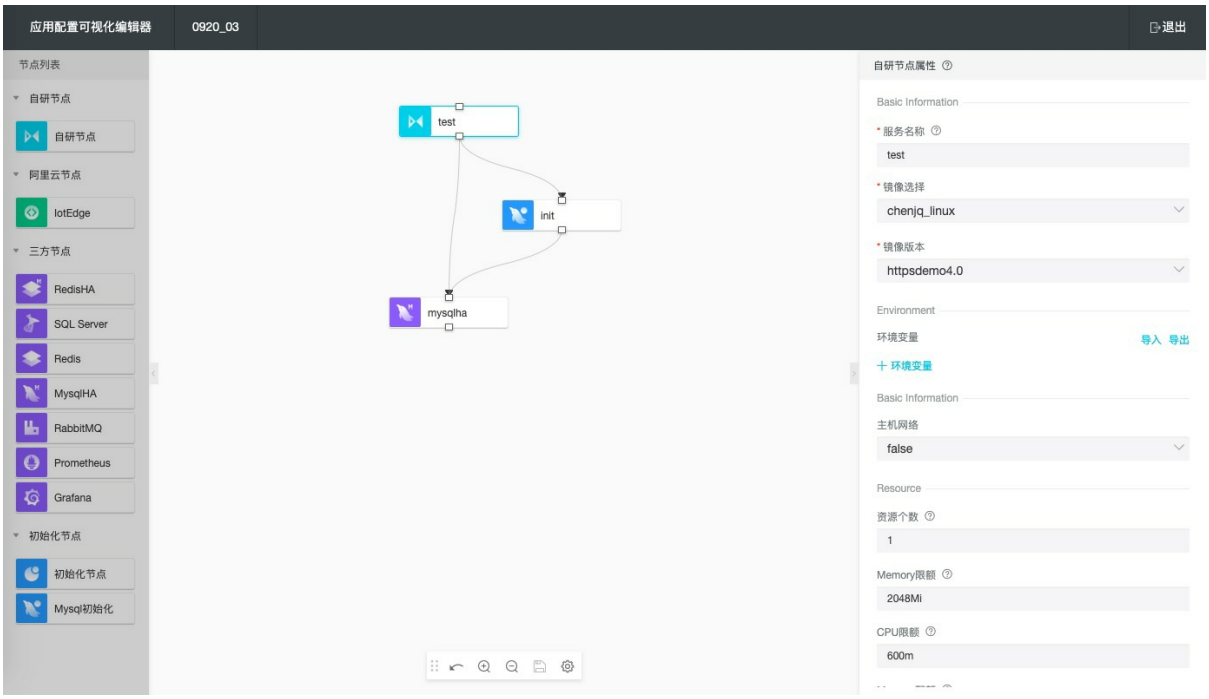
基本介绍：一般情况下，一个应用是由多个节点构成的。这里的节点是指独立运行的应用进程。比如一个常规业务管理系统的应用，常常是一个Tomcat节点，再加一个数据库MySQL，这种情况下，这个应用就是由两个节点构成，这里的Tomcat节点，在可视化编排页面表现为一个“自研节点”；数据库节点，则是一个MySQL中间件。对应前者，用户需要打包成以一个docker容器镜像，在自研节点的右侧配置项中选择上传的镜像及版本；后者，用户可以直接拖动左侧现成的中间件节点，用户无需自行打包。

页面介绍：应用配置可视化编辑页面分为三部分内容：

- 左侧部分为节点：显示已支持的各类节点，所有节点配置请查看节点说明
- 中间部分为画布：显示应用需要的节点及部署顺序关系。注意⚠️，节点间的部署顺序关系，是通过连线来表示的，连线箭头所指节点先部署，连线起点后部署。
- 右侧部分为节点属性：显示某个节点可以配置的参数。

左侧的节点列表，分成了四类：

- 自研节点：用户自己上传的镜像的载体。一份配置，可以有多个自研节点。
- 阿里云节点：一个节点，在应用被部署之后，对应阿里云RDS产品的一个实例（如一个RDS for MySQL数据库实例）。
- 三方节点：各类主流开源中间件Docker镜像。这一类镜像完全来自于第三方，功能、性能、质量、安全等，均保持不变。
- 初始化节点：这类节点属于辅助节点，在整个部署的生命周期，只会执行一次（其他节点通常不止一次，比如节点健康检查失败就会重启）。点击 **版本管理 > 可视化编排** ，可视化编排是对一个应用的定义。包括一份应用中的节点构成、节点的配置、启动顺序等，如图所示：



对于自研节点的配置项目：

- **服务名称**：节点之间的访问地址，并不是传统的IP，而是用”**服务名“替代**。用户不需要每次部署时去配置不同节点的访问IP。
- **CPU/Memory配额**：节点启动所期望的最小资源需求。如果在部署中发现主机资源不满足该条件，将导致节点无法启动。
- **CPU/Memory限额**：节点运行过程中所占用的资源的上限。防止运行在同一主机内的其他应用的运行资源被无限制的抢占。
- **资源个数**：该节点启动的副本数量。通过设置多个副本，可以充分利用多个主机的资源，系统通过内部网络实现自动的负载均衡。需要注意的是多个副本之间的状态是无法自动同步的，因此建议开发阶段注意节点无状态设计。
- **环境变量**：这是一个键值对，节点启动的时候，系统会将这里设定的键值对，放在环境变量中，供系统获取。
- **应用日志目录**：该目录是用来指定应用程序自身存储日志的路径。节点部署时，会自动注入一个日志采集插件，该插件将会根据该路径获取日志，并上传到云端的日志系统中，用户可以进行日志分析。
- **自研节点端口对**：该节点开放的内部端口和外部系统访问的端口之间的映射关系。
- **主机网络**：启用主机网络，则该外部系统可以直接使用应用的 **“当前主机地址”** + **“端口对的外部端口”** 进行访问。
- **服务类型**：

ClusterIP：这种模式，节点开放的端口仅能被集群内其他节点访问，外部无法访问；访问时使用服务名称

+端口对配的外部端口号进行访问。

Loadbalance：会从创建集群时提供的服务地址段中分配一个IP绑定到该服务，可以在**集群外可通局域网络**

访问；访问端口使用**“端口对的外部端口”**。

NodePort：外部系统通过集群里面“任何一个主机的物理IP”+“系统随机分配的外部端口”访问该节点

端口的内部端口对，随机分配的端口可在控制台应用详情页节点列表查询获得。该端口一般会分配30000以

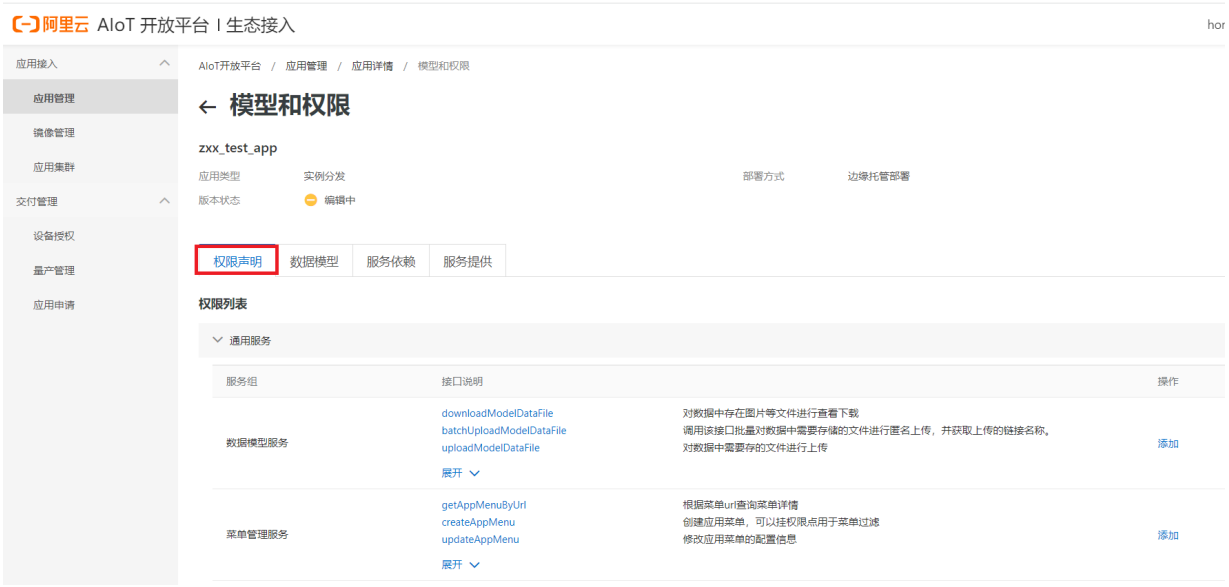
上。

- **主机文件映射**：需要输入的信息有：容器内路径、主机路径、路径类型（普通文件、字符设备、文件夹、套接字、块设备）。容器路径是用户应用代码的路径，主机路径是物理主机的路径。该配置项用于指定应用读写的文件与实际物理文件之间的映射关系。
- **存储目录**：需要输入的信息有：标识名称、容器内路径。容器路径是用户应用代码的路径。该配置项用于指定NAS存储的相对路径。不涉及NAS存储的，不需要配置。

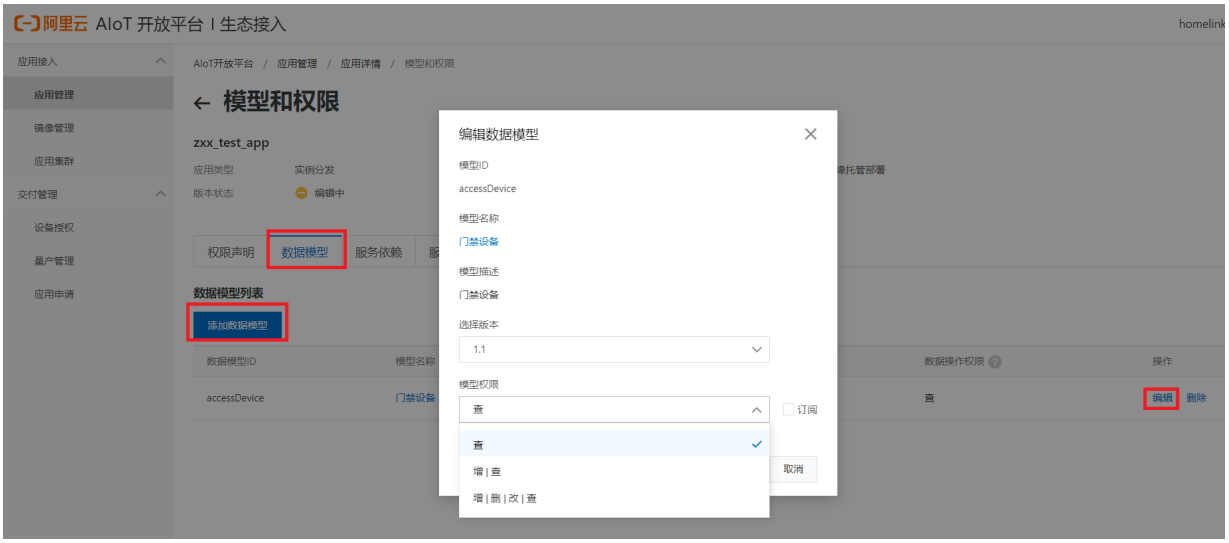
4.2 模型与权限

权限声明

点击**版本管理>模型与权限>权限声明**，在应用的分发模式，每一次分发，都有可能归属到不同的买家。因此，每一个应用实例，都需要一个唯一的值来代表应用实例的身份。这里，我们采用了AppKey + AppSecret的模式。随之而来的是，我们如何给AppKey授权。应用在调用IoT的各种API的时候，拿的这个AppKey需要被事先授权。因此，应用上架时，有必要清晰的指出该应用会调用哪些API，这样才能在AppKey产生的那一刻，给他授相应的权限，在此页面可对实例进行授权等操作，如图所示：

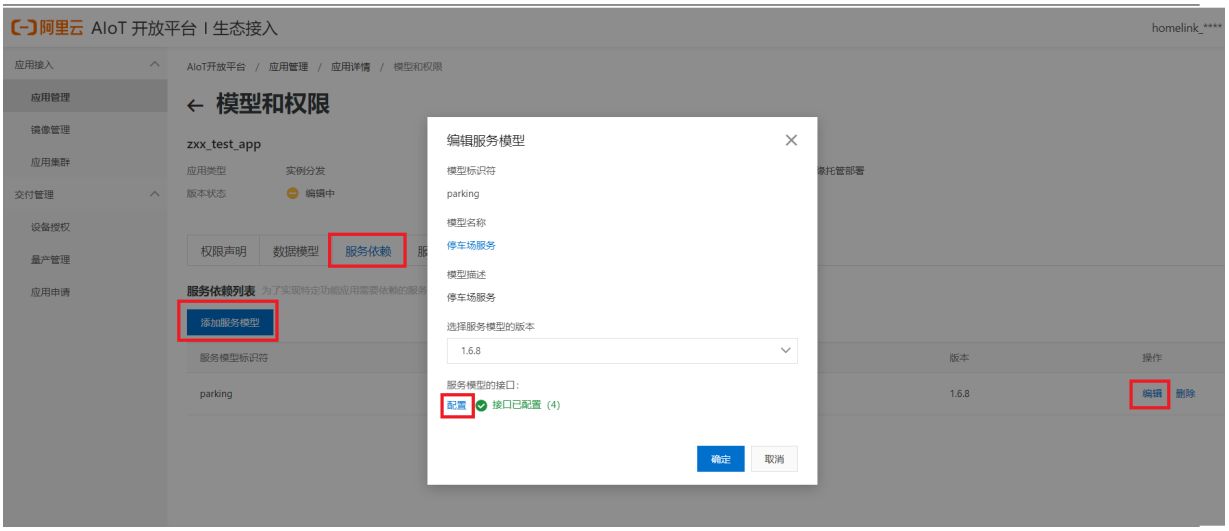


注意⚠ 若需要对接数据模型服务需要添加**数据模型权限**，对接单租户免登，需要开通**oauth协议服务权限**。数据模型点击**数据模型>添加数据模型**，在列表中，除了选择要声明的模型之外，还要选择版本、数据权限、订阅。其中，数据权限有三种：查、增|查、增|删|改|查，分别对应不同的操作类型。如图所示：



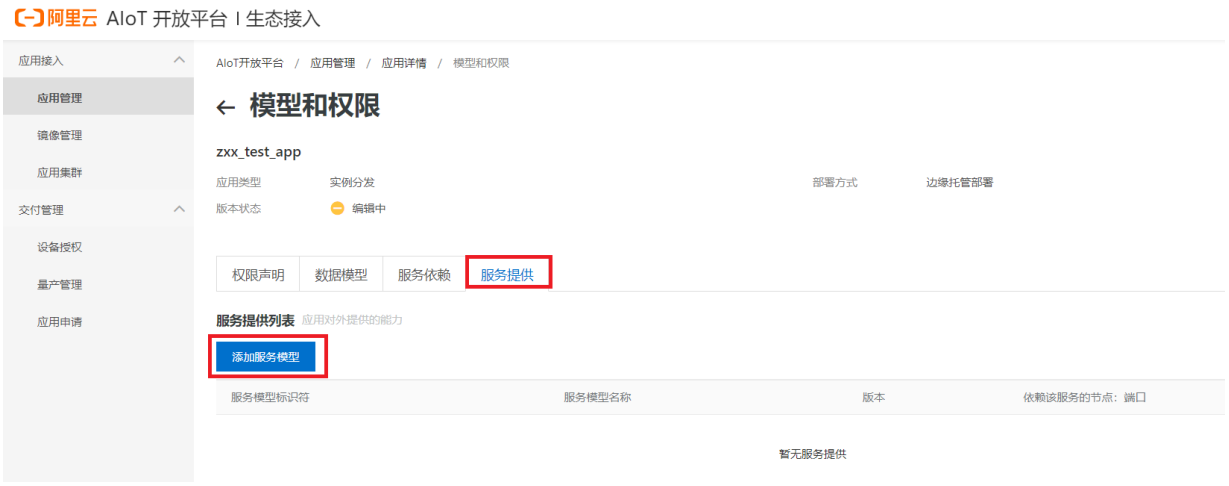
服务依赖

点击**服务依赖>添加服务模型**（应用为服务依赖方添加，服务提供方无需添加），选择对应的服务模型，并配置服务模型的API，如图所示：



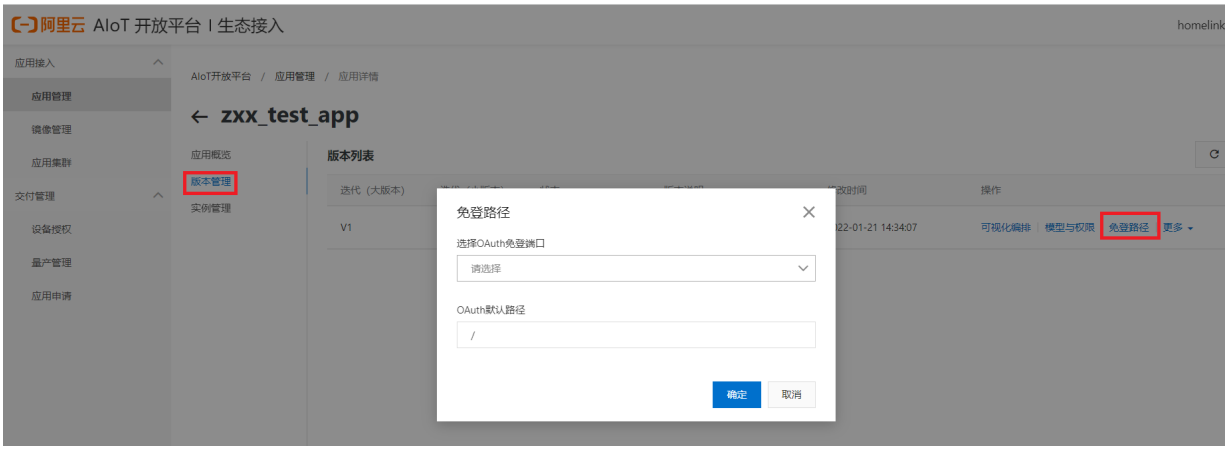
服务提供

点击**服务提供>添加服务模型**（应用为服务提供方添加，服务依赖方无需添加），选择对应的服务模型，并选择相应的端口，如图所示：



4.3 免登路径

如果需要在边缘控制台中查看已部署的应用信息，则需要配置**免登路径**，否则边缘控制台的免登跳转会无法识别跳转路径，如图所示：



5. 发布版本

您需要先**发布版本**，版本发布后部署的应用才可以走升级流程，如图所示：



6. 部署实例

点击实例管理>部署实例



按照界面提示填写参数，如图所示：



参数设置如下：

参数	描述
应用名称	部署后应用实例的名称。该名称需账号内唯一。
应用配置	选择应用配置界面中创建的配置。

部署区域	当应用配置为边缘配置时，显示您在集群管理中创建的边缘集群。
资源信息状态	展示您选择的配置所有的节点及占有的资源情况

- 单击部署，执行部署任务。
- 部署应用时，需要根据应用配置占用相应的云资源。
- 部署过程可能会花费较长时间，请耐心等待。
- 应用部署成功后，您可以在部署记录页进行查看、运维和删除等操作，详情请参考运维工具文档。

3.4.7.3. 边缘应用运维

修改记录及版本信息

序号	版本号	修改内容	修改时间
1	Rev1.0	文档创建	2022-01-21

应用运维功能包括：监控、升级等。如需对应用进行升级，您需要**使用发布成功的版本进行部署**，才可以走后续的应用升级流程。

应用升级

登录AIoT开放平台， 在应用管理里找到相应应用后，点击**版本管理 >应用升级**，如图所示：



点击**可视化编辑**，可对镜像版本，接口访问路径等信息进行修改，如图所示：



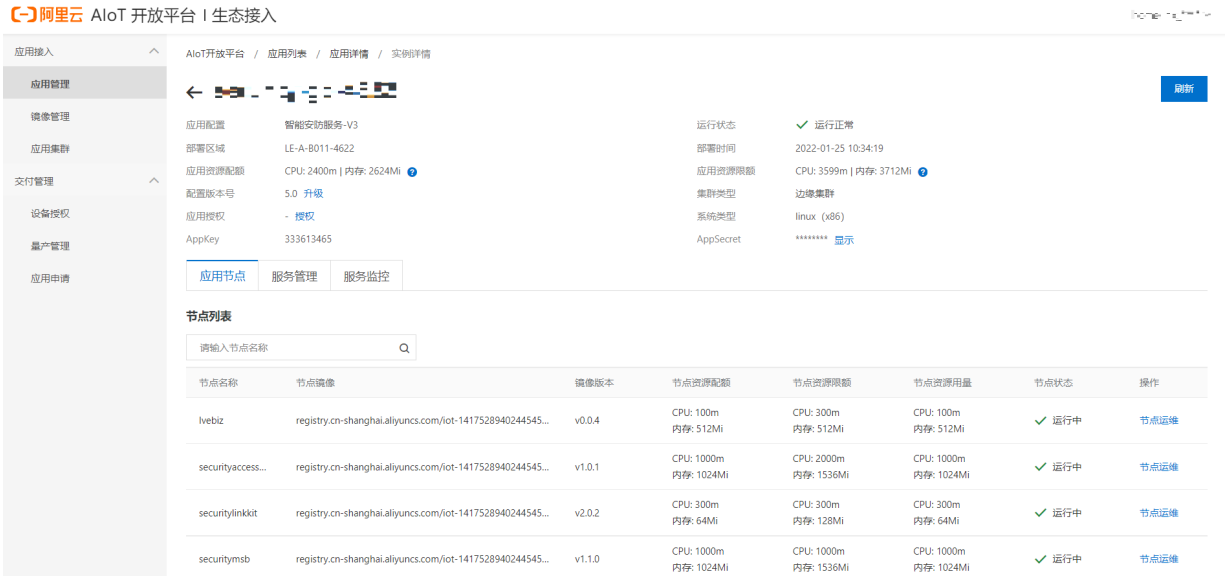
镜像版本更新成功后，点击**版本管理>发布版本**，即可启动应用升级。



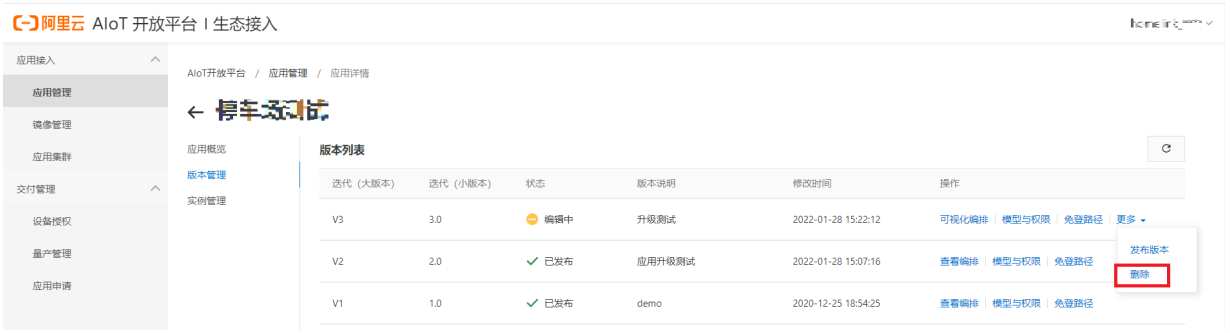
完成新版本的实例部署之后，进入应用，确认应用运行情况，如图所示：



应用详情页面如下：



对于正在编辑中的版本，若取消升级可点击**删除**按钮，如下图图所示：

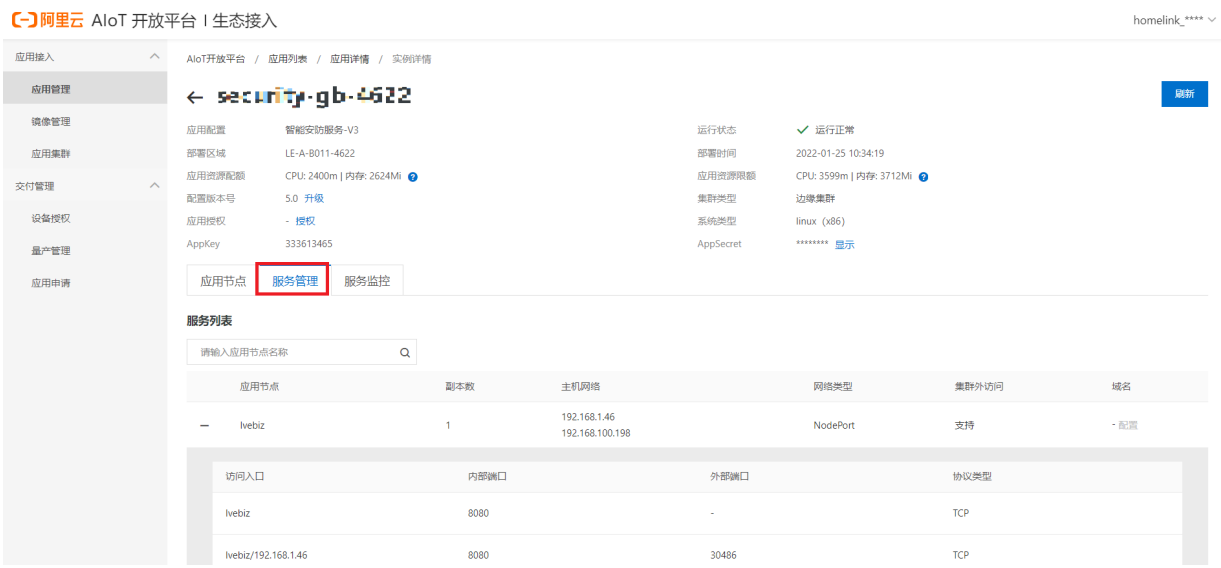


服务管理

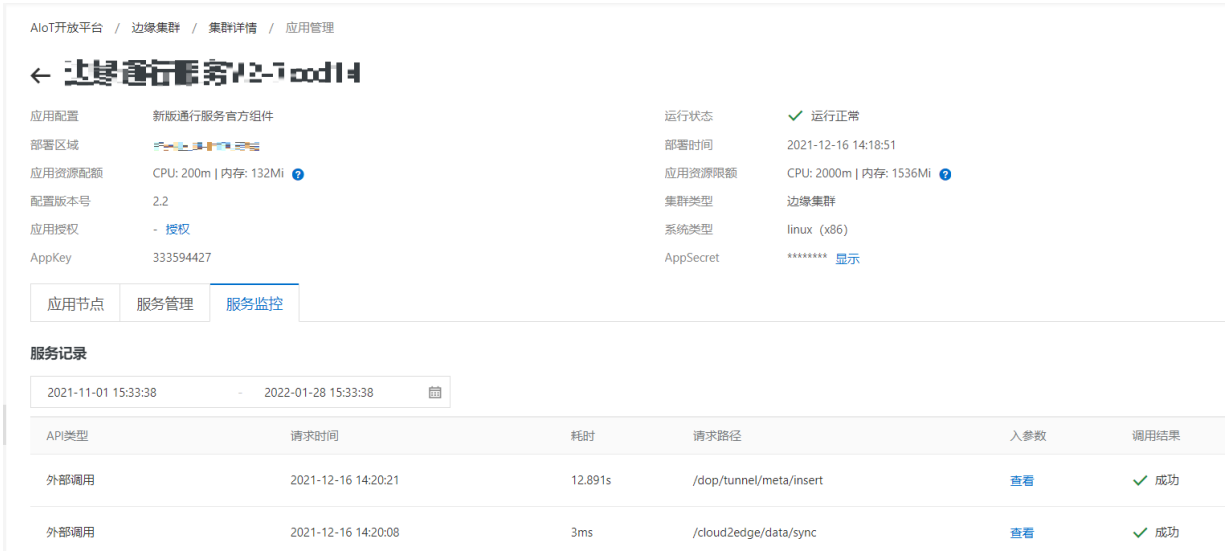
点击实例管理 > 管理，可以进入应用详情页面：



点击服务管理，可以查看应用节点的端口信息和协议信息，如下图所示：



点击服务监控，可以查看服务记录，包括请求路径和请求时间，如下图所示：



3.5. 边缘接入

3.5.1. 人行标准

3.5.1.1. 1 人行方案选型标准

1 编写目的

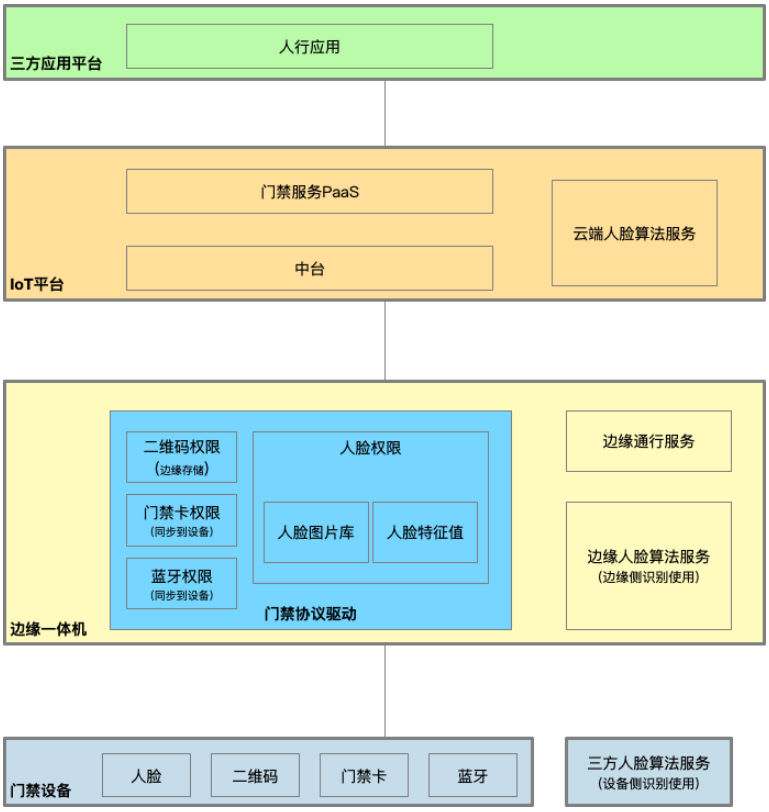
文档主要是介绍人行对接的整体方案选择，同时介绍了人脸、刷卡、二维码、蓝牙四种门禁对接方案和可视对讲对接方案的技术特点，这样设备厂商可以方便的进行对接方案选择。

2 方案说明

人行设备接入方案，从对接协议上分为“门禁一体机协议”和“自定义协议”。

门禁一体机协议：基于门禁一体机上已有的门禁协议驱动（下图蓝色部分），定义了驱动与门禁设备间的标准通信接口。门禁设备需要根据该标准接口进行对接开发。例如可视对讲一体机设备可选择该接入方案。

自定义协议：需要设备厂商或ISV开发自定义驱动程序，驱动与设备之间采用私有协议通信。例如刷卡门禁控制器可选择该接入方案。



3 门禁一体机协议对接

门禁一体机协议支持人脸、刷卡、二维码、蓝牙、可视对讲这五个功能。

人脸：人脸比对方式，分为“边缘服务器端比对”和“设备端比对”。

刷卡：包括“一体式门禁”对接和“门禁服务器”对接：

一体式门禁：是读卡器和控制器集成一体的刷卡门禁设备。

门禁服务器：是代理门禁控制器接入，做为门禁一体机驱动与门禁控制器之间的中介。

二维码：门禁设备端进行二维码识别，边缘服务器端进行二维码比对。

蓝牙：门禁蓝牙模块按照标准协议开发，实现与手机的身份认证。

可视对讲：门禁设备使用边缘服务器提供的SIP账号密码与SIP服务器通信。

3.1 人脸

人脸比对分边缘服务器端比对和门禁设备端比对：

边缘服务器端人脸比对

优势：

i. 人脸照片等敏感数据集中管理，统一安全管控。
ii. 人脸算法更新迭代方便，质量持续改进。
iii. 免去设备集成人脸算法成本。
iv. 不依赖特定设备和特定人脸识别算法。

门禁设备端人脸比对

优势：

i. 设备自身集成人脸算法，可在离网环境下进行人脸比对。

下面分别介绍这两种比对方案：

3.1.1 边缘服务器端比对

方案要求

1. 需要保障边缘服务器与门禁设备间的网络畅通，且低延时。
2. 需要保障边缘服务器的电源供电稳定。

方案流程

1. 云端应用添加人脸，IoT平台校验人脸照片质量
2. 云端应用配置人脸权限，并同步到边缘服务器

3. 边缘服务器计算人脸特征值，并存储在边缘服务器上
4. 人员刷脸通行，门禁机抓拍人脸照片
5. 门禁机发送人脸照片给边缘服务器认证
6. 认证通过后，门禁机开门

开发接口

1. 人脸识别

请求方向：设备 -> 边缘

2. 获取人脸库照片

请求方向：设备 -> 边缘

3.1.2 门禁设备端比对

方案要求

1. 设备使用的人脸算法模型需要与边缘服务器保持一致。
2. 设备需要具备人脸特征值存储和比对的能力。

方案流程

1. 云端应用添加人脸，IoT平台校验人脸照片质量
2. 云端应用配置人脸权限，并同步到边缘服务器
3. 边缘服务器计算人脸特征值，并同步到门禁机上
4. 人员刷脸通行，门禁机抓拍人脸照片
5. 门禁机认证人脸照片
6. 认证通过后，门禁机开门

驱动接口

1. 同步人脸信息

请求方向：边缘 -> 设备

2. 查询设备上人脸库信息

请求方向：边缘 -> 设备

3. 人脸通行事件

请求方向：设备 -> 边缘

4. 获取人脸库照片

请求方向：设备 -> 边缘

3.2 刷卡

分为“一体式门禁”对接和“门禁服务器”对接两种方式。

门禁服务器的对接方案详见《2.2门禁服务器对接方案介绍》

“一体式门禁”方案要求

门禁设备需要具备门禁卡的存储管理能力。

“门禁服务器”方案要求

门禁服务器或门禁控制器具备门禁卡的存储管理能力。

方案流程

(1) 云端应用添加门禁卡信息

(2) 云端应用配置门禁卡权限，并同步到边缘服务器

(3) 边缘服务器同步门禁卡信息到门禁机（或门禁服务器，再由服务器同步到门禁控制器）

(4) 人员刷卡通行

(5) 门禁机认证门禁卡信息（或由门禁控制器认证）

(6) 认证通过后，门禁机开门

驱动接口

(1) 同步卡权限

请求方向：边缘->设备

(2) 刷卡通行事件

请求方向：设备 -> 边缘

(3) 查询设备上的卡权限信息

请求方向：边缘->设备

(4) 上报设备上的卡权限信息

请求方向：设备 -> 边缘

3.3 二维码

方案要求

(1) IoT门禁服务生成二维码接口返回32位随机字符串。如果采用标准生成方式，需要门禁设备支持识别32位字符串长度的二维码（包括数字、大小写字母），否则需要采用自定义生成二维码的方式。

方案流程

(1) 手机APP申请二维码，云端生成二维码，返回给APP显示

(2) 手机APP分配二维码权限，并同步到边缘服务器

(3) 手机APP扫码，门禁机识别二维码

(4) 门禁机发送二维码给边缘服务器认证

(5) 认证通过后，门禁机开门

驱动接口

(1) 二维码识别

请求方向：设备->边缘

3.4 蓝牙门禁方案

方案要求

(1) 门禁设备存储边缘服务器下发的蓝牙卡号。

(2) 门禁设备与手机按照标准协议对接，详见《2.4 附件：门禁设备与手机蓝牙对接协议》。

方案流程

(1) 云端应用添加蓝牙卡信息

(2) 云端应用配置蓝牙卡权限，并同步到边缘服务器

(3) 边缘服务器同步蓝牙卡信息到门禁机

(4) 手机APP连接门禁机蓝牙

(5) 蓝牙连接成功后，发送加密卡号

(6) 认证通过后，门禁机开门

驱动接口

(1) 同步蓝牙权限

请求方向：边缘->设备

(2) 蓝牙通行事件

请求方向：设备 -> 边缘

3.5 可视对讲方案

方案要求

(1) 室内机15秒内无响应时，自动切换为云对讲呼叫。

方案流程

初始化

(1) 云端应用为住户向SIP服务器申请SIP账号，住户手机APP用此SIP账号与SIP服务器建立连接

(2) 云端应用为门禁机向SIP服务器申请SIP账号，并通过到边缘服务器同步到门禁机，门禁机用此SIP账号与SIP服务器建立连接

(3) 云端应用向SIP服务器申请SIP组，将房号与SIP组进行一对一关联，并将映射关系同步到边缘服务器

(4) 云端应用根据住户与房号的关系，将住户的SIP号分配到对应的SIP组

门禁对讲呼叫手机APP

(1) 访客在门禁机上输入房号，门禁机使用房号向边缘服务器查询SIP组号，再使用SIP组号向SIP服务器发起呼叫

(2) SIP服务器转发呼叫请求到云端应用，云端应用通知SIP组下的所有SIP账号

(3) 手机APP接入SIP服务器，与门禁机建立可视对讲通道，进行视频通话

(4) 手机APP发送开锁指令，门禁机执行开门

手机APP呼叫门禁对讲

(1) 手机APP向云端应用查询指定门禁的SIP号

(2) 手机APP使用门禁的SIP号向SIP服务器发起呼叫

(3) 手机APP与门禁机建立可视对讲通道，进行视频通话

(4) 手机APP发送开锁指令，门禁机执行开门

门禁对讲语音呼叫手机

(1) 访客在门禁机上输入手机号，门禁机向边缘服务器校验手机号，门禁机异步等待

(2) 边缘服务器向云端查询手机号有效性，返回结果转发到门禁机

(3) 门禁机通过本地语音网关呼叫住户手机

(4) 住户接听后，进行语音通话

(5) 住户点击*号键，门禁机执行开门

驱动接口

(1) 颁发门禁机SIP号

请求方向：边缘->设备

(2) 获取SIP组号

请求方向：设备 -> 边缘

(3) 校验手机号

请求方向：设备 -> 边缘

(4) 手机号校验结果

请求方向：边缘->设备

4 自定义协议对接

自定义协议一般使用设备厂商的私有协议，依据标准物模型来开发基于Link IoT Edge环境的驱动程序，实现设备接入。

请参考 设备接入简介

门禁控制器的对接方案内容详见《3.1 门禁控制器对接方案介绍》

5 文档说明

详细的对接方案说明和协议说明及接口定义，请参见下面文档

5.1 门禁协议对接

方案介绍：

门禁对讲一体机对接：《2.1 门禁对讲一体机对接方案介绍》

5.1.1 门禁（人脸、二维码、门禁卡、蓝牙）

对接接口定义：《2.2 门禁一体机标准协议接口定义》

蓝牙通信协议：《2.3 门禁设备与手机蓝牙对接协议》

5.1.2 可视对讲（云对讲、语音对讲）

设备端Android SDK：《2.4 可视对讲设备端SDK接口说明[Andriod]》

SIP接入协议：《2.5 可视对讲SIP服务器接入协议》

5.2 自定义协议对接

方案介绍：《3.1 门禁控制器对接方案介绍》

物模型：《3.3 人行设备物模型》

3.5.1.2. 2 门禁一体机协议对接方案

3.5.1.2.1. 2.1 门禁对讲一体机对接方案介绍

1 编写目的

用于指导人行设备厂家的技术人员评估和开发工作，从而实现人行设备接入。人行设备包括人脸门禁、刷卡门禁、二维码门禁、蓝牙门禁和可视对讲功能。

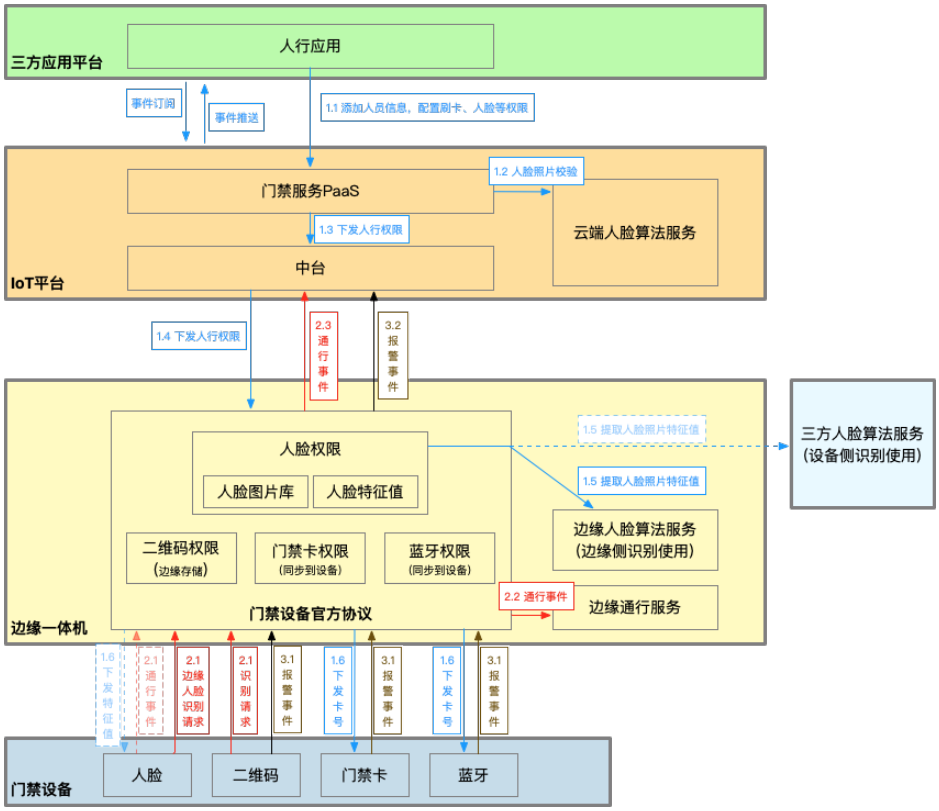
2 方案介绍

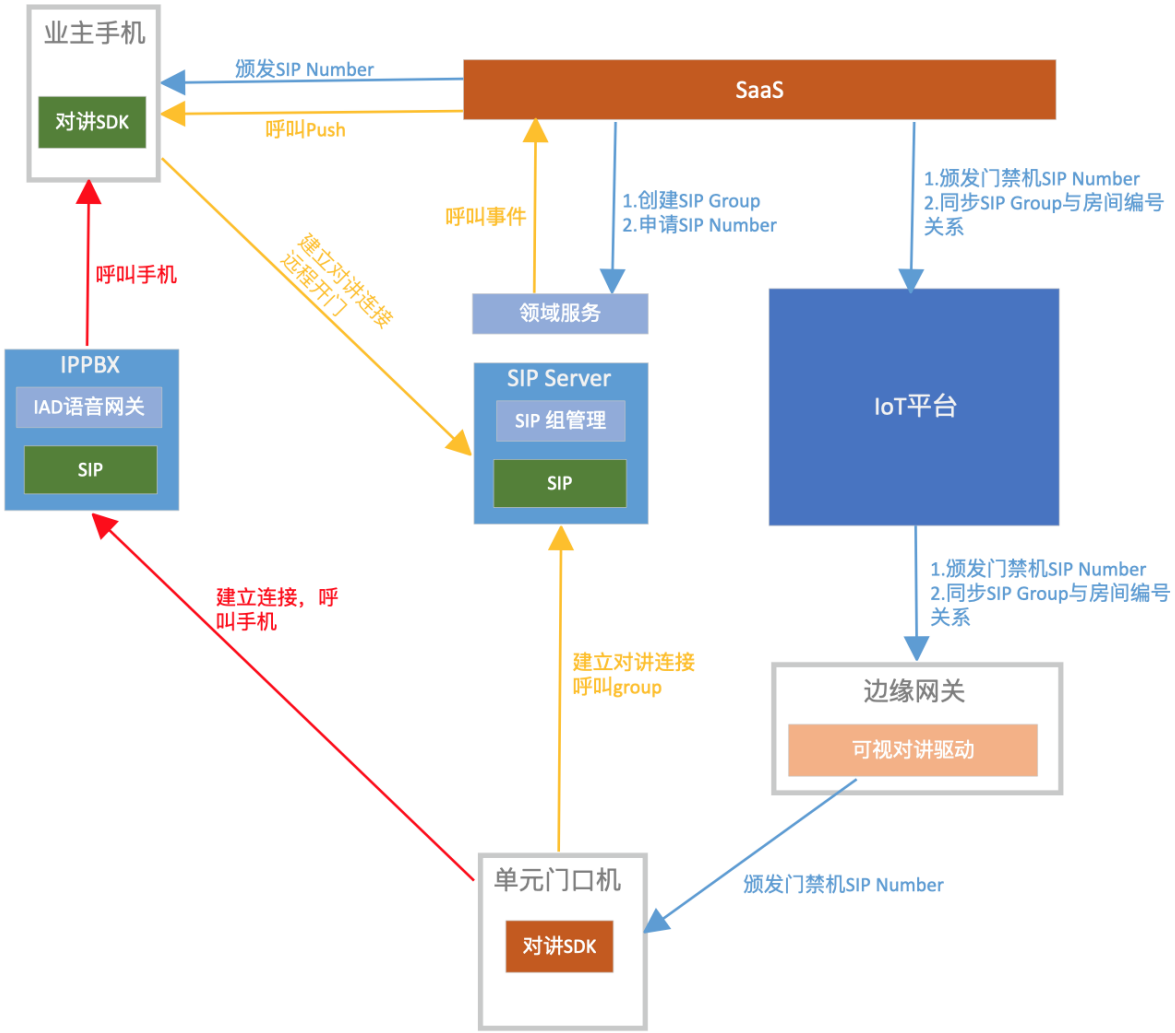
功能	方案描述

人脸门禁	人脸识别方案分为边缘服务器端比对和设备端比对。
	• 边缘服务器端比对：
	  ◦ 特点：人脸照片存储在边缘服务器上，特征值存储在边缘服务器上。
	  ◦ 流程：门禁机把抓拍、扣脸之后的照片发送给边缘服务器端比对，门禁按照比对结果开门。设备需要开发对接边缘服务器人脸识别接口。
	• 设备端比对：
	  ◦ 特点：人脸照片存储在边缘服务器上，特征值存储在设备上。
	  ◦ 流程：边缘服务器提取人脸图片特征值后，同步给门禁机，设备端进行人脸识别和比对，设备端需要保持和边缘服务器端算法模型一致。设备需要开发对接边缘服务器同步人脸特征值和通行事件接口。

二维码门禁	云端平台生产二维码，下发到边缘服务器，门禁机识别到二维码发送给边缘服务器比对，按照比对结果开门，设备需要开发对接边缘服务器二维码识别接口。
刷卡门禁	云端平台管理门禁卡权限，通过边缘服务器，下发门禁卡白名单到门禁机，门禁机自行判断刷卡通行，设备需要开发对接边缘服务器卡权限和事件接口。
蓝牙门禁	云端平台管理蓝牙卡权限，通过边缘服务器，下发蓝牙卡白名单到门禁机，门禁机通过蓝牙通信标准协议与手机APP通信，判断通行权限，设备需要开发对接边缘服务器卡权限和事件接口，蓝牙通信标准。
可视对讲	设备端开发对接云端平台和边缘服务器，实现云对讲和呼手机功能。

3 架构介绍





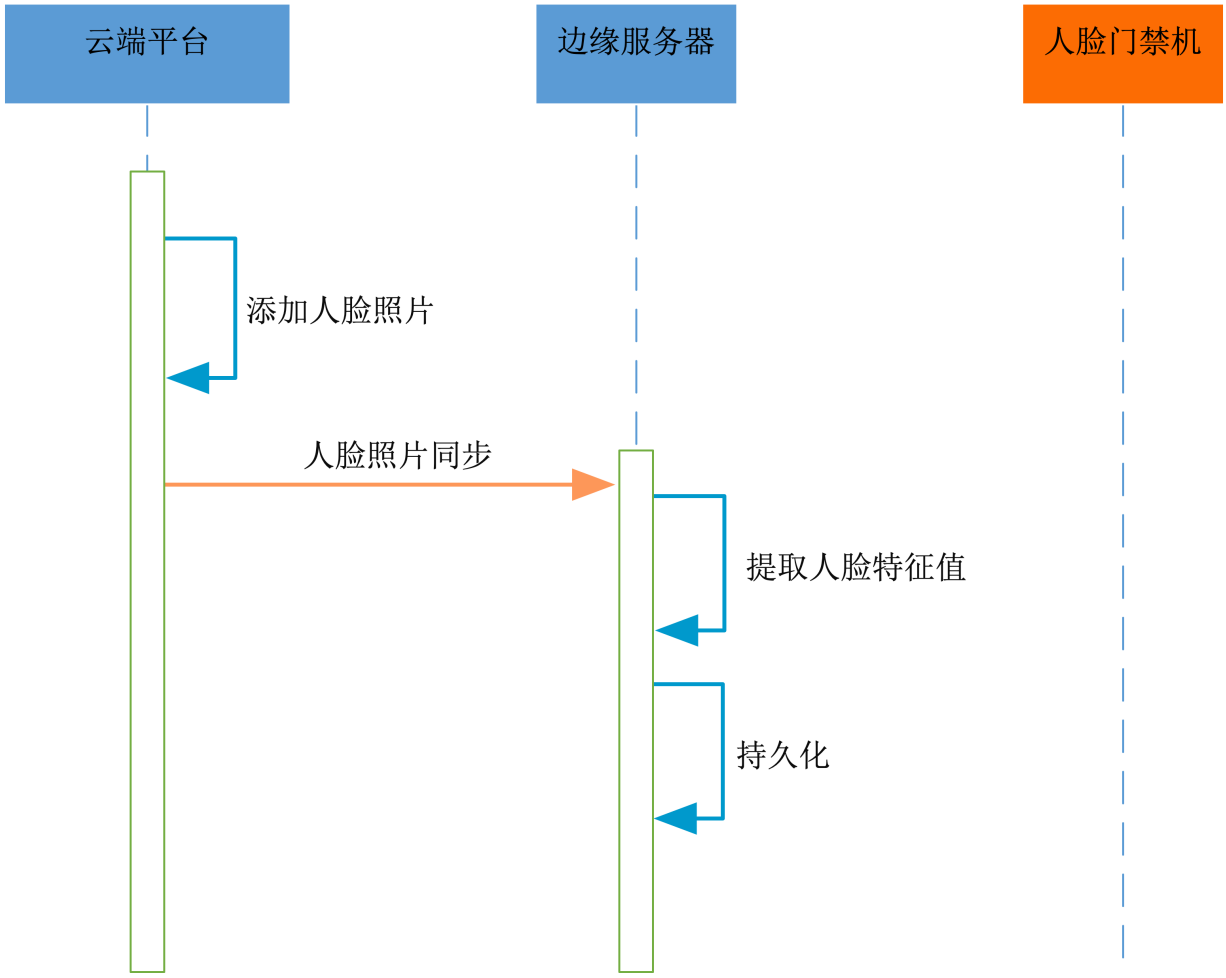
- 边缘网关管理可视对讲单元门口机在IoT平台上的在线状态，并负责下发SIP Number等。
- 可视对讲单元门口机与手机的视频或者语音通话过程由独立的SIP server及IPPBX管理。

4 交互流程

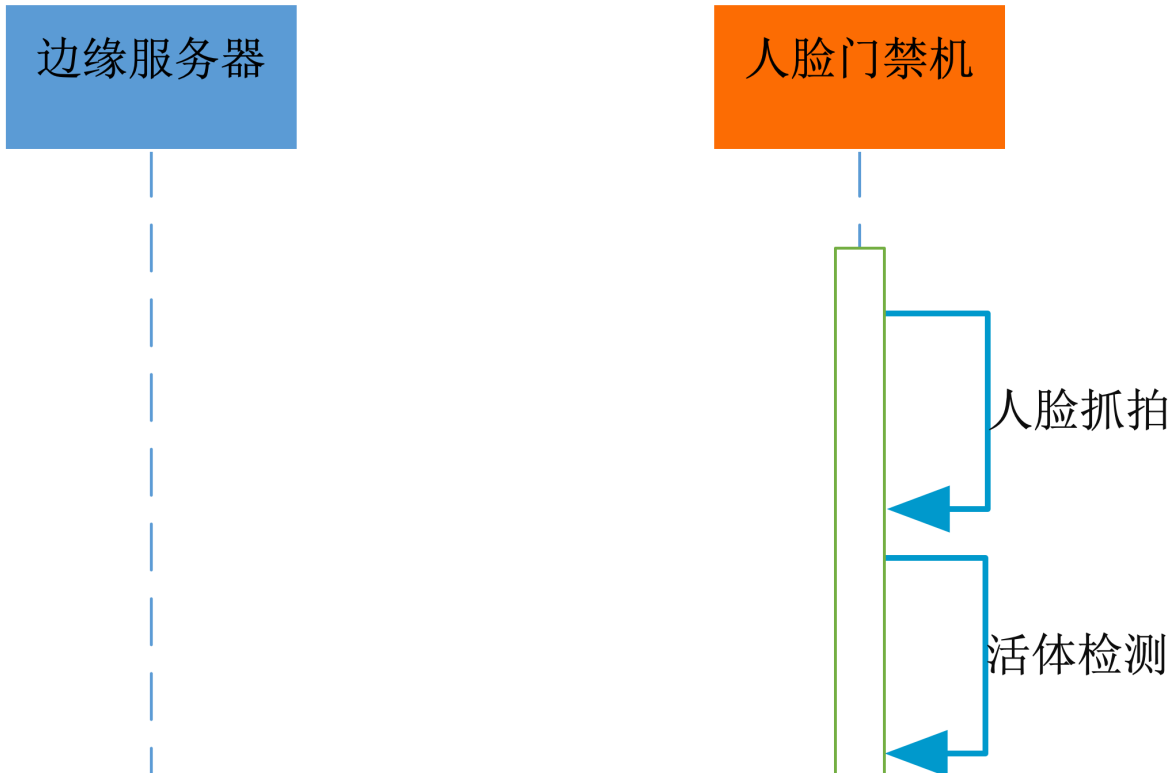
4.1 人脸门禁

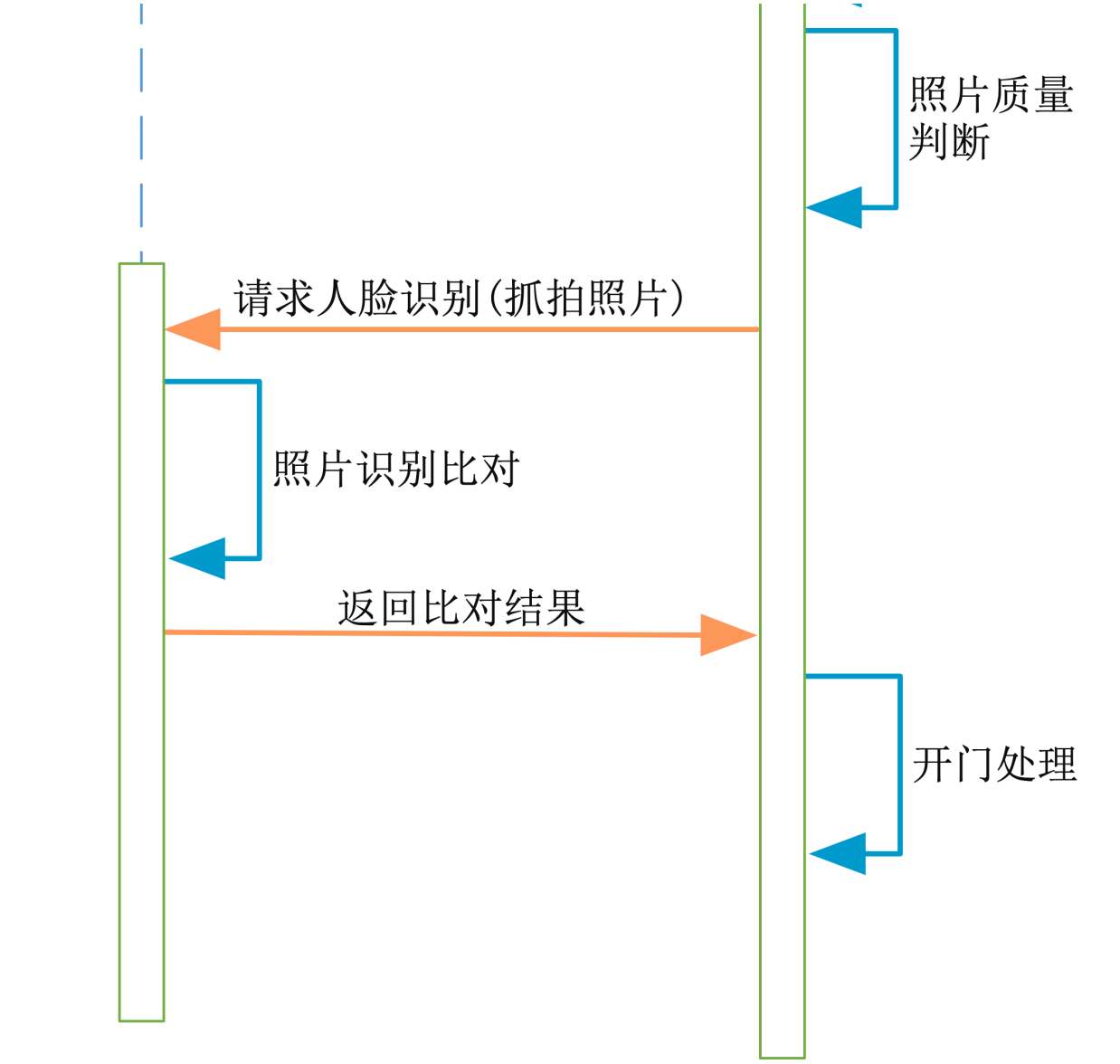
4.1.1 边缘端比对

人脸图片下发流程


云端平台
边缘服务器
人脸门禁机
添加人脸照片
人脸照片同步
提取人脸特征值
持久化

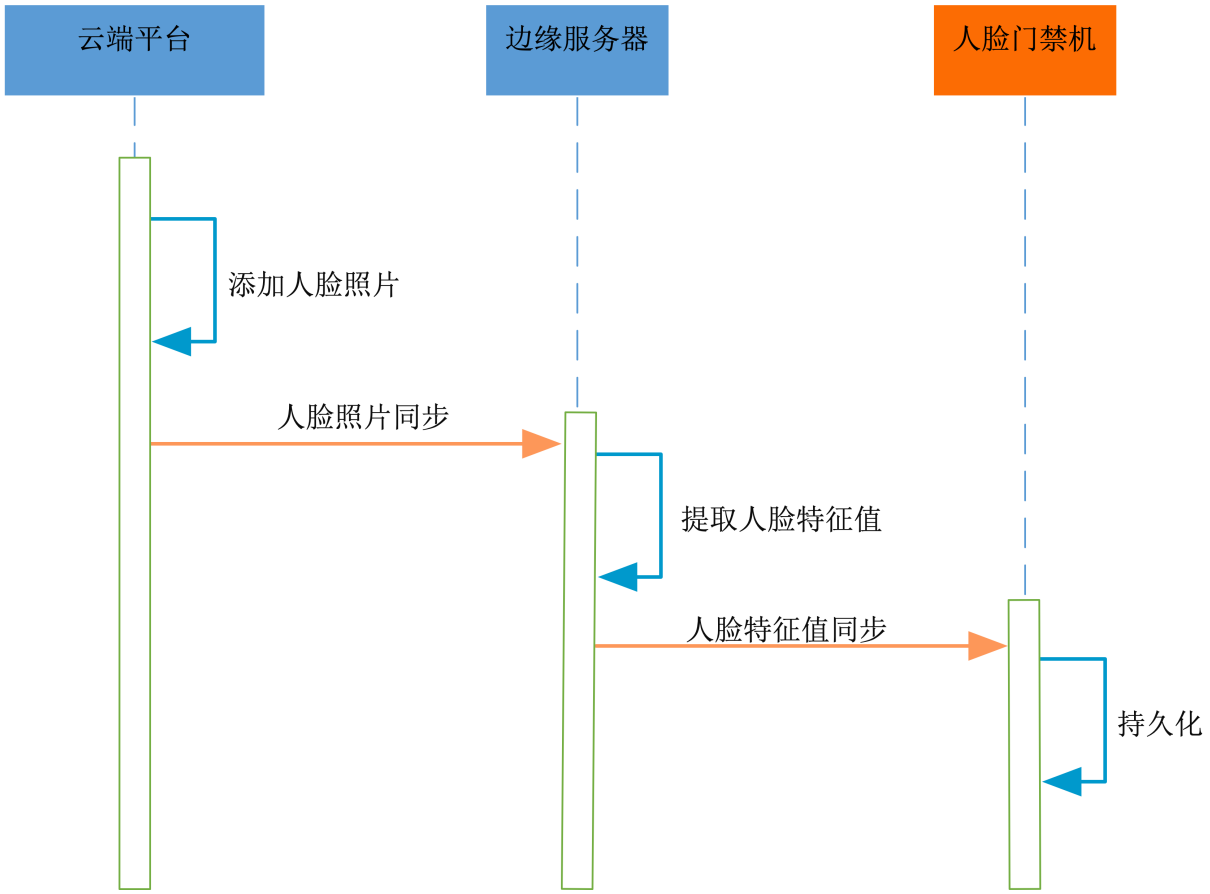
人脸识别流程


边缘服务器
人脸门禁机
人脸抓拍
活体检测

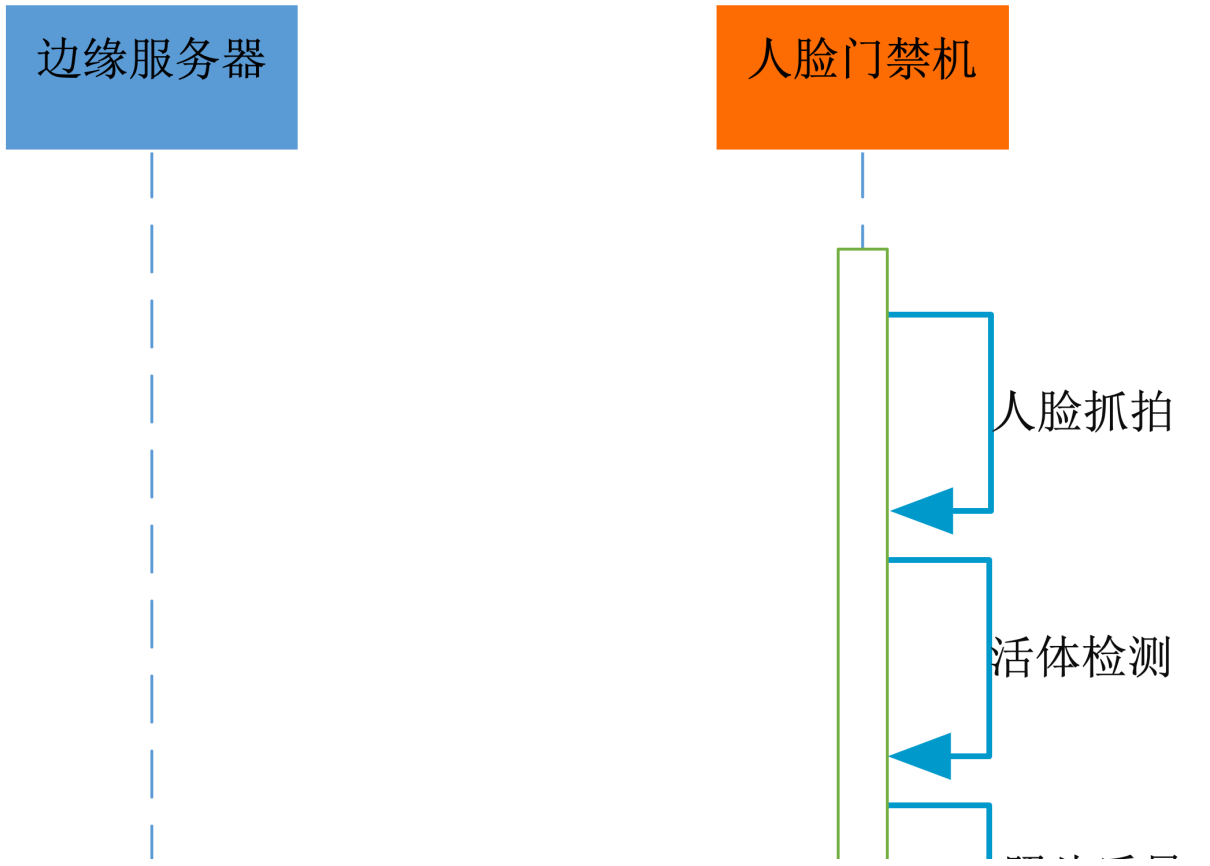


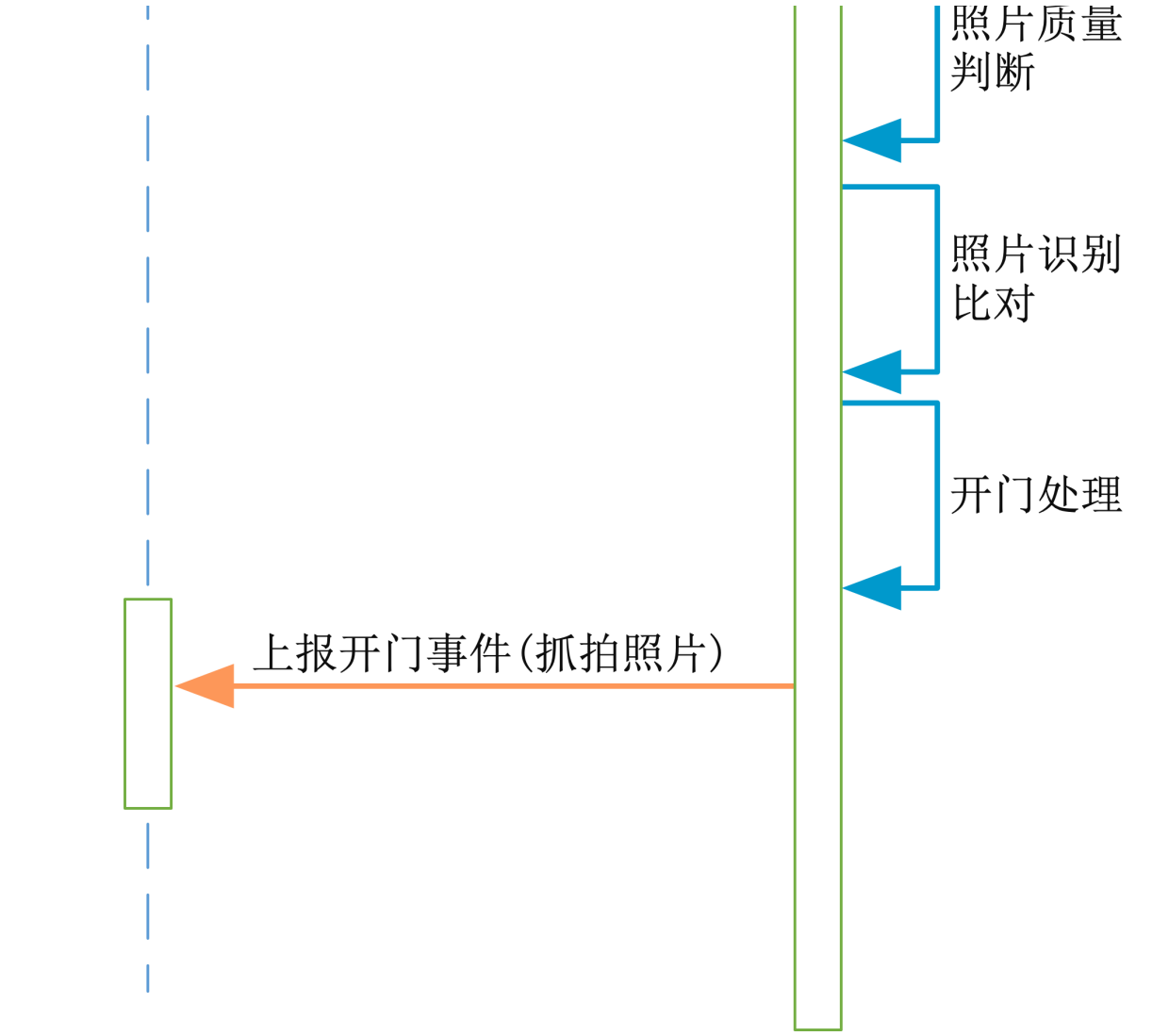
4.1.2 设备端比对

人脸图片下发流程

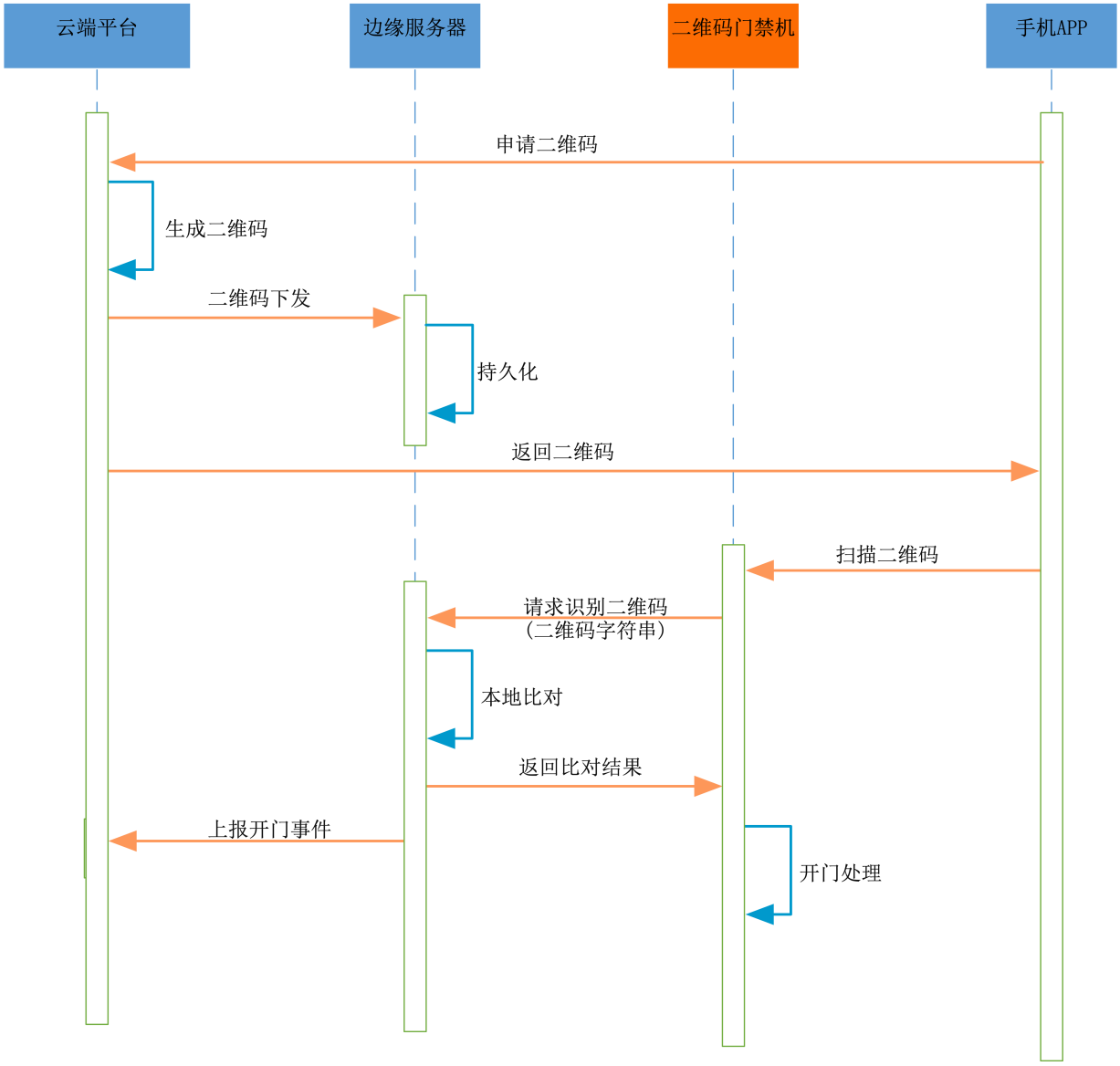

云端平台
边缘服务器
人脸门禁机
添加人脸照片
人脸照片同步
提取人脸特征值
人脸特征值同步
持久化

人脸识别流程


边缘服务器
人脸门禁机
人脸抓拍
活体检测

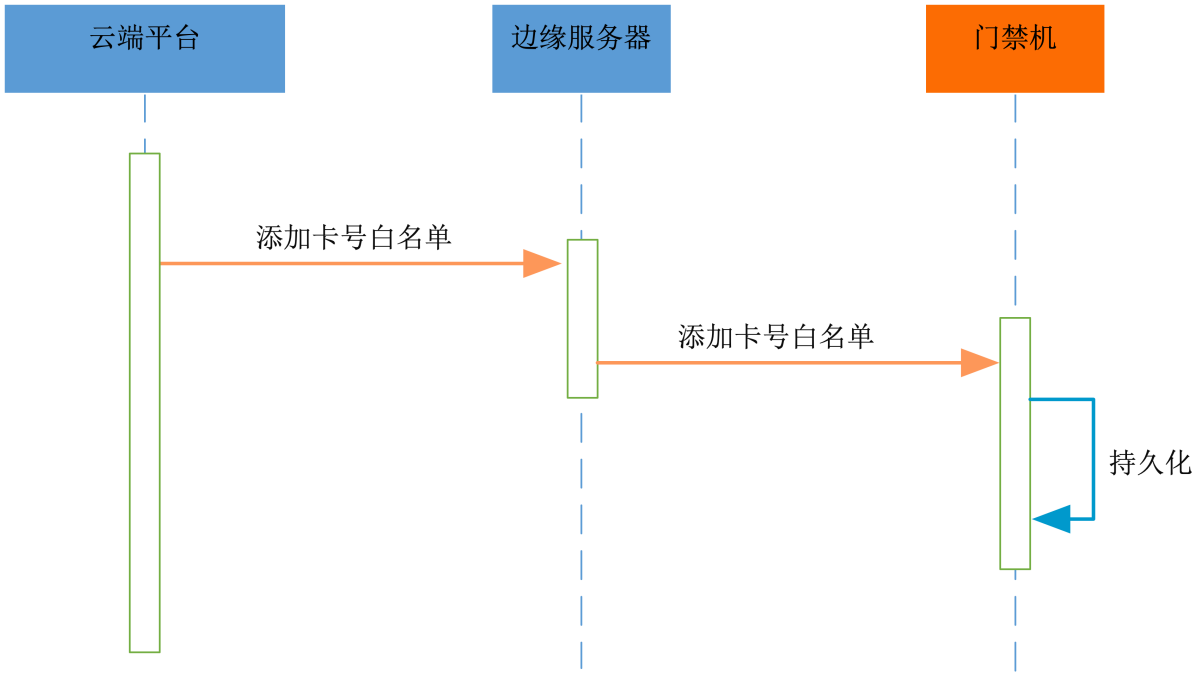


4.2 二维码门禁

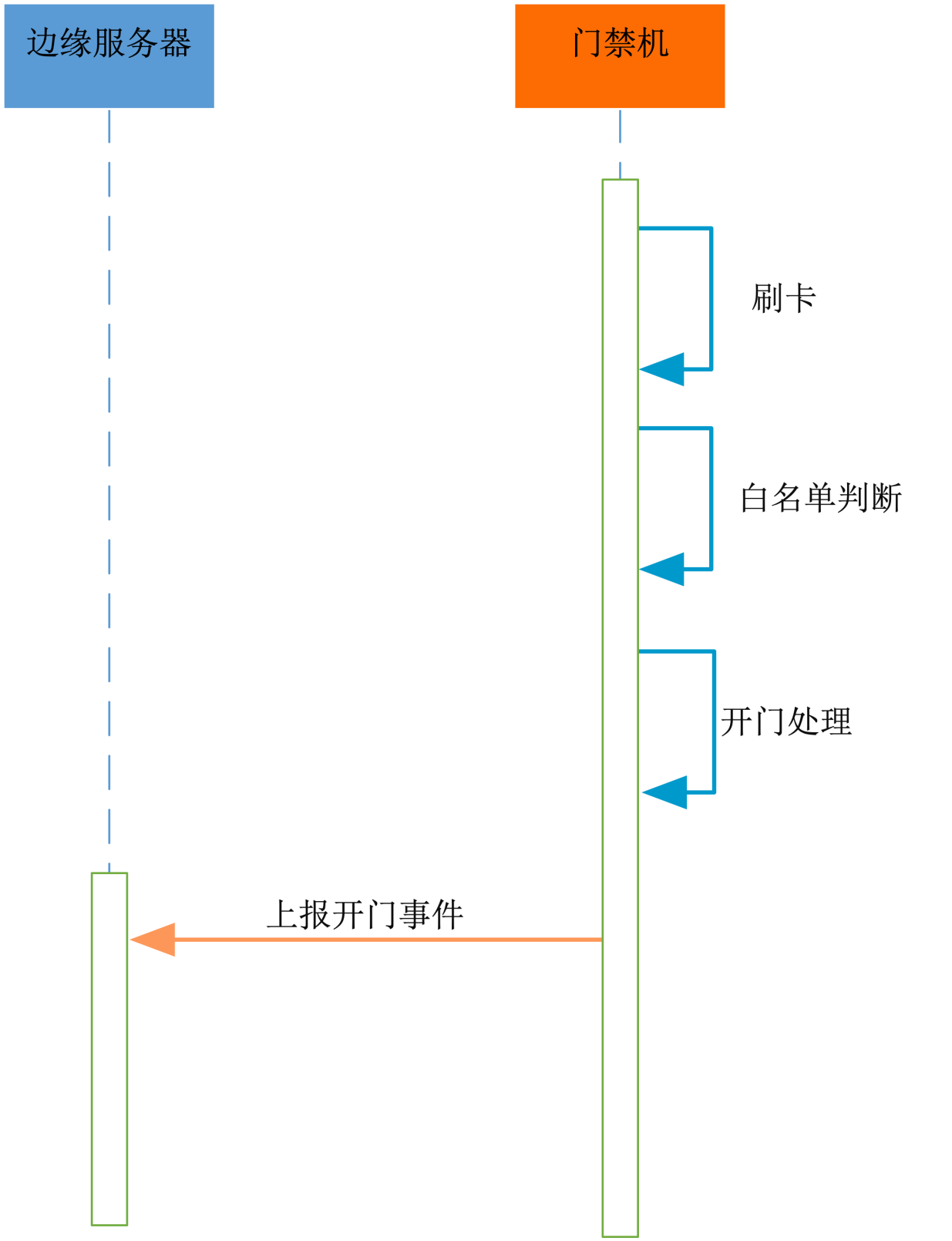


4.3 刷卡门禁

卡号绑定流程

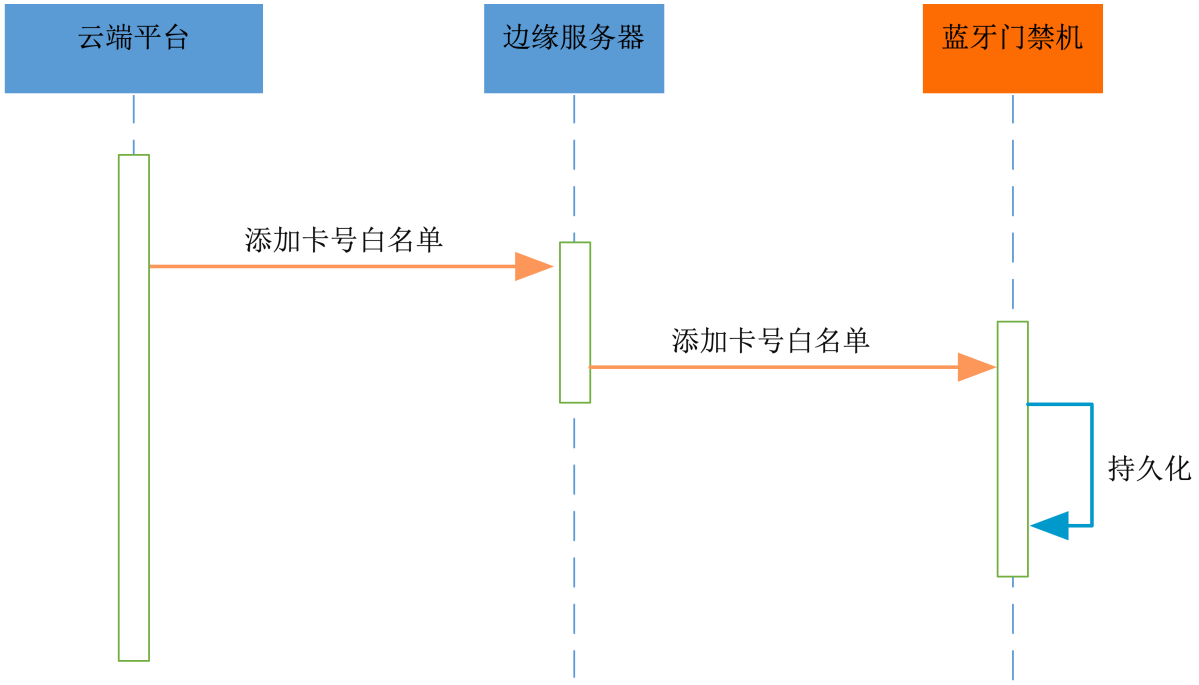


开门流程

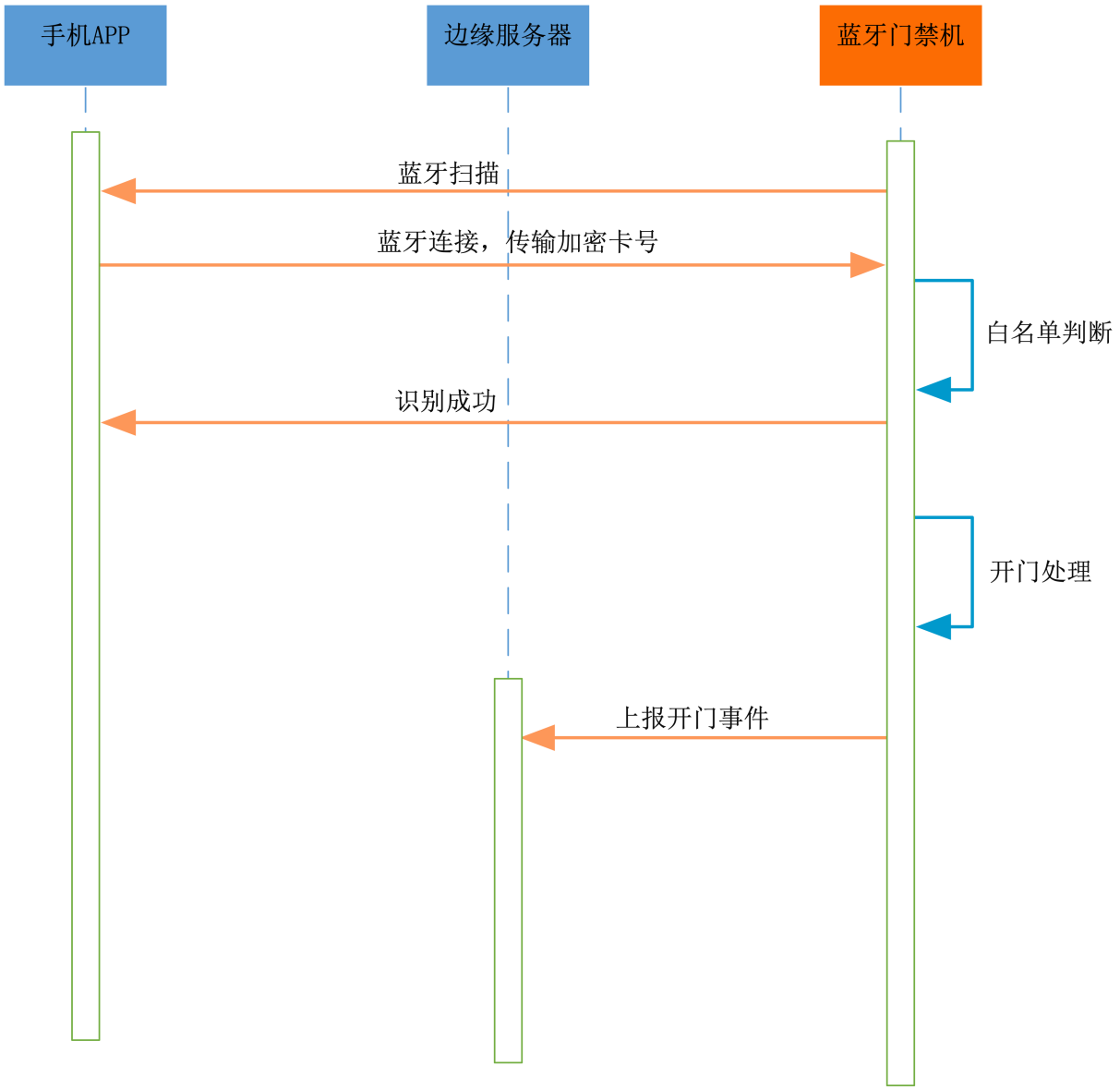


4.4 蓝牙门禁

卡号绑定流程

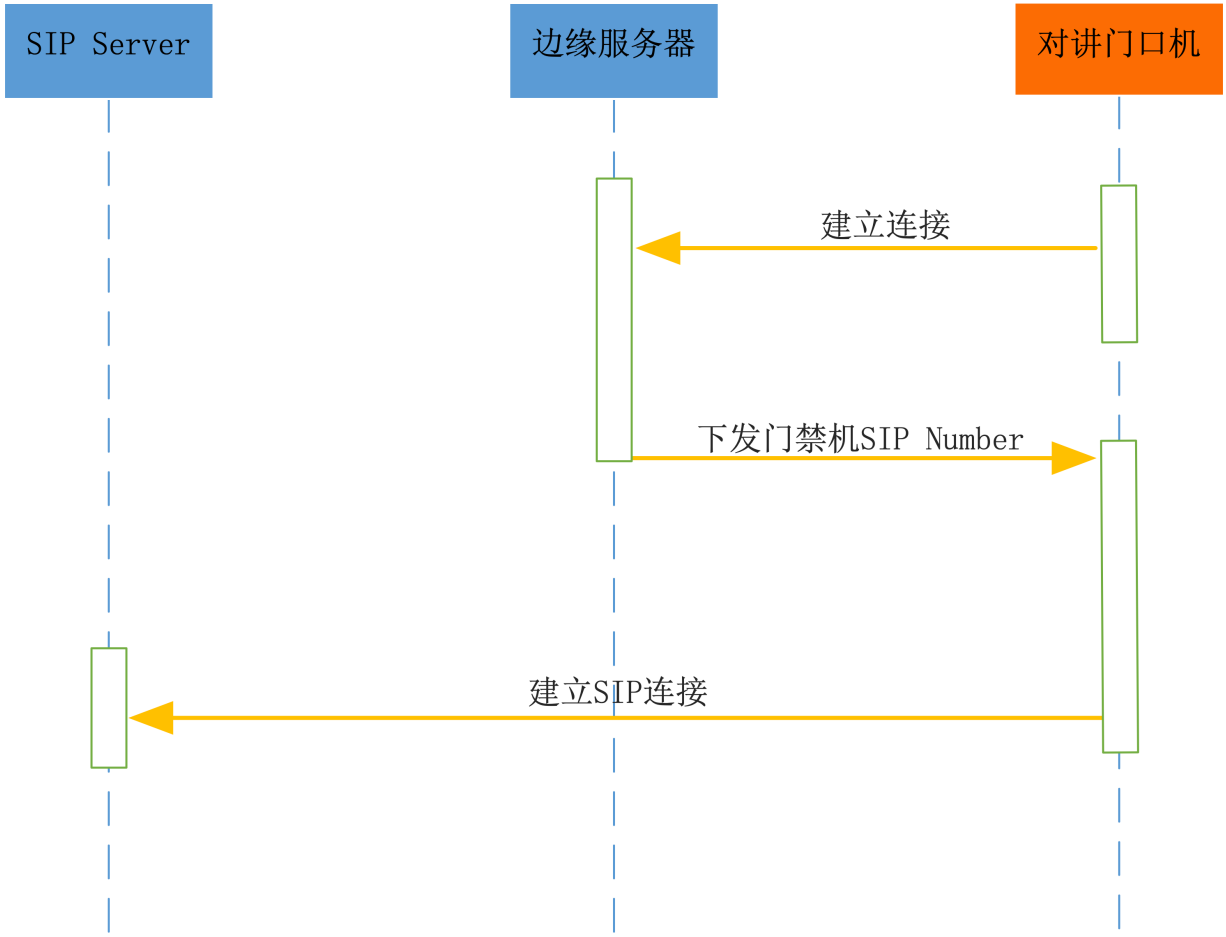


开门流程

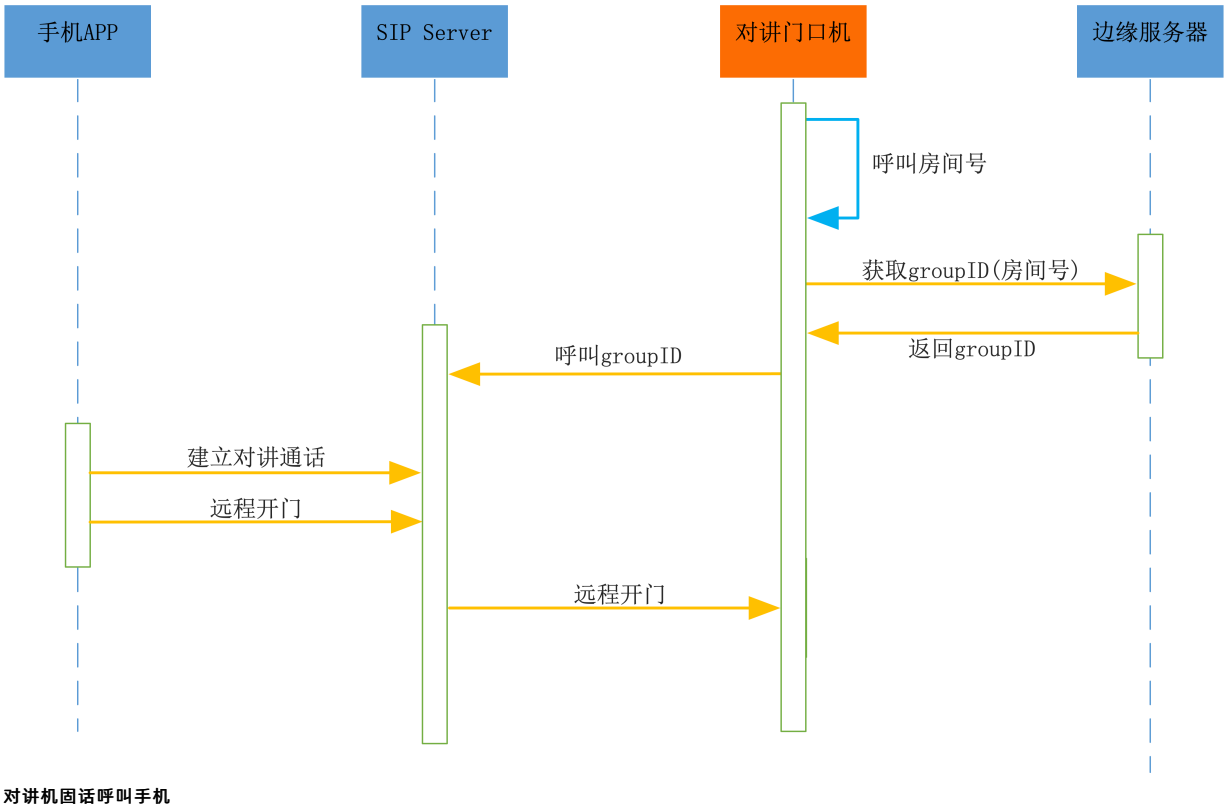


4.5 可视对讲

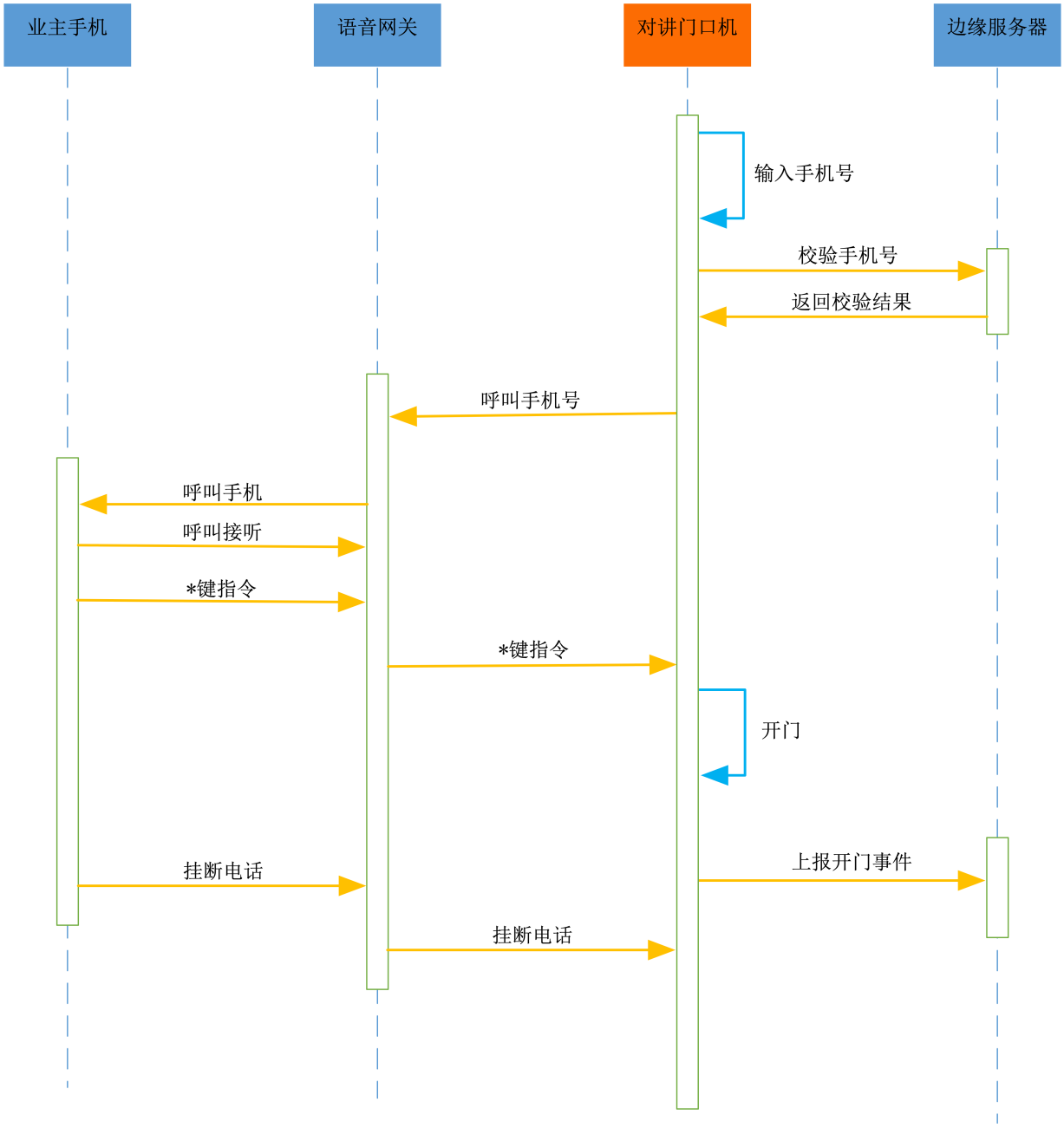
设备初始化



云对讲呼叫手机APP



对讲机固话呼叫手机



5 对接流程

5.1 对接说明

- 门禁功能（人脸、二维码、刷卡、蓝牙）、可视对讲功能、门禁机连接的电梯控制器功能访问边缘服务器10000端口
- 如果设备支持端侧人脸识别、刷卡、蓝牙、远程开门功能，需要设备端也建立HTTP Server，用以边缘服务器向设备推送数据。

5.2 对接准备

- **设备ID**：device ID做为设备的唯一标识，建议厂商根据MAC地址生成，长度为4-32个字符，可以包含英文字母、数字和特殊字符：连字符（-）、下划线（_）、at符号（@）、点号（.）、和英文冒号（:）。
- **人脸照片**：准备人脸照片一张，图片分辨率不低于120 * 120 像素，大小不超过1MB。
- **二维码**：长度为32位字符串的二维码，例如：ALI5466697328cff9121c21cf59dcec4
- **门禁卡号**：例如：F47B0BA2
- **蓝牙卡号**：例如：514152694808390

5.3 对接开发

接口定义，参考下面2个文档，根据实际对接内容，完成对接开发。

《2.2 门禁一体机标准协议接口定义》

《2.5 可视对讲SIP服务器接口定义》

蓝牙通信协议，参考下面文档。

《2.3 门禁设备与手机蓝牙对接协议》

可视对讲设备SDK与SIP协议，参考下面2个文档。

设备端Android SDK：《2.4 可视对讲设备端SDK接口说明[Andriod]》

SIP接入协议：《2.5 可视对讲SIP服务器接入协议》

3.5.1.2.2. 2.2 门禁一体机标准协议接口定义

1 方案介绍

门禁一体机标准协议支持人脸识别、二维码、刷卡、蓝牙、可视对讲、梯控功能接入物业管理一体机。

2 通用

门禁机对接边缘服务器，访问10000端口。

边缘服务器访问门禁机的端口由设备厂商自定义，定义后固定不变。

所有HTTP 报文的首部信息中，不能带有Connection: Keep-Alive字段，可以设置为Connection:Close。

物业管理一体机，以下简称“边缘服务器”。

下面接口是必须实现的通用接口。

2.1 建立连接

调用此接口建立门禁机与边缘服务器的双向通信连接，门禁设备跟边缘服务器互为客户端和服务端。

```
URL：http://[边缘服务器IP 或 门禁机IP]:[端口]/connect
Method：POST
Content-Type：application/json
```

说明

连接建立流程：

1. 门禁机携带下面的请求参数向边缘服务器发起连接请求。
2. 边缘服务器校验ip参数是否已经配置，如未配置，拒绝连接请求。设备信息配置需要预先在云端配置好，并部署到边缘。
3. 门禁机收到边缘服务器返回的成功消息后，单向连接建立完成。响应消息中包含一个有效的token信息。
4. 边缘服务器携带下面的请求参数向门禁机发起连接请求。
5. 门禁机收到请求后，返回成功消息，里面包含一个有效的token信息。
6. 边缘服务器收到成功消息后，双向连接建立完成。

请求参数

名称	类型	是否必选	示例值	描述
sn	String	是	123456789	长度为4-32个字符，可以包含英文字母、数字和特殊字符：连字符（-）、下划线（_）、at符号（@）、点号（.）、和英文冒号（:）。
time	Integer	是	1589472000	请求的时间戳。以秒为单位，需要使用UTC时间。
ip	String	是	192.168.1.128	连接发起方的IPv4地址。
mac	String	是	00-**-3D-40	连接发起方的MAC地址。
keepalive	Integer	是	30	单位：秒，合法取值范围[10，120]。
signMethod	String	是	md5	签名方式，目前支持MD5、SHA-1、SHA-256。取值范围{ md5, sha1, sha256 }。

sign	String	是	4db511**42	需要签名的内容封装格式： "TIME{time}MAC{mac}SN{sn}IP{ip}"大括号内的是具体内容，其外面的是名称。备注：time要转换成string类型。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
data	JSONObject		调用成功时，返回内容定义如下。

名称	类型	示例值	描述
token	String	6f2a-***-1-128	连接成功后，由接收连接的server端颁发的连接成功标识，代表一个有效连接，最大长度位64字节。在后续client的请求中需要携带，如“心跳保活”。

示例

请求示例

```
{
    "sn": "123456789",
    "time": 1589472000,
    "ip": "192.168.1.128",
    "mac": "00-16-EA-AE-3D-40",
    "keepalive": 30,
    "signMethod": "md5",
    "sign": "4db51117ad87cef349f45a3b14c05142"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success",
    "data":{
        "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128"
    }
}
```

错误码

code	描述
7	设备ip地址未在边缘服务器上注册。需要在IoT物联网平台上添加子设备，配置好ip，部署到边缘服务器上，即可。

附录查看更多错误码

2.2 心跳保活

门禁机与边缘服务器之间定期发送心跳消息，以确保消息通道的连通性，维持对方的在线状态。

```
URL: http://[边缘服务器IP 或 门禁机IP]:[端口]/keepalive
Method: POST
Content-Type: application/json
```

说明

建立连接后，服务器会向客户端颁发一个token，心跳消息中需要携带此token。当发送心跳时，对方返回token无效，那么需要重新建立连接，获取有效token。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-1-128	有效连接的标识。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

code	描述
4	token失效，需要重新建立连接，获取有效token。

附录查看更多错误码

2.3 远程开门

边缘服务器发送开门指令给门禁机。

```
URL: http://[门禁机IP]:[端口]/remoteOpen
Method: POST
Content-Type: application/json
```

说明

云端调用IoT设备的远程开门服务，边缘端接收到消息后，发送开门指令给门禁机，设备收到指令后开门放行。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-1-128	有效连接的标识。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

code	描述
4	token失效，需要重新建立连接，获取有效token。

附录查看更多错误码

2.4 报警事件

门禁机上报报警事件给边缘服务器。

```
URL: http://[边缘服务器IP]:[端口]/alarmEvent
Method: POST
Content-Type: application/json
```

说明

门禁机上报设备报警事件给边缘服务器，边缘服务器发送事件到云端。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
alarmType	Integer	是	1	报警类型，枚举如下： • 0：防拆报警 • 1：防拆报警解除 • 2：无效门禁卡 • 3：无效蓝牙卡 • 4：强制开门报警 • 5：强制开门报警解除 • 6：超时未关门报警 • 7：超时未关门报警解除
cardId	String	否	777342461	门禁卡或蓝牙卡的卡号。当alarmType为2或3时，需要加入此参数。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "alarmType":2,
    "cardId":"777342461"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

code	描述
4	token失效，需要重新建立连接，获取有效token。

附录查看更多错误码

2.5 设置常开状态

边缘服务器设置门禁设备的常开状态。

```
URL：http://[门禁机IP]:[端口]/setKeepOpen
Method：POST
Content-Type：application/json
```

说明

目前设置门常开状态的功能只支持具有刷卡功能的门禁设备。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
data	JSONArray	是		
• keepOpen	Integer	是	1	门禁设备常开状态，枚举如下： • 0：关闭 • 1：开启（常开状态）

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "data":{
        "keepOpen": 1
    }
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

附录查看更多错误码

2.6 查询常开状态

边缘服务器查询门禁设备的常开状态。

```
URL: http://[门禁机IP]:[端口]/getKeepOpen
Method: POST
Content-Type: application/json
```

说明

目前查询门常开状态的功能只支持具有刷卡功能的门禁设备。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
data	JSONObject		
• keepOpen	Integer	0	门禁设备常开状态，枚举如下： • 0：关闭 • 1：开启（常开状态）

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success",
    "data":{
        "keepOpen": 0
    }
}
```

错误码

附录查看更多错误码

2.7 设置测温规则

边缘服务器设置门禁的测温开关和测温的阈值。

```
URL：http://[门禁机IP]:[端口]/setTempCheckRule
Method：POST
Content-Type：application/json
```

说明

对门禁机设置用户通行认证的测温规则，包括测温开关，温度上下限。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
data	JSONObject	是		
• isCheck	Integer	是	1	门禁测温开关，枚举如下： • 0：关闭 • 1：开启
• maxThreshold	String	否	37.5	测温阈值的上限。默认单位为摄氏度。isCheck为1时，传递此参数。
• minThreshold	String	否	36.5	测温阈值的下限。默认单位为摄氏度。isCheck为1时，传递此参数。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "data":{
        "isCheck": 1,
        "maxThreshold": "37.5",
        "minThreshold": "36.5"
    }
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

3 人脸

3.1 【边缘端识别】人脸识别

门禁机抓拍通行人员的人脸照片，向边缘服务器发起人脸识别请求。

```
URL：http://[边缘服务器IP]:[端口]/recognize
Method：POST
Content-Type：multipart/form-data
```

说明

当有人员通行时，门禁设备向边缘服务器发起人脸识别。边缘认证成功后，返回给门禁设备允许放行，并上报人脸匹配通行事件。如果认证失败，返回失败结果给设备。

如果开启体温检测功能，体温值超过正常范围，则不允许通行。体温的正常范围由后台配置。

图片质量要求：

- 抓拍的图片中需要包含完整人脸像素
- 可根据自身设备能力选择在设备端或者边缘端完成人脸抠图处理（推荐设备端完成）
- 如果设备端进行人脸抠图，要求图片尺寸(宽、高)保证一致
- 图片分辨率不低于120 * 120 像素，大小不超过1MB

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
photo	File	是		支持jpg，jpeg，png格式
disableEdgeDetectFace	String	否	1	参数固定为1，表示边缘服务器对图片中的人脸进行检测定位，可以提高识别准确率，但识别时间会增加。
temperature	String	否	36.5	通过人脸识别获取的面部体温值。默认单位为摄氏度。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
data	JSONObject		调用成功时，返回内容定义如下。
• faceId	String	ac1c2-31a7-3**066	人脸（人员身份）的唯一标识，字符串长度最大64字节。
• name	String	王*飞	识别的人员名字。
• photoId	String	d8a9a-dd-**-d355	在边缘人脸照片库中，通行人员人脸照片的唯一标识。
• userId	String	777342461	用户ID

示例

请求示例

```
token: "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128"
photo: xxx.jpg
disableEdgeDetectFace: "1"
temperature: "36.5"
```

正常返回示例

```
{
    "code": 0,
    "message": "success",
    "data":{
        "faceId": "ac4af1c2-31a7-4206-9c8d-2a69cdaac066",
        "name": "王小飞",
        "photoId": "d8a0e59a-ddeb-11ea-a8c7-6c96cfdfd355",
        "userId": "123456"
    }
}
```

错误码

code	描述
1	人脸识别认证失败。

附录查看更多错误码

3.2 【设备端识别】同步人脸信息

边缘服务器同步人脸特征值信息到门禁机。

```
URL: http://[门禁机IP]:[端口]/faceAC/syncFaces
Method: POST
Content-Type: application/json
```

说明

当人脸数据超过50条，会被拆分为多次同步调用。每次同步最大50条，每次同步超时时间为10秒。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
faces	JSONArray	是		
• type	Integer	是	1	人脸信息操作类型，枚举如下： • 1：增加 • 2：删除 • 3：更新 • 4：删除全部 更新操作中有三种情形： 情形一：只更新userInfo 情形二：只更新feature 情形三：userInfo和feature都更新
• faceId	String	否	ac1c2-31a7-3**066	人脸（人员身份）的唯一标识，字符串长度最大64字节。
• feature	JSONArray	否		人脸照片的特征值。 元素类型为浮点型。
• featureMd5	String	否	259**704	人脸照片特征值（feature）的MD5值。
• userInfo	JSONObject	否		人员信息。

◦ userId	String	否		用户ID 增加、更新时必选
◦ userName	String	否	王*飞	人员姓名。
◦ expiryTime	String	否	1905004800000	人脸信息失效时间。以毫秒为单位，需要使用UTC时间。
◦ isInBlackList	Bool	否	true	代表人员是否在黑名单中。
• userInfoMd5	String	否	56s**sk9	人员信息（userInfo）的MD5值。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "faces":[
        {
            "type":4
        },
        {
            "type":1,
            "faceId":"452f3570-4717-4a39-910b-d72542550001",
            "userInfo":{
                "userName":"张三",
                "expiryTime":"1599892792000",
                "isInBlackList":false
            },
            "userInfoMd5":"25984f381c98bfc842280b528f424701",
            "feature":[0.0069077396765351295, 0.06679696589708328,  ...],
            "featureMd5":"4be6f1e4bda138a555416b866a90530c"
        },
        {
            "type":3,
            "faceId":"452f3570-4717-4a39-910b-d72542550002",
            "feature":[0.0069077396765351295, 0.06679696589708328,  ...],
            "featureMd5":"4be6f1e4bda138a555416b866a90530c"
        },
        {
            "type":3,
            "faceId":"452f3570-4717-4a39-910b-d72542550003",
            "userInfo":{
                "userName":"李四",
                "expiryTime":"1599892792111",
                "isInBlackList":true
            },
            "userInfoMd5":"25984f381c98bfc842280b528f424703"
        },
        {
            "type":3,
            "faceId":"452f3570-4717-4a39-910b-d72542550004",
            "userInfo":{
                "userName":"赵五",
                "expiryTime":"1599892792222",
                "isInBlackList":false
            },
            "userInfoMd5":"25984f381c98bfc842280b528f424704",
            "feature":[0.0069077396765351295, 0.06679696589708328,  ...],
            "featureMd5":"4be6f1e4bda138a555416b866a90530c"
        },
        {
            "type":2,
            "faceId":"452f3570-4717-4a39-910b-d72542550009"
        }
    ]
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

code	描述
0	同步成功，非0为同步失败。

附录查看更多错误码

3.3 【设备端识别】查询设备上人脸库信息

边缘服务器查询设备上的人脸特征值信息。

```
URL: http://[门禁机IP]:[端口]/faceAC/queryFaces
Method: POST
Content-Type: application/json
```

说明

当门禁机首次连接到边缘服务器或重新连接到边缘服务器时，获取设备上的人脸特征值存储情况，用于计算还需要同步到设备端的人脸数据。每次查询超时时间为10秒。

当人脸数量较多时，支持分页查询。边缘服务器先向设备发送全量查询请求，如果设备上的人脸数超过设备可处理的最大值，那么设备返回第1页的人脸数据和分页信息，之后边缘服务器会进行分页查询。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
currentPage	Integer	否	1	当前页号。 如果不传此参数，表示全量查询。设备最多返回一次可查询的最大人脸数。
pageSize	Integer	否	200	指定返回结果中每页的最大人脸数量。 与currentPage参数一起使用。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
data	JSONObject		
• currentPage	Integer	1	当前页号。
• pageCount	Integer	93	总页数。
• pageSize	Integer	200	每页的最大人脸数量。
• total	Integer	18483	设备上的人脸总数。
• faceInfo	JSONArray		
	String	ac1c2-31a7-3**066	人脸（人员身份）的唯一标识，字符串长度最大64字符。
	String	259**704	人员信息（userInfo）的MD5值。
	String	56s**sk9	人脸照片特征值（feature）的MD5值。

说明：faceInfo中元素顺序固定。

1. 人脸的唯一标识
2. 人员信息的MD5值
3. 人脸照片特征值的MD5值

示例

请求示例

```
{
    "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "currentPage": 1,
    "pageSize": 200
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success",
    "data": {
        "currentPage": 1,
        "pageCount": 93,
        "pageSize": 200,
        "total": 18483,
        "faceInfo": [
            [
                "452f3570-4717-4a39-910b-d72542550001",
                "25984f381c98bfc842280b528f424701",
                "4be6f1e4bda138a555416b866a90530c"
            ],
            [
                "452f3570-4717-4a39-910b-d72542550004",
                "25984f381c98bfc842280b528f424704",
                "4be6f1e4bda138a555416b866a90530c"
            ]
        ]
    }
}
```

错误码

code	描述
0	同步成功，非0为同步失败。

附录查看更多错误码

3.4 【设备端识别】人脸通行事件

门禁机上报通行人员识别人脸的通行事件给边缘服务器。

```
URL: http://[边缘服务器IP]:[端口]/faceAC/passEvent
Method: POST
Content-Type: multipart/form-data
```

说明

门禁机上报通行人员刷脸成功的事件给边缘服务器，边缘服务器发送事件到云端。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
faceId	String	是	ac1c2-31a7-3**066	人脸（人员身份）的唯一标识，字符串长度最大64字节。
similarity	String	是	0.68	通行人脸照片与人脸库照片匹配的相似度。内容为0-1之间的小数，0代表匹配度最低，1代表匹配的最高。
photo	File	是		人员通行时抓拍的人脸照片。
temperature	String	否	36.5	通过人脸识别获取的面部体温值。默认单位为摄氏度。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
data	JSONObject		
• photoId	String	d8a9a-dd-**-d355	在边缘人脸照片库中，通行人员人脸照片的唯一标识。

示例

请求示例

```
token: "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128"
faceId: "ac4af1c2-31a7-4206-9c8d-2a69cdaac066"
photo: xxx.jpg
similarity: "0.78654"
temperature: "36.5"
```

正常返回示例

```
{
    "code": 0,
    "message": "success",
    "data":{
        "photoId": "d8a0e59a-ddeb-11ea-a8c7-6c96cfdfd355"
    }
}
```

错误码

code	描述
5	人脸照片内容无效

附录查看更多错误码

4 二维码

4.1 二维码识别

门禁机识别通行人员的手机二维码，向边缘服务器发起二维码识别请求。

```
URL: http://[边缘服务器IP]:[端口]/qrcode
Method: POST
Content-Type: application/json
```

说明

当有人员通行时，门禁设备向边缘服务器发起二维码识别。边缘认证成功后，返回给门禁设备允许放行，并上报通行事件。如果认证失败，返回失败结果给设备。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
qrCode	String	是	ALIw***xz6x	长度为32个字节。

返回参数

名称	类型	示例值	描述

code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
data	JSONObject		
• userId	String	777342461	用户ID

示例

请求示例

```
{
    "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "qrCode": "ALI1234567890azxswdfghggfgsgERTG"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success",
    "data": {
     "userId": "123456"
    }
}
```

错误码

code	描述
1	二维码认证失败。

附录查看更多错误码

5 门禁卡

5.1 同步卡权限

边缘服务器同步卡权限信息到门禁机。

```
URL: http://[门禁机IP]:[端口]/cardAC/syncPermissions
Method: POST
Content-Type: application/json
```

说明

边缘服务器接收云端下发的卡权限信息，调用此接口发送到设备端。每次同步最大200条，同步超时时间为5秒。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
permissions	List	是		

• type	Integer	是	1	门禁卡操作类型，枚举如下： • 1：增加 • 2：删除 • 3：修改 说明：对于增加和修改处理方式相同，如果该卡信息存在则对已有的卡信息做更新处理，如果该卡信息不存在则做新增处理。
• userId	String	否	777342461	用户ID
• cardId	String	是	777342461	门禁卡的卡号。
• effectiveTime	String	是	1589472000000	门禁卡生效的时间。以毫秒为单位，需要使用UTC时间。
• expiryTime	String	是	1905004800000	门禁卡过期的时间。以毫秒为单位，需要使用UTC时间。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "permissions":[
        {
            "type":1,
            "cardId":"777342461",
            "effectiveTime":"1589472000000",
            "expiryTime":"1905004800000"
        },
        {
            "type":2,
            "cardId":"777342461",
            "effectiveTime":"1589472000000",
            "expiryTime":"1905004800000"
        },
        {
            "type":3,
            "cardId":"777342461",
            "effectiveTime":"1589472000000",
            "expiryTime":"1905004800000"
        }
    ]
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

附录查看更多错误码

5.2 刷卡通行事件

门禁机上报刷卡通行事件给边缘服务器。

```
URL: http://[边缘服务器IP]:[端口]/cardAC/passEvent
Method: POST
Content-Type: application/json
```

说明

门禁机上报人员刷卡通行的事件给边缘服务器，边缘服务器发送事件到云端。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
cardId	String	是	777342461	门禁卡的卡号。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "cardId": "777342461"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

附录查看更多错误码

6 蓝牙

6.1 同步蓝牙权限

边缘服务器同步蓝牙权限信息到门禁机。

```
URL: http://[门禁机IP]:[端口]/bleAC/syncPermissions
Method: POST
Content-Type: application/json
```

说明

边缘服务器接收云端下发的蓝牙权限信息，调用此接口发送到设备端。每次同步最大200条，同步超时时间为5秒。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
permissions	List	是		

• type	Integer	是	1	蓝牙卡操作类型，枚举如下： • 1：增加 • 2：删除 • 3：修改 说明：对于增加和修改处理方式相同，如果该卡信息存在则对已有的卡信息做更新处理，如果该卡信息不存在则做新增处理。
• userId	String	否	777342461	用户ID
• cardId	String	是	777342461	蓝牙卡的卡号。
• effectiveTime	String	是	1589472000000	蓝牙卡生效的时间。以毫秒为单位，需要使用UTC时间。
• expiryTime	String	是	1905004800000	蓝牙卡过期的时间。以毫秒为单位，需要使用UTC时间。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "permissions":[
        {
            "type":1,
            "cardId":"777342461",
            "effectiveTime":"1589472000000",
            "expiryTime":"1905004800000"
        },
        {
            "type":2,
            "cardId":"777342461",
            "effectiveTime":"1589472000000",
            "expiryTime":"1905004800000"
        },
        {
            "type":3,
            "cardId":"777342461",
            "effectiveTime":"1589472000000",
            "expiryTime":"1905004800000"
        }
    ]
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

附录查看更多错误码

6.2 蓝牙通行事件

门禁机上报蓝牙通行事件给边缘服务器。

```
URL: http://[边缘服务器IP]:[端口]/bleAC/passEvent
Method: POST
Content-Type: application/json
```

说明

门禁机上报人员蓝牙通行的事件给边缘服务器，边缘服务器发送事件到云端。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
cardId	String	是	777342461	蓝牙卡的卡号。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "cardId":"777342461"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

附录查看更多错误码

7 对讲

7.1 颁发门禁机SIP Number

边缘服务器同步SIP Number给门禁机。

```
URL: http://[门禁机IP]:[端口]/visualIntercom/syncSipNumber
Method: POST
Content-Type: application/json
```

说明

当云端下发的SIP账号时，边缘服务器推送SIP Number等信息给到门禁机。

每次门禁机与边缘服务器建立连接后，边缘服务器会再次同步SIP Number给门禁机。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
sipNumber	String	是	MNUt***CBz	连接SIP Server的账号。

password	String	是	******	连接SIP Server的密码。
sipServerAddr	String	是	47.103.*.251	SIP Server的IP地址。
sipServerPort	String	是	6050	SIP Server的端口。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "sipNumber": "MNUtFSr9soldjp34IxYCBz",
    "password": "******",
    "sipServerAddr": "47.103.XX.XXX",
    "sipServerPort": "6050"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

附录查看更多错误码

7.2 获取groupID

门禁机向边缘服务器查询房号的SIP组ID。

```
URL: http://[边缘服务器IP]:[端口]/visualIntercom/getGroupId
Method: POST
Content-Type: application/json
```

说明

用户输入呼叫的房号时，门禁机向边缘服务器查询房号对应的SIP group ID。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
roomId	String	是	1-9-3-1202	规则：期号-楼幢号-单元号-房号 例如： 1-9-3-1202 1-10-3A-1709 1-5-767-801 最大位数：1位-3位-4位-5位，规则数字+字母。 期号默认为1。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
groupId	String	4sews***HlbWkjbL	成功时，返回房号对应的SIP组ID。

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "roomId": "1-9-3-1202"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success",
    "groupId": "4sewsEXT8ezMnlHlbWkjbL"
}
```

错误码

附录查看更多错误码

7.3 验证手机号

门禁机向边缘服务器查询房号的SIP组ID。

```
URL：http://[边缘服务器IP]:[端口]/visualIntercom/checkPhoneNumber
Method：POST
Content-Type：application/json
```

说明

由门禁机通过边缘服务器向云端发送手机号校验请求。边缘服务器返回给门禁机的结果只能表明边缘服务器是否已成功向云端发起校验请求，并不能代表校验结果。校验结果会通过“7.4 手机号校验结果”接口异步下发，因此设备需要自己处理等待校验结果超时的情况。

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
phoneNumber	String	是	180****5678	待校验的手机号。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "phoneNumber": "18012345678"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

附录查看更多错误码

7.4 手机号校验结果

云端将手机号校验结果，通过边缘服务器下发给门禁机。

```
URL: http://[门禁机IP]:[端口]/visualIntercom/verifyPhoneResult
Method: POST
Content-Type: application/json
```

说明

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
phoneNumber	String	是	180****5678	待校验的手机号。
result	Integer	是	0	校验结果，枚举如下： • 0：成功 • 1：失败

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "phoneNumber": "18012345678",
    "result": 0
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

错误码

附录查看更多错误码

7.5 上报开门事件

门禁机向边缘服务器上报对讲开门事件。

```
URL: http://[边缘服务器IP]:[端口]/visualIntercom/reportOpenDoor
Method: POST
Content-Type: application/json
```

说明

请求参数

名称	类型	是否必选	示例值	描述
token	String	是	6f2a-***-128	有效连接标识。
type	Integer	是	0	对讲类型，枚举如下： • 0：云对讲 • 1：固话对讲
number	String	是	MNUt***CBz	云对讲类型，值为SipNumber。 固话对讲类型，值为手机号。
roomId	String	否	1-9-3-1202	对讲类型为云对讲时，提供此参数。 规则：期号-楼幢号-单元号-房号 例如： 1-9-3-1202 1-10-3A-1709 1-5-767-801 最大位数：1位-3位-4位-5位，规则数字+字母。 期号默认为1。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "type": 0,
    "number": "MNUtFSr9soldjp34IxYCBz",
    "roomId": "1-9-3-1202"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

8 梯控接口

8.1 上报电梯运行状态

梯控上报电梯的运行状态给边缘服务器。

```
URL：http://[边缘服务器IP]:[端口]/elevator/reportRunState
Method：POST
Content-Type：application/json
```

说明

每次建立连接后，上报一次电梯运行状态。当电梯运行状态变化时，立即上报实时的电梯工作状态。

限制

调用该接口的每秒请求数（QPS）最大限制为100。

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-12-13	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-12-13`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。
data	JSONArray	是		电梯信息。
• elevatorId	String	否	01	电梯ID，电梯的唯一标识。
• currentFloor	String	是	9	电梯当前所在楼层。
• carState	Integer	是	1	电梯轿厢的运行状态，枚举如下： • 0：停止 • 1：运行
• carDirection	Integer	是	2	电梯轿厢的运行方向，枚举如下： • 0：无方向 • 1：上行 • 2：下行

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
  "version": "2021-12-13",
  "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
  "data": [
    {
      "elevatorId": "01",
      "currentFloor": "9",
      "carState": 1,
      "carDirection": 2
    },
    {
      "elevatorId": "02",
      "currentFloor": "1",
      "carState": 0,
      "carDirection": 0
    }
  ]
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

8.2 查询电梯运行状态

边缘服务器获取电梯当前运行状态。

```
URL：http://[门禁机IP]:[端口]/elevator/getRunState
Method：POST
Content-Type：application/json
```

说明

可以查询一个或多个电梯的运行状态，如果未指定电梯ID，需要返回全部电梯的运行状态。

限制

调用该接口的每秒请求数（QPS）最大限制为100。

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-12-13	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-12-13`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。
elevatorList	JSONArray	否	01,02	待查询的elevatorId列表。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
data	JSONArray		电梯信息。
• elevatorId	String	01	【可选参数】电梯ID，电梯的唯一标识。
• currentFloor	String	9	【可选参数】电梯当前所在楼层。
• carState	Integer	1	【可选参数】电梯轿厢的运行状态，枚举如下： • 0：停止 • 1：运行
• carDirection	Integer	2	【可选参数】电梯轿厢的运行方向，枚举如下： • 0：无方向 • 1：上行 • 2：下行

示例

请求示例

```
{
  "version": "2021-12-13",
  "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
  "data": [
    "01",
    "02"
  ]
}
```

正常返回示例

```
{
  "code": 0,
  "message": "success",
  "data": [
    {
      "elevatorId": "01",
      "currentFloor": "9",
      "carState": 1,
      "carDirection": 2
    },
    {
      "elevatorId": "02",
      "currentFloor": "1",
      "carState": 0,
      "carDirection": 0
    }
  ]
}
```

8.3 释放受控楼层

边缘服务器发送释放楼层控制指令，通知电梯开放指定楼层，允许用户选择已被授权的楼层。

```
URL：http://[门禁机IP]:[端口]/elevator/releaseFloors
Method：POST
Content-Type：application/json
```

说明

当用户在轿厢内进行权限认证时，例如刷卡，电梯释放其所在楼层权限。多用于用户拥有多楼层权限的情况，由用户手动选择目的楼层。

限制

调用该接口的每秒请求数（QPS）最大限制为100。

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-12-13	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-12-13`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。
elevatorId	String	否	01	电梯的唯一标识。 如果不传此参数，表示操作所有电梯。
floorRoomList	JSONArray	否		楼层房号列表。 如果不传此参数，表示释放全部楼层。
• floor	String	否	9	需要释放的楼层。 如果存在贯通门的情况，使用房号区分不同的电梯门。如果不存在贯通门，不需要传房间号。

• room	String	否	601	需要释放的楼层的房间号。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
  "version": "2021-12-13",
  "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
  "elevatorId": "01",
  "floorRoomList": [
    {
      "floor": "9",
      "room": "901"
    },
    {
      "floor": "6",
      "room": "602"
    }
  ]
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

8.4 点亮目标楼层

边缘服务器发送点亮楼层控制指令，电梯自动选中并点亮指定楼层。

```
URL：http://[门禁机IP]:[端口]/elevator/activateFloor
Method：POST
Content-Type：application/json
```

说明

当用户在轿厢内进行权限认证时，例如刷卡，电梯自动点亮其所在楼层权限。多用于用户只拥有某一楼层权限的情况。

一次接口调用可点亮多个楼层。

限制

调用该接口的每秒请求数（QPS）最大限制为100。

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-12-13	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-12-13`。接口可能存在多个版本。

token	String	是	6f2a-***-128	有效连接标识。
elevatorId	String	否	01	电梯的唯一标识。 如果不传此参数，表示操作所有电梯或默认电梯。
floorRoomList	JSONArray	否		楼层房号列表。 如果不传此参数，表示释放全部楼层。
• floor	String	是		目标楼层
• roomNumber	String	否	01	目标房间号。 存在贯通门的情况，需要传递此参数。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
  "version": "2021-12-13",
  "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
  "elevatorId": "01",
  "floorRoomList": [
    {
      "floor": "9",
      "roomNumber": "901"
    },
    {
      "floor": "6",
      "roomNumber": "602"
    }
  ]
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

8.5 呼叫电梯

边缘服务器发送呼叫电梯指令，控制电梯到指定楼层。

```
URL：http://[门禁机IP]:[端口]/elevator/call
Method：POST
Content-Type：application/json
```

说明

- 起始楼层是用户出发时乘坐电梯的楼层。
- 目标楼层是用户乘坐电梯将要达到的楼层。
- 当只给出起始楼层时，目标楼层为空，表示指派电梯到起始楼层等待用户乘用。
- 当给出起始楼层和目标楼层时，表示指派电梯到起始楼层接用户到目标楼层。
- 如果电梯ID字段为空，那么梯控可以使用默认电梯或随机指定电梯。

限制

调用该接口的每秒请求数（QPS）最大限制为100。

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-12-13	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-12-13`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。
elevatorId	String	否	01	电梯ID，电梯的唯一标识。如果电梯ID字段为空，那么梯控可以使用默认电梯或随机指定电梯。
startFloor	String	是	6	用户乘梯出发的楼层。
startRoom	String	否	601	用户在出发楼层的房间号。
targetFloor	String	否	9	用户乘梯到达的楼层。
targetRoom	String	否	902	用户在到达楼层的房间号。
targetDirection	Integer	否	1	乘坐电梯要去的方向，枚举如下： • 1：上行 • 2：下行 此参数针对电梯轿厢外呼场景，在不指定targetFloorName的时候使用。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
  "version": "2021-12-13",
  "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
  "elevatorId": "01",
  "startFloor": "1",
  "startRoom": "01",
  "targetFloor": 8,
  "targetRoom": "02"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

8.6 获取电梯信息

边缘服务器获取梯控的电梯ID和用户楼层房间信息。

```
URL: http://[门禁机IP]:[端口]/elevator/info/get
Method: POST
Content-Type: application/json
```

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-12-13	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-12-13`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
data	JSONObject		
• elevatorMode	String	” GROUP “	【必选参数】 SINGLE：单控；只能控制特定的电梯，多部电梯时需指定elevatorId GROUP：群控；多部电梯时呼叫电梯根据梯控内部算法进行就近呼叫，不能指定特定电梯 ALL：双模（单控和群控）；多部电梯即可进行指定电梯呼叫，也可根据梯控内部算法进行就近呼叫
• floorInfoList	JSONArray		
◦ elevatorId	String	01	【可选参数】 电梯设备唯一标识。
◦ elevatorName	String		【可选参数】 电梯设备别名
◦ isDualCardDoor	Bool		【必选参数】 True：贯通门 False：单开门
◦ floorInfo	JSONObject		【必选参数】 楼层信息

■ floor	String		【必选参数】 住户可见楼层
■ roomList	JSONArray		【可选参数】 住户可见房间列表
■ roomNumber	String		【可选参数】 住户可见房间

示例

请求示例

```
{
  "version": "2021-12-13",
  "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128"
}
```

正常返回示例

```
{
  "code": 0,
  "message": "success",
  "data": {
    "elevatorMode": "ALL",
    "floorInfoList": [
      {
        "elevatorId": "01",
        "elevatorName": "dongti",
        "isDualCardDoor": true,
        "floorInfo": [
          {
            "floor": "5",
            "roomList": [
              "501",
              "502"
            ]
          },
          {
            "floor": "6",
            "roomList": [
              "601",
              "602"
            ]
          }
        ]
      }
    ]
  }
}
```

9 人梯联动

> 注意
> 如果设备支持人梯联动功能，本小节涉及的接口需要全部实现。

9.1 同步用户电梯规则

边缘服务器同步用户电梯规则信息到梯控。

```
URL: http://[门禁机IP]:[端口]/elevator/rule/sync
Method: POST
Content-Type: application/json
```

说明

当数据超过200条，会被拆分为多次同步调用。每次同步最大200条，每次同步超时时间为10秒。

请求参数

名称	类型	是否必选	示例值	描述

version	String	是	2021-12-13	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-12-13`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。
rules	JSONArray	是		
• type	Integer	是	1	设备规则操作类型，枚举如下： • 1：增加 • 2：删除 • 3：更新 • 4：删除全部；清除设备端所有用户电梯规则
• ruleId	String	否		电梯设备规则ID
• elevatorId	String	否	01	电梯ID，电梯的唯一标识。
• userId	String	否	777342461	用户ID
• controlMode	String	否		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
• startFloor	String	否		触发联动场景的设备所在楼层
• startRoom	String	否		触发联动场景的设备所在楼层的房间号
• targetFloorRoomList	JSONArray			用户拥有权限的楼层房号列表。
◦ floor	String	否	8	用户拥有权限的楼层。 如果存在贯通门的情况，使用房号区分不同的电梯门。如果不存在贯通门，不传房间号。
◦ roomNumber	String	否	01	用户拥有权限的房间号。
• 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
• effectiveTime	String	否	1589472000000	用户电梯规则生效的时间。以毫秒为单位，需要使用UTC时间。
• expiryTime	String	否	1905004800000	用户电梯规则过期的时间。以毫秒为单位，需要使用UTC时间。

说明：

此模式下需要传入参数：

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
  "version": "2021-12-13",
  "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
  "rules": [
    {
      "ruleId": "fa1a6-31a7-3**098",
      "userId": "ac1c2-31a7-3**066",
      "elevatorId": "01",
      "controlMode": "CALL_WITH_TARGET",
      "startFloor": "01",
      "startRoom": "601",
      "targetFloorRoomList": [
        {
          "floor": "9",
          "roomNumber": "01"
        }
      ],
      "delaySeconds": 10,
      "effectiveTime": "1589472000000",
      "expiryTime": "1905004800000"
    }
  ]
}
```

正常返回示例

```
{
    "code":0,
    "message":"success"
}
```

9.2 获取用户电梯规则

边缘服务器从梯控获取用户电梯规则。

```
URL：http://[门禁机IP]:[端口]/elevator/rule/get
Method：POST
Content-Type：application/json
```

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-12-13	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-12-13`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。

currentPage	Integer	否		当前页号。 如果不传此参数，表示全量查询。设备最多返回一次可查询的最大人脸数。
pageSize	Integer	否	200	指定返回结果中每页的最大人脸数量。 与currentPage参数一起使用。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
data	JSONObject		
• currentPage	Integer	1	必填，当前页号。
• pageCount	Integer	93	必填，总页数。
• pageSize	Integer	200	必填，每页的最大数量。
• userData	JSONArray		必填，子项目是否必填与《同步用户电梯规则》一致
◦ ruleId	String		电梯设备规则ID
◦ elevatorId	String	01	电梯ID，电梯的唯一标识。
◦ userId	String	777342461	用户ID
◦ controlMode	String	CALL_ONLY	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
◦ startFloor	String	1	触发联动场景的设备所在楼层
◦ startRoom	String	601	触发联动场景的设备所在楼层的房间号
◦ targetFloorRoomList	JSONArray		用户拥有权限的楼层房号列表。
▪ floor	String	8	用户拥有权限的楼层。 如果存在贯通门的情况，使用房号区分不同的电梯门。如果不存在贯通门，不传房间号。
▪ roomNumber	String	01	用户拥有权限的房间号。

○ delaySeconds	Integer		延迟执行的秒数，不填则表示立即执行，最大支持300秒
○ effectiveTime	String	1589472000000	用户电梯规则生效的时间。以毫秒为单位，需要使用UTC时间。
○ expiryTime	String	1905004800000	用户电梯规则过期的时间。以毫秒为单位，需要使用UTC时间。

示例

请求示例

```
{
  "version": "2021-12-13",
  "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
  "currentPage": "01",
  "pageSize": "01"
}
```

正常返回示例

```
{
  "code": 0,
  "message": "success",
  "data": {
    "currentPage": 1,
    "pageCount": 10,
    "pageSize": 2,
    "userData": [
      {
        "ruleId": "fa1a6-31a7-3**098",
        "userId": "ac1c2-31a7-3**066",
        "elevatorId": "01",
        "controlMode": "CALL_WITH_TARGET",
        "startFloor": "01",
        "startRoom": "601",
        "targetFloorRoomList": [
          {
            "floor": "9",
            "roomNumber": "01"
          }
        ],
        "delaySeconds": 10,
        "effectiveTime": "1589472000000",
        "expiryTime": "1905004800000"
      }
    ]
  }
}
```

9.3 上报人梯联动事件

梯控系统向边缘服务器上报人梯联动结果。

```
URL：http://[边缘服务器IP]:[端口]/elevator/rule/report
Method：POST
Content-Type：application/json
```

说明

当用户在电梯内刷门禁卡、蓝牙卡、扫描二维码、人脸识别后呼梯或释放楼层，执行完成后由梯控系统向边缘服务器发起事件上报请求。

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-12-13	API版本号，为日期形式：YYYY-MM-DD，最新版本为2021-12-13 。接口可能存在多个版本。

token	String	是	6f2a-***-128	有效连接标识。
ruleId	String	是		设备规则ID
elevatorId	String	否		电梯编号
userId	String	是	777342461	用户ID
authType	String	是		认证类型： "face"人脸 ” qrcode “二维码 ” card “卡 ” bluetooth “蓝牙
authContent	String	是		认证内容，由authType决定该内容。 authType为card时，该值填入cardId。 authType为bluetooth时，该值填入cardId。 authType为qrcode时，该值填入qrcode。 authType为face时，该值填入faceId。
controlMode	String	是		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
startFloor	String	否	1	用户乘梯出发的楼层。 说明：相对于最底层的相对楼层数。
startRoom	String	否	01	用户在出发楼层的房间号。
targetFloorRoomList	JSONArray			用户拥有权限的楼层房号列表。
• floor	String	否	9	用户拥有权限的楼层。 如果存在贯通门的情况，使用房号区分不同的电梯门。如果不存在贯通门，不传房间号。
• roomNumber	String	否	601	用户拥有权限的房间号。
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

controlMode为CALL_ONLY：

参数	是否必填
startFloor	必填

startRoom	根据《获取电梯信息》返回值决定， 对应floor有roomNumber时为必填，否则为不填。
targetFloorRoomList	不填

controlMode为CALL_WITH_TARGET：

参数	是否必填
startFloor	必填
startRoom	根据《获取电梯信息》返回值决定， 对应floor有roomNumber时为必填，否则为不填。
targetFloorRoomList	必填
floor	必填
roomNumber	根据《获取电梯信息》返回值决定， 对应floor有roomNumber时为必填，否则为不填

controlMode为RELEASE_ONLY：

参数	是否必填
startFloor	不填
startRoom	不填
targetFloorRoomList	必填
floor	必填
roomNumber	根据《获取电梯信息》返回值决定， 对应floor有roomNumber时为必填，否则为不填

controlMode为ACTIVATE_ONLY：

参数	是否必填
startFloor	不填
startRoom	不填
targetFloorRoomList	必填
floor	必填
roomNumber	根据《获取电梯信息》返回值决定， 对应floor有roomNumber时为必填，否则为不填

返回参数

名称	类型	示例值	描述

code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
  "version": "2021-12-13",
  "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
  "ruleId": "fa1a6-31a7-3**098",
  "elevatorId": "01",
  "userId": "ac1c2-31a7-3**066",
  "authType": "face",
  "authContent": "452f3570-4717-4a39-910b-d72542550001",
  "controlMode": "CALL_WITH_TARGET",
  "startFloor": "01",
  "startRoom": "601",
  "targetFloorRoomList": [
    {
      "floor": "9",
      "roomNumber": "01"
    }
  ],
  "delaySeconds": 10
}
```

正常返回示例

```
{
    "code":0,
    "message":"success",
}
```

9.4 设置人梯联动开关

边缘服务器同步人梯联动规则使能信息到梯控。

```
URL: http://[门禁机IP]:[端口]/elevator/rule/setAvailability
Method: POST
Content-Type: application/json
```

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-12-13	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-12-13`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。
availability	String	是	enable	” disable “：禁用 ” enable “：使能，设备出厂默认打开

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。

message	String	success	调用结果描述。

示例

请求示例

```
{
  "version": "2021-12-13",
  "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
  "availability": "enable"
}
```

正常返回示例

```
{
    "code":0,
    "message":"success"
}
```

9.5 获取人梯联动开关状态

边缘服务器获取人梯联动规则使能信息。

```
URL: http://[门禁机IP]:[端口]/elevator/rule/getAvailability
Method: POST
Content-Type: application/json
```

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-12-13	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-12-13`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
data	JSONObject		
• availability	String	enable	”disable“：禁用 ”enable“：使能，设备出厂默认打开

示例

请求示例

```
{
  "version": "2021-12-13",
  "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128"
}
```

正常返回示例

```
{
  "code": 0,
  "message": "success",
  "data": {
    "availability": "enable"
  }
}
```

错误码

附录查看更多错误码

附录

错误码

结果码	结果说明	
0	成功	success
1	识别认证错误	
2	保留字段	
3	保留字段	
4	token失效	device is offline
5	参数错误	
6	签名校验错误	verify sign error
7	设备ip地址未配置	device is illegal
99	其他错误	

3.5.1.2.3. 2.3 门禁设备与手机蓝牙对接协议

1 通讯方式简介

1.1 拓扑结构

星形拓扑结构：门口机作为主设备，APP 为从设备。主设备管理连接，主设备只能连接一个从设备；一个从设备只能连接一个主设备。

1.2 设备状态

蓝牙门口机作为从设备，共有以下三种状态：

待机状态（standy）	设备没有传输和发送数据，并且没有连接到任何设备
广播状态（advertiser）	周期性广播状态，等待被连接
从设备（slave）	作为从设备连接其他设备

1.2.1 状态以及连接过程

蓝牙门口机不用连接就可以广播数据（门口机名称和其他信息），手机 APP 扫描广播设备，蓝牙门口机接受到扫描后会发送一个扫描回应（扫描回应数据就是广播数据），APP 接收到扫描回应后发起连接，如果蓝牙门口机同意该连接请求，那么门口机和手机都会进入已连接状态，发起连接的设备手机变为主设备，接受连接请求的设备门口机变为从设备。

1.3 蓝牙门口机的连接参数

1.3.1 蓝牙门口机的广播事件

广播包发送是单向的，不需要任何的连接。广播包里可以包含特定的数据定义，最大 31 字节。

广播间隔：两次广播事件之间的最小的时间间隔，一般取值范围在 20ms~10.24s 之间，链路层会在每次广播时间期间产生一个随机广播延时时间加在广播时间里，这样来避免多设备之间的数据碰撞。

1.3.2 蓝牙连接参数

连接参数如下：

参数	说明
通道映射	连接使用的频道
调频增量	一个 5~16 之间的随机
连接间隔	1.25ms 的倍数，7.5ms~4s
监督超时	10ms 的倍数，100ms~32s
从机潜伏	0~499

1.3.3 BLE 连接事件

所有的通信都发生在两个设备的连接事件期间，连接事件周期的发生，按照连接参数指定的间隔联系，每个连接时间期间，主设备先发送，从设备会在 150us 之后做出回应，即使一个连接事件发生，但相互没有数据发送，两个设备仍然认为对方存在并保持活跃的连接。

潜伏值：从设备如果没有数据发送，允许跳过连接时间，潜伏值是允许跳过的最大的连接次数。从机不潜伏，在每个连接事件期间回应一个“空包”，除非有数据发送；从机潜伏使能， 从机可以跳过n 个连接时间，除非有数据发送才被唤醒。

1.3.4 终止连接

Master 和slave 都可以主动断开连接。

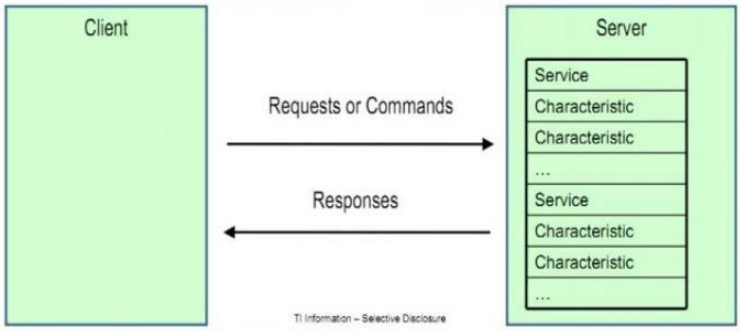
1.4 GATT 的client/service 架构

1.4.1 GATT 层

GATT 是 BLE 协议栈的通用属性配置文件层，规定了配置文件（profile）的结构。

1.4.2 GATT 的 Profile 层次结构

Profile 通常由一个或者多个服务（service）组成。每个 service 包含一个或多个特征值characteristic 特征值。每个具体的 characteristic 特征值才是 BLE 通信的主体，characteristic 特征值可以理解为标签，通过这个标签可以获取和写入想要的内容。



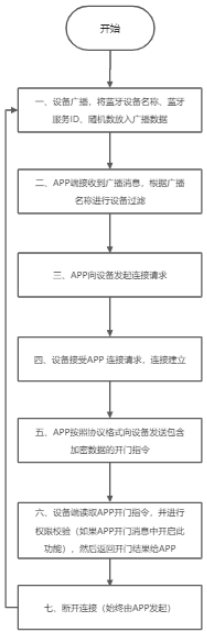
1.4.3 characteristic 属性

characteristic 的属性有以下几种：read，constant，write，write_no_response，notify，indicate，authenticated_read，authenticated_write，reliable_write。

2 通讯协议

2.1 手机app 与蓝牙门口机的交互过程

一旦建立连接后，门口机不会主动断开连接，需由 APP 端主动断开。



2.2 通讯参数设置

BLE 4.0 协议为低功耗蓝牙规范，但暂时不考虑功耗问题，为了使蓝牙能够快速响应并处理传输数据，将连接间隔事件时间设置为接近最短时间，将扫描和广播的频率提高但是连接的两个设备都会以高能耗运行。

2.2.1 蓝牙连接参数

参数	说明（后期根据实际测试情况调整）
通道映射	7: Advertise on all channels
调频增量	默认
有连接广播间隔	Min: 20ms (0.625ms*0x20) Max: 20ms (0.625ms*0x20)
无连接广播间隔	Min: 100ms (0.625ms*0xa0) Max: 100ms (0.625ms*0xa0)
扫描间隔	APP 根据情况自定义
扫描窗口	APP 根据情况自定义
连接间隔	Min: 7.5ms (1.25ms*6) Max: 12.5ms (1.25ms*10)
监督超时	100ms
从机潜伏	0

2.2.2 广播数据格式

蓝牙广播包数据为一串最大长度不超过 31 字节的数据，标准格式如下：

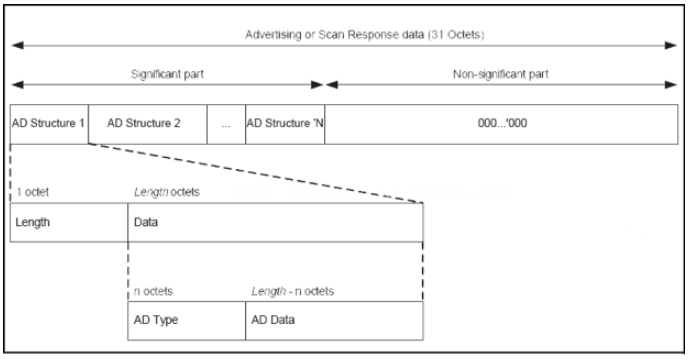


Figure 11.1: Advertising and Scan Response data format

为了APP 能够找到设备，广播数据中加入了蓝牙设备名称、2 字节的incomplete ServiceUUID。

具体格式如下(各字段含义见 2.2.3 节 广播数据格式说明):

字段含义	ASCII 码值	字节含义	字节序列索引
蓝牙广播包头	0x02	长度(字节)	0
	0x01	Flags	1
	0x06	广播标志头	2
蓝牙设备名称	0x11	长度(字节)	3
	0x09	Complete Local Name	4
	0x53	‘S’	5
	0x52	‘R’	6
	0x45	‘E’	7
	0x31	‘1’	8
	0x30	‘0’	9
	0x30	‘0’	10
	0x31	‘1’	11
	0x30	‘0’	12
	0x30	‘0’	13
	0x30	‘0’	14
	0x31	‘1’	15
	0x30	‘0’	16
	0x30	‘0’	17
	0x30	‘0’	18

	0x30	‘0’	19
	0x31	‘1’	20
服务ID	0x03	长度(字节)	21
	0x02	Incomplete List of 16-bit Service Class UUIDs	22
	0xff	UUID	23
	0xfa		24
自定义数据	0x05	长度(字节)	25
	0xff	Manufacturer Specific Data	26
	..	厂商自定义数据： 4 字节随机数。 每次新的连接前都会更新随机数。	27
	..		28
	..		29
	..		30
广播包总大小：			31 字节

2.2.3 广播数据格式说明

1. 具体蓝牙相关标志的细节请参考： https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile

2. 蓝牙设备名称

规则(一共 16 字节长度):

- 门口机模式：

SRE+期号(1 字节)+楼栋号(3 字节)+单元号(4 字节)+编号(2 字节)+层号(3 字节)

编号：主门口机使用 00 表示、从门口机从 01 开始(01,02 加 1 递增)

层号：分正楼层和负楼层(第一个字符为 0 表示正楼层,为 1 表示负楼层)

例如默认的1 期1 幢1 单元，放置在地上一层的主门口机的广播名称为：SRE1001000100001，放置在地下一层则为：SRE1001000100101

- 围墙机模式：

SRE+期号(1 字节)+0000000+编号(2 字节)+层号(3 字节)

围墙机模式下的楼栋号、单元号使用 0000000 代替，剩下的期号、编号、层号规则与门口机模式一致。

3. 服务 ID

广播中的 ID 是不完整的 16bit service UUID。

4. 自定义数据

厂商自定义数据，此处存放的是由时间加密因子计算出的 4 字节随机数，用于与双方约定的 16 字节固定秘钥生成随机秘钥，每次链接断开后都会更新 4 字节随机数。

2.2.4 服务和特征值说明

蓝牙门口机作为从机（服务器），手机作为主机（客户端）。蓝牙门口机新增一个服务（service），在此服务中添加characteristic 特征值，作用及命名如下。

特征值名称	长度	属性	作用
发送指令 (send command)	n	write	手机 APP 将要发送的指令报文分包写入此特征值， 每包小于等于下文 “传输协议” 中约定的字节数。

返回指令 (return command)	n	read	门口机将返回操作的内容以及状态报文写入此特征值，并通过写的方式发送给手机 APP。

2.2.5 设备蓝牙的UUID

名称	不完整值	完整值
Service UUID	0xFFFA	f6ecfffa-bda1-46ec-a43a-6d86de88561d
Write Characteristics UUID	0xFFA7	af20ffa7-2518-4998-9af7-af42540731b3
Read Characteristics UUID	0xFFA8	af20ffa8-2518-4998-9af7-af42540731b4

注：上表中的 read、write 均是从设备端蓝牙芯片来看的。

2.3 传输协议

2.3.1 APP->门口机

帧头 0X24 1 BYTE	返回结果类型 1 BYTE	权限有效长度 1 BYTE	权限内容 n BYTE	累加和 checksum 1 BYTE

说明：

1. 返回结果类型：

- 0x00：表示门口机收到 APP 开门指令后，将会进行数据格式+权限内容校验，并将校验之后的结果返回给手机 APP（校验通过说明蓝牙开门成功）。
- 0x01：表示门口机收到 APP 开门指令后，将只校验数据格式，并不关心权限内容，如果数据格式正确则返回开门成功消息给手机 APP。

2. 权限有效长度：

加密后的权限内容长度。

3. 权限内容：

标红色的权限内容字段为 AES 加密目标，用于存放开门卡号等信息。此处 nByte 表示加密后的数据长度。n 取值：0~n，暂定最大 32 字节，如果超过 32 字节则需要双方重新约定。如果加密前权限长度小于等于 15 字节，则此处加密后的长度为 16 字节；如果是大于 15 字节，小于等于 31 字节，则此处加密后的长度为 32 字节。因此最大的权限长度为 31 字节。

如果 APP 端传递的不是开门卡号，则需要另行商定。

4. 累加和：

校验位采用累加和校验，从帧头开始到权限内容最后一个字节的累加和对 256 取余，计算的是加密之后的累积和。

2.3.2 门口机->APP

帧头 0X24 1 BYTE	保留 2 BYTE	开门结果反馈 1 BYTE	累加和 checksum 1 BYTE

说明：

返回给 APP 的开门响应数据不会被加密。

开门结果反馈的状态码列表如下：

返回码	中文描述	英文描述
0x01	校验错误	CS_ERROR
0x02	帧格式不正确	FRAMEFORMAT_ERROR
0x03	认证失败	NO_PERMISSION_ERROR
成功信息		
0x00	成功	OK

2.4 加密

2.4.1 秘钥说明

为确保安全性，需要保证每个对接项目的 APP 不可相互开门，此次定制的门口机程序，只能和本地门口机程序对应的 APP 通讯开门。

因此将 16 字节秘钥按照如下约定生成：

定制单编号：如：DZP20200117037

长度为 14 字节，后面填 0 扩充至 16 字节：DZP2020011703700

转换为 ASCII 码：

N[16]={0x44, 0x5A, 0x50, 0x32, 0x30, 0x32, 0x30, 0x30, 0x31, 0x31, 0x37, 0x30, 0x33, 0x37, 0x30, 0x30};

在开发APP 前需获取本次对接时的定制单编号！

2.4.2 通信帧加密

采用 128 位密钥的 AES 加密 CBC 模式（初始化向量：“1234567890abcdef”，填充模式为zeropadding），只对协议格式里面DATA 数据区加密，密钥的生成如下：

1.设备存储一个固定密钥（初始密钥）：

N[16]={0x44, 0x5A, 0x50, 0x32, 0x30, 0x32, 0x30, 0x30, 0x31, 0x31, 0x37, 0x30, 0x33, 0x37, 0x30, 0x30};

2.生成一个 4 字节的随机数组

R[4]={0x45,0x18,0x9F,0x5C};



3.如上图所示，按照间隔 4 的大小，将 1、5、9、13 字节固定秘钥与随机数组 1 字节进行与操作，得到新的 1、5、9、13 字节动态秘钥；其他字节也按照上图所示进行加密，最终得到用于实际 AES128 加解密处理的 16 字节动态秘钥。

说明：“加”操作超出 0xff 取余；随机数在每次断开连接后会重新生成，所以每次操作的密钥都不相同。

举例：

固定秘钥：

N[16]={0x44, 0x5A, 0x50, 0x32, 0x30, 0x32, 0x30, 0x30, 0x31, 0x31, 0x37, 0x30, 0x33, 0x37, 0x30, 0x30};

随机数：

{0x45,0x18,0x9F,0x5C};

则得到的新的密钥为：

{0x44, 0x72, 0xDF, 0x6E, 0x00, 0x4A, 0xBF, 0x6C, 0x01, 0x49, 0xBF, 0x6C, 0x01, 0x4F, 0xBF, 0x6C}

转换代码 ：

```
for(i=0; i<4; i++) {
szDynamicKey[4*i] = gDefaultAesKey[4*i]&gRandNum[0];
szDynamicKey[4*i+1] = gDefaultAesKey[4*i+1] + gRandNum[1];
szDynamicKey[4*i+2] = gDefaultAesKey[4*i+2] | gRandNum[2];
szDynamicKey[4*i+3] = gDefaultAesKey[4*i+3] ^ gRandNum[3];
}
```

2.4.3 秘钥一致性

本章节通过举例方式列出几种待加密数据的加密结果，供三方 APP 开发时参考，确保双方使用”相同待加密数据、相同固定秘钥、相同 4 字节随机数”加密得到的密文一致。

数据长度	待加密数据	固定秘钥	4 字节 随机数	随机秘钥	加密后密文 (16 进制 byte array)

14	"12345601010702"	DZP2020011703700	0x45, 0x18, 0x9F, 0x5C	'0x44', '0x72', '0xdf', '0x6e', '0x0', '0x4a', '0xbf', '0x6c', '0x1', '0x49', '0xbf', '0x6c', '0x1', '0x4f', '0xbf', '0x6c'	'0xf7', '0x2f', '0x0', '0xed', '0xfc', '0x2a', '0x83', '0xcd', '0xc9', '0x6c', '0x5', '0xbc', '0x95', '0x64', '0xa6', '0x75'
16	"1234567812345678"	DZP2020011703700	0x45, 0x18, 0x9F, 0x5C	'0x44', '0x72', '0xdf', '0x6e', '0x0', '0x4a', '0xbf', '0x6c', '0x1', '0x49', '0xbf', '0x6c', '0x1', '0x4f', '0xbf', '0x6c'	'0xce', '0x53', '0x38', '0x7f', '0x11', '0x5d', '0xf4', '0xaa', '0xed', '0x9d', '0x1c', '0xe8', '0x74', '0x5d', '0x39', '0xe1', '0xeb', '0xdc', '0x81', '0x62', '0x35', '0x38', '0x78', '0xee', '0x34', '0x11', '0xd0', '0xfa', '0xcd', '0x39', '0x98', '0xa'
31	"12345678123456781234567812345678"	DZP2020011703700	0x45, 0x18, 0x9F, 0x5C	'0x44', '0x72', '0xdf', '0x6e', '0x0', '0x4a', '0xbf', '0x6c', '0x1', '0x49', '0xbf', '0x6c', '0x1', '0x4f', '0xbf', '0x6c'	'0xce', '0x53', '0x38', '0x7f', '0x11', '0x5d', '0xf4', '0xaa', '0xed', '0x9d', '0x1c', '0xe8', '0x74', '0x5d', '0x39', '0xe1', '0x45', '0x3b', '0x2', '0xc0', '0x5a', '0x71', '0x35', '0x7a', '0xfa', '0x21', '0xf5', '0x14', '0xde', '0x43', '0x6f', '0xf9'

3.5.1.2.4. 2.4 可视对讲设备端SDK接口说明-[Android]

1 概述

1.1 编写目的

本文是可视对讲设备云对讲和呼手机功能SDK接口说明。

2. 介绍

通过集成对接SDK可以使门禁设备快速实现包括门禁机呼叫APP、APP调看门禁机、门禁机呼叫手机号等对讲功能。本文旨在描述该SDK的集成方式及主要业务代码。

3. SDK集成方法

3.1 导入所需文件

新建或打开已有工程，拷贝leephone.aar包到app/libs目录下.在app/build.gradle中添加aar包的引用，如图所示。

```
...
android {
  ...
    defaultConfig {
    ...
        ndk.abiFilters "armeabi-v7a", "arm64-v8a"
    }
}
...
repositories {
    flatDir {
        dirs 'libs' // aar dir
    }
}
...
dependencies {
 implementation(name:'leephone', ext:'aar')
  ...
}
```

3.2 在AndroidManifest.xml配置

支持的最低SDK版本为5.0:

<uses-sdk android:minSdkVersion="21" />

添加必要权限:

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

3.3 代码混淆配置

如果有做代码混淆，请确保以下类不加入混淆

```
-keep class com.evideo.voip.** { *; }
-keep class com.lee.phone.** { *; }
```

4. 接口列表

名称	类名	方法名
初始化	EVVoipManager	init(Context context, EVVoipManager.OnInitCallback callback)
反初始化	EVVoipManager	deInit(Context context)
登入边缘网关	EdgeBoxManager	loginEdgeBox(Context context, String edgeBoxIP, String displayName, String localIp, String localMac, EdgeBoxManager.OnLoginCallback callback)
登出边缘网关	EdgeBoxManager	logoutEdgeBox();
账号状态监听	EVVoipAccount	setAccountStateCallback(EVVoipAccount.AccountStateCallbackaccountStateCallback)
来电监听	EVVoipManager	setIncomingCallback(EVVoipManager.IncomingCallbackincomingCallback)
呼叫APP	EdgeBoxManager	callRoomID(String roomID, EVVoipCallParams params, EdgeBoxManager.OnCallRoomCallback callback)
接听	EVVoipCall	accept(EVVideoView display)
挂断	EVVoipCall	hangup()
通话状态监听	EVVoipCall	setCallStateCallback(EVVoipCall.CallStateCallbackcallStateCallback)
事件监听	EVVoipManager	setOnEventReceivedListener(EVVoipManager.OnEventReceived)
呼叫手机号	EVVoipManager	callPhoneNumber(String phoneNumber)

5. 主要业务流程

5.1 初始化

在进行对讲业务之前，需要先初始化该SDK，才能保证后续业务正常，后续的所有API调用都必须在SDK初始化成功的情况下执行。在应用的生命周期内，一般进行一次初始化就可以了。

```
// 初始化对讲SDK
EVVoipManager.init(getApplicationContext(), new EVVoipManager.OnInitCallback() {
    @Override
    public void complete() {
        // 对讲SDK初始化成功
    }
    @Override
    public void error(int code) {
        // 对讲SDK初始化失败，错误代码code
    }
});
```

当不再使用对讲业务时，可以调用deInit进行SDK反初始化。

```
// 反初始化对讲SDK
EVVoipManager.deInit(getApplicationContext());
```

监视开门事件

```
EVVoipManager.setOnEventReceivedListener(new EVVoipManager.OnEventReceived() {
    @Override
    public void callback(int eventCode, String fromSipNumber) {
        switch(eventCode) {
            case EVENT_CODE_UNLOCK_BY_APP://APP开门指令
                //执行开门动作
                break;
            case EVENT_CODE_UNLOCK_BY_PHONE://手机开门指令
                //执行开门动作
                break;
            default:
                break;
        }
    }
});
```

5.2 登入边缘服务器

在对讲SDK初始化成功之后，需要进行边缘网关的登入，edgeBoxIP为边缘服务器IP，localIp为本地IP，localMac为本地Mac地址，需要应用层下发，登入成功后可以得到EVVoipAccount对象。

```
private EVVoipAccount mEVVoipAccount;
EdgeBoxManager.loginEdgeBox(this, edgeBoxIP, localIp, localMac, new EdgeBoxManager.
OnLoginCallback() {
    @Override
    public void onSuccess(EVVoipAccount evVoipAccount) {
        mEVVoipAccount = evVoipAccount;
        // ...
    }
    @Override
    public void onFailure(int code, String message) {
    }
});
```

为了能够实时显示账号状态，本SDK提供 AccountStateCallback 接口实时监听当前账号状态。

```
mEVVoipAccount.setAccountStateCallback(new EVVoipAccount.AccountStateCallback() {
    @Override
    public void onState(EVVoipAccount.AccountState state) {
        /* 获取该账号的状态
         * AccountState.ONLINE 在线
         * AccountState.OFFLINE 离线
         * AccountState.LOGINPROCESS 登入中
         * AccountState.NONE 无账号、未登入
         */
    }
});
```

当不再使用对讲业务，可以通过以下接口进行登出操作。

```
mEVVoipAccount.logoutEdgeBox();
```

5.3 门禁机呼叫APP

设备端呼叫APP，可调用 callRoomID 接口，并传入roomID进行呼叫，roomID为呼叫号。

```
// 初始化呼叫配置参数集
EVVoipCallParams params = new EVVoipCallParams();
params.setDisplay(mEVVideoView);
try {
    // 开始呼叫
    evCall = EVVoipManager.callRoomID(roomID, params);
} catch (EVVoipException e) {
    e.printStackTrace();
}
```

呼叫接口会返回的EVVoipCall对象，可设置回调监听通话状态。

```
evCall.setCallStateCallback(new EVVoipCall.CallStateCallback() {
    @Override
    public void onState(CallState state, EndReason reason) {
            if (CallState.INCOMING == state) {
            } else if (CallState.OUTGOING == state) {
            } else if (CallState.CONNECTED == state) {
            } else if (CallState.END == state) {
            if (reason != null && EndReason.NONE != reason) {
                /*
                 * NONE - 无错误，正常状态
                 * BUSY - 对方忙
                 * TIMEOUT - 呼叫超时
                 * NOTFOUND - 对方不在线
                 * REJECTED - 一方拒接
                 * UNKNOW - 未知错误
                 */
            }
        }
    }
});
```

设备端可以主动挂断通话。

```
evCall.hangup();
```

5.4 支持APP调看门禁机

为了使设备端能够响应APP的调看请求，需要注册一个来电回调监听接口。

```
EVVoipManager.setIncomingCallback(new EVVoipManager.IncomingCallback() {
    @Override
    public void inComing(EVVoipCall evCall) {
        // evCall表示当前来电的通话
    }
});
```

当收到APP调看请求回调时，会得到一个EVVoipCall，表示当前的通话。

```
@Override
public void inComing(EVVoipCall evCall) {
  // 保存当前通话
  // 直接接听或跳转到通话界面
}
```

为了响应APP的监控请求，需要提供一个EVVideoView作为视频容器，如果不提供则为音频通话。

```
// 初始化视频绘制容器，EVVideoView是一个android.app.Fragment，目前只支持在xml布局文件中配置
mEVVideoView= (EVVideoView) getFragmentManager().findFragmentById(R.id.display_view);
try {
    // 响应监控请求
    evCall.accept(mEVVideoView);
} catch (EVVoipException e) {
    e.printStackTrace();
}
```

为了能够实时获取当前的通话状态，可以在EVVoipCall对象上设置一个通话状态回调接口。

```
evCall.setCallStateCallback(new EVVoipCall.CallStateCallback() {
    @Override
    public void onState(CallState state, EndReason reason) {
        // 参考门禁机呼叫APP示例
    }
});
```

设备端可以主动挂断通话。

```
evCall.hangup();
```

5.5 音视频扩展

SDK默认是自己管理音视频的，如果开发者有自己的音视频数据流，需要调用以下接口接入。

视频流接入

```
// 使能外部视频数据流接入
EVVoipManager.enableCameraWrapper(width, height);
// 注入外部视频数据流
EVVoipManager.putImage(data);
```

音频流接入

```
// 使能外部音频数据流接入
EvideoVoipRecorder.setEvideoVoipRecorderListener(
samplerate, samples, new RecorderListener() {
    @Override
    public void startRecording() {
        // 启动外部音频流
    }
    @Override
    public void stopRecording() {
        // 停止外部音频流
    }
});
// 注入外部音频数据流
if (EvideoVoipRecorder.isRecording())
EvideoVoipRecorder.putStream(buff, size, framCount);
```

5.6 门禁机呼叫手机号

Step1：获取IPPBX的账号信息并调用 loginIppbx 登入IPPBX，接口返回EVVoipAccount对象。

```
private EVVoipAccount mEVVoipAccount;
try {
    mEVVoipAccount = EVVoipManager.loginIppbx(String sipnum, String password,
    String displayName, String domain, int port);
} catch (EVVoipException e) {
    e.printStackTrace();
}
// 当不再需要该账号时进行登出
mEVVoipAccount.logoutIppbx();
```

Step2：监听账号登入状态。

```
mEVVoipAccount.setAccountStateCallback(new EVVoipAccount.AccountStateCallback() {
    @Override
    public void onState(EVVoipAccount.AccountState state) {
        // 参考边缘网关账号登入
    }
});
```

Step3：账号登入成功后，呼叫手机号，手机号需通过应用层下发。

```
try {
    evCall = EVVoipManager.callPhoneNumber(String phoneNumber)
} catch (EVVoipException e) {
    e.printStackTrace();
}
```

Step4：监听通话状态。

```
evCall.setCallStateCallback(new EVVoipCall.CallStateCallback() {
    @Override
    public void onState(CallState state, EndReason reason) {
        // 参考边缘网关通话状态监听
    }
});
```

Step5：挂掉电话，设备端可以主动挂断通话。

```
evCall.hangup();
```

5.7 门禁机上报开门事件

APP发送开门指令给门禁机，门禁机开门后，上报开门事件。

```
mIppbxEVVoipCall = EVVoipManager.callPhoneNumber(phoneNumber);
EVVoipManager.setOnEventReceivedListener(new EVVoipManager.OnEventReceived() {
    @Override
    public void callback(int eventCode, String number) {
        switch (eventCode) {
            case EVVoipManager.EVENT_CODE_UNLOCK_BY_APP: //云对讲APP开门
                EdgeBoxManager.reportOpenDoor(EdgeBoxManager.EVENT_CODE_CLOUD_UNLOCK, number);
                break;
            case EVVoipManager.EVENT_CODE_UNLOCK_BY_PHONE: //语音固话开门
                EdgeBoxManager.reportOpenDoor(EdgeBoxManager.EVENT_CODE_FIXED_UNLOCK, number);
                break;
        }
    }
});
```

3.5.1.2.5. 2.5 可视对讲SIP服务器接入协议

1. 概述

1.1 编写目的

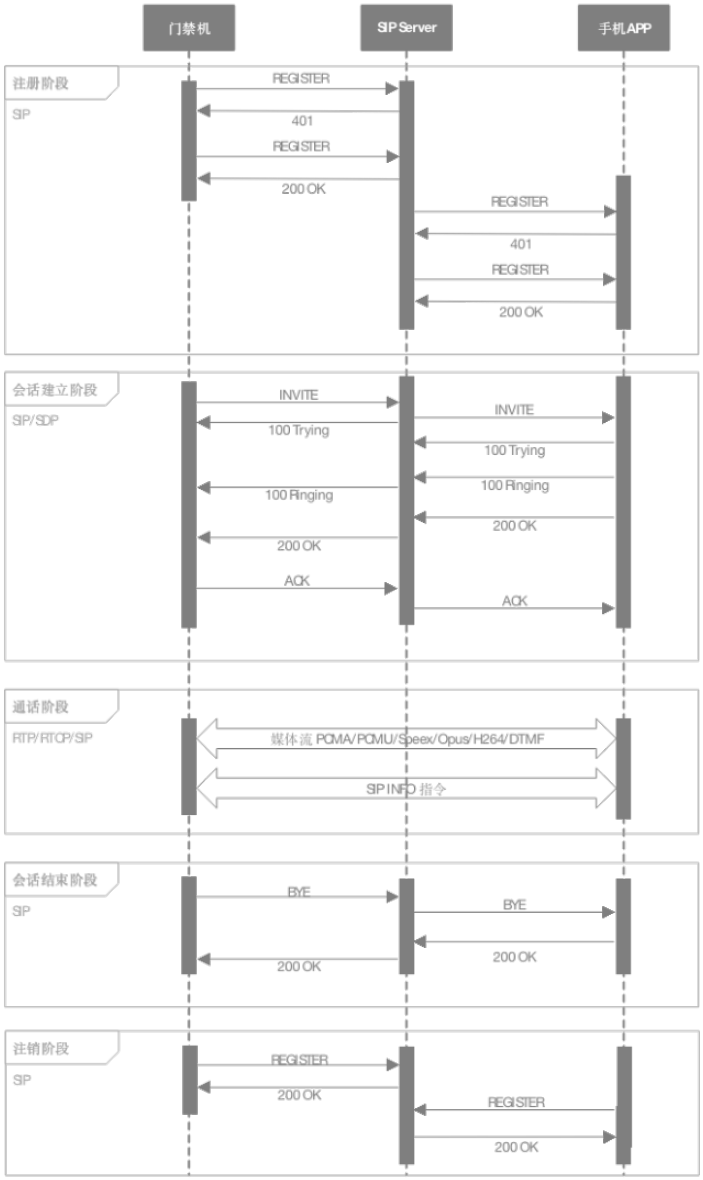
本文是可视对讲设备云对讲功能SIP服务接入协议说明。

1.2 名词解释

名词	解释
IoT	物联网

2. 介绍

本文档旨在描述SIP对讲服务接入规范，云对讲采用标准SIP协议，其主要分为注册、会话建立、通话、会话结束、注销等几个阶段。涉及SIP、SDP、RTP、RTCP、PCMA、PCMU、Speex、Opus、H264、DTMF等协议。



3. 协议规范

3.1 SIP

SIP(Session Initiation Protocol)是一个应用层的信令控制协议。用于创建、修改和释放一个或多个参与者的会话。这些会话可以是Internet多媒体会议、IP电话或多媒体分发。会话的参与者可以通过组播（multicast）、网状单播（unicast）或两者的混合体进行通信。SIP协议详见《RFC 3261 SIP:Session Initiation Protocol》。SIP有很多开源库，如PJSIP、osip2、opensipstack等，可以根据自身平台特性选择。

如下是注册阶段的示例报文:

```
09:57:14.634407 IP 192.168.1.233.62039 > 47.112.101.107.6050: UDP, length 605
E..y#...@..<..../pek.W...e..REGISTER sip:47.112.101.107 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.233:62039;branch=z9hG4bK.7jiHBl8Sm;rport
From: <sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107>;tag=nPfvQ6pKa
To: sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107
CSeq: 20 REGISTER
Call-ID: 7ay7mqUOym
Max-Forwards: 70
Supported: replaces, outbound
Accept: application/sdp
Accept: text/plain
Accept: application/vnd.gsma.rcs-ft-http+xml
Contact: <sip:TU9e8mJvLJLcSsy9dmDKuc@192.168.1.233:62039;transport=udp>;+sip.instance="<urn:uuid:418ee786-4cf0-435a-b42c-5ab6d06f8b88>"
Expires: 600
User-Agent: Linphone Desktop/4.1.1 (belle-sip/1.6.3)

09:57:14.658069 IP 47.112.101.107.6050 > 192.168.1.233.62039: UDP, length 525
E..).&..2..2/pek.......W..._SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 192.168.1.233:62039;branch=z9hG4bK.7jiHBl8Sm;received=121.204.13.62;rport=62039
From: <sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107>;tag=nPfvQ6pKa
To: sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107;tag=as4fef1d8e
Call-ID: 7ay7mqUOym
CSeq: 20 REGISTER
Server: JustPBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH,MESSAGE
Supported: replaces, timer
WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="59320d53"
Content-Length: 0

09:57:14.670190 IP 192.168.1.233.62039 > 47.112.101.107.6050: UDP, length 786
E....k..@.>...../pek.W......REGISTER sip:47.112.101.107 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.233:62039;branch=z9hG4bK.awvGndk3C;rport
From: <sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107>;tag=nPfvQ6pKa
To: sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107
CSeq: 21 REGISTER
Call-ID: 7ay7mqUOym
Max-Forwards: 70
Supported: replaces, outbound
Accept: application/sdp
Accept: text/plain
Accept: application/vnd.gsma.rcs-ft-http+xml
Contact: <sip:TU9e8mJvLJLcSsy9dmDKuc@121.204.13.62:62039;transport=udp>;+sip.instance="<urn:uuid:418ee786-4cf0-435a-b42c-5ab6d06f8b88>"
Expires: 600
User-Agent: Linphone Desktop/4.1.1 (belle-sip/1.6.3)
Authorization: Digest realm="asterisk", nonce="59320d53", algorithm=MD5, username="TU9e8mJvLJLcSsy9dmDKuc", uri="sip:47.112.101.107", re
sponse="414132ffca95be7131b756d4890b1fc4"

09:57:14.691681 IP 47.112.101.107.6050 > 192.168.1.233.62039: UDP, length 575
E..[./..2.../pek.......W.G.gSIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.233:62039;branch=z9hG4bK.awvGndk3C;received=121.204.13.62;rport=62039
From: <sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107>;tag=nPfvQ6pKa
To: sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107;tag=as4fef1d8e
Call-ID: 7ay7mqUOym
CSeq: 21 REGISTER
Server: JustPBX
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH,
MESSAGE
Supported: replaces, timer
Expires: 600
Contact: <sip:TU9e8mJvLJLcSsy9dmDKuc@121.204.13.62:62039;transport=udp>;expires=600
Date: Thu, 16 Jan 2020 01:57:14 GMT
Content-Length: 0
```

3.2 SDP

SDP(Session Description Protocol)是一种会话描述格式 — 它不属于传输协议 — 它只使用不同的适当的传输协议，包括会话通知协议（SAP）、会话初始协议（SIP）、实时流协议（RTSP）、MIME 扩展协议的电子邮件以及超文本传输协议（HTTP）。SDP协议是基于文本的协议，这样就能保证协议的可扩展性比较强，这样就使其具有广泛的应用范围。SDP不支持会话内容或媒体编码的协商，所以在流媒体中只用来描述媒体信息。SDP协议详见《RFC 4566 - SDP: Session Description Protocol》

如下是INVITE携带SDP报文示例：

```
10:04:41.223197 IP 192.168.1.233.62039 > 47.112.101.107.6050: UDP, bad length 2114 > 1472
E..... .@.T...../pek.W...J,.INVITE sip:M28PLP9MSvu6w0Vc13HPnC6@47.112.101.107 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.233:62039;branch=z9hG4bK.8aOxITrEA;rport
From: <sip:TU9e8mJvLJLcSsy9dmDKuc@47.112.101.107>;tag=7B13LTdJn
To: sip:M28PLP9MSvu6w0Vc13HPnC6@47.112.101.107
CSeq: 20 INVITE
Call-ID: hpgH62gK2G
Max-Forwards: 70
Supported: replaces, outbound
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO,UPDATE
Content-Type: application/sdp
Content-Length: 1449
Contact: <sip:TU9e8mJvLJLcSsy9dmDKuc@121.204.13.62:62039;transport=udp>;+sip.instance="<urn:uuid:418ee786-4cf0-435a-b42c-5ab6d06f8b88>"
User-Agent: Linphone Desktop/4.1.1 (belle-sip/1.6.3)

v=0
o=TU9e8mJvLJLcSsy9dmDKuc 2846 1117 IN IP4 192.168.1.233
s=Talk
c=IN IP4 192.168.1.233
t=0 0
a=ice-pwd:305b42a8561a2b3f06a158d7
a=ice-ufrag:a60560d1
a=rtcp-xr:rcvr-rtt=all:10000 stat-summary=loss,dup,jitt,TTL voip-metrics
m=audio 54480 RTP/AVPF 96 97 98 0 8 101 99 100
c=IN IP4 121.204.13.62
a=rtpmap:96 opus/48000/2
a=fmtp:96 useinbandfec=1
a=rtpmap:97 speex/16000
a=fmtp:97 vbr=on
a=rtpmap:98 speex/8000
a=fmtp:98 vbr=on
a=rtpmap:101 telephone-event/48000
a=rtpmap:99 telephone-event/16000
a=rtpmap:100 telephone-event/8000
a=rtcp:51670
a=candidate:1 1 UDP 2130706431 192.168.1.233 54480 typ host
a=candidate:1 2 UDP 2130706430 192.168.1.233 51670 typ host
a=candidate:2 1 UDP 1694498815 121.204.13.62 54480 typ srflx raddr 192.168.1.233 rport 54480
a=candidate:2 2 UDP 169449881
```

3.3 RTP/RTCP

RTP(Real-time Transport Protocol)即实时传输协议，是一个网络传输协议，它是由IETF的多媒体传输工作小组1996年在RFC 1889中公布的。RTP协议详细说明了在互联网上传递音频和视频的标准数据包格式。它一开始被设计为一个多播协议，但后来被用在很多单播应用中。RTP协议常用于流媒体系统（配合RTSP协议），视频会议和一键通（Push to Talk）系统（配合H.323或SIP），使它成为IP电话产业的技术基础。RTP协议和RTP控制协议RTCP一起使用，而且它是创建在UDP协议上的。

RTP标准定义了两个子协议，RTP和RTCP。数据传输协议RTP，用于实时传输数据。该协议提供的信息包括：时间戳（用于同步）、序列号（用于丢包和重排序检测）、以及负载格式（用于说明数据的编码格式）。控制协议RTCP，用于QoS反馈和同步媒体流。相对于RTP来说，RTCP所占的带宽非常小，通常只有5%。RTP/RTCP协议详见《RFC 3550 - RTP: A Transport Protocol for Real-Time Applications》

3.4 音频

目前支持PCMA、PCMU、Speex 8kHz/16kHz、Opus音频格式

PCMA/PCMU

PCMA/PCMU协议接入详见《RFC 3551 - RTP Profile for Audio and Video Conferences with Minimal Control》

Speex

Speex是一个声音编码格式，目标是用于网络电话、线上广播使用的语音编码，基于CELP开发，Speex宣称可以免费使用，以BSD授权条款开放原始码。 Speex的开发者将这个格式视为Vorbis的补充。 Speex是一种有损格式，这意味著使用此格式的音讯，品质将会永久性地降低以减少档案的大小。接入详见《RFC 5574 - RTP Payload Format for the Speex Codec》Speex支持8kHz和16kHz

Opus

Opus是一个有损声音编码的格式，由Xiph.Org基金会开发，之后由网际网络工程任务组进行标准化，目标是希望用单一格式包含声音和语音，取代Speex和Vorbis，且适用于网络上低延迟的即时声音传输，标准格式定义于RFC 6716档案。Opus格式是一个开放格式，使用上没有任何专利或限制。接入详见《RFC 7587 - RTP Payload Format for the Opus Speech and Audio Codec》

3.5 视频

目前支持H264视频格式

H264

H264是国际标准化组织（ISO）和国际电信联盟（ITU）共同提出的继MPEG4之后的新一代数字视频压缩格式。主要目标是：与其它现有的视频编码标准相比，在相同的带宽下提供更加优秀的图象质量。通过该标准，在同等图象质量下的压缩效率比以前的标准（MPEG2）提高了2倍左右。接入详见《RFC 3984 - RTP Payload Format for H.264 Video》

3.6 DTMF

DTMF(Dual Tone Multi Frequency)即双音多频，由高频群和低频群组成，高低频群各包含4个频率。一个高频信号和一个低频信号叠加组成一个组合信号，代表一个数字。DTMF信号有16个编码。利用DTMF信令可选择呼叫相应的对讲机。DTMF在对讲中用于发送开门指令（*键）。接入详见《RFC2833 - RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals》

4. 扩展协议

4.1 开锁协议

为了扩展功能，现在SIP基础上，通过SIP INFO方法（详见RFC 6086 - Session Initiation Protocol (SIP) INFO Method and Package Framework）来发送、接收自定义请求，由于目前SIP采用udp，可能会存在丢包，为了保证成功率，这里采用双包策略，即发送端同时发送2个请求（响应）包，接收端判断3秒内如果是同一个包（根据reqsn），则第二个包不做处理。请求方发送命令后，在3秒内未收到响应包，则认为请求失败，并返回给上层。为了尽量减少数据包大小，采用x-www-form-urlencoded格式。

- 公共头

需设置如下header

Content-Type:application/json

- 请求

参数	类型	必填	描述
req	string	是	请求命令，值为unlock
reqsn	int	是	请求序列号，1-999范围内循环自增，每次请求加1，双包策略中第二个包reqsn不自增

示例：

{

"req":"unlock",

"reqsn":100

}

- 响应

参数	类型	必填	描述
res	string	是	返回请求的命令，值为unlock
reqsn	int	是	返回请求的reqsn
status	Int	是	状态码，200成功，400以上失败

示例：

{

"res":"unlock",

"reqsn":100,

"status":200

}

5. 附录

《RFC 3261 - SIP: Session Initiation Protocol》 - https://tools.ietf.org/html/rfc3261

《RFC 4566 - SDP: Session Description Protocol》 - https://tools.ietf.org/html/rfc4566

《RFC 3550 - RTP: A Transport Protocol for Real-Time Applications》-https://tools.ietf.org/html/rfc3550

《RFC 3551 - RTP Profile for Audio and Video Conferences with Minimal Control》-https://tools.ietf.org/html/rfc3551

《RFC 5574 - RTP Payload Format for the Speex Codec》 - https://tools.ietf.org/html/rfc5574

《RFC 7587 - RTP Payload Format for the Opus Speech and Audio Codec》 -https://tools.ietf.org/html/rfc7587

《RFC 3984 - RTP Payload Format for H.264 Video》 - https://tools.ietf.org/html/rfc3984

《RFC 2833 - RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals》-https://tools.ietf.org/html/rfc2833

《RFC 6086 - Session Initiation Protocol (SIP) INFO Method and Package Framework》-https://tools.ietf.org/html/rfc6086

3.5.1.2.6. 2.6 人脸算法服务 API

本文档旨在定义与人脸算法服务交互的 API。

1、 方案介绍

人脸算法服为边缘服务器提供人脸算法支持，当云端下发人脸权限时，边缘服务器会通过预设的人脸算法获取权限对应的人脸特征值。

2、通讯协议

驱动和边缘人脸算法间通过 HTTP 协议进行数据通讯，边缘服务器作为 HTTP Client 人脸算法服务作为 HTTP Server。

3、接口定义

人脸算法服务应提供以下接口供边缘服务器调用：

3.1、可用检查

边缘服务器调用该接口检查人脸算法是否可用。

```
URL：http://人脸算法IP:[端口]/
Method：GET
```

请求参数：

无

示例：

```
curl -L -X GET 'http://人脸算法IP:[端口]/'
```

返回参数：

成功：返回 "Face Recognition Service"

失败：HTTP请求失败，无法访问该算法服务

3.2、检测照片上的人脸并提取特征值

边缘服务器发送照片到人脸算法服务，人脸算法服务检测照片上的人脸，当检测到人脸后提取、返回该照片的人脸特征值。

```
URL：http://人脸算法IP:[端口]/match_with_face_detect
Method：POST
Content-Type：multipart/form-data
```

请求参数：

key	value	描述
photo	照片文件	照片文件的大小不得超过 1M

示例：

```
curl -L -X POST 'http://人脸算法IP:[端口]/match_with_face_detect' \-F 'photo=@"/face.gif"'
```

返回参数：

参数	类型	解释
code	Integer	成功： 200 输入文件无效： 400 从照片提取特征值失败：500
message	String	成功： 'success' 失败： 描述错误原因
data	JsonArray	成功：返回人脸照片对应的特征值信息 失败： 空数组

示例：

```
{
  'code': 200,
  'message':'success',
  'data': [
    -0.009695335291326046,
    0.04982885345816612
  ]
}
```

3.3、仅提取照片中的人脸特征值

边缘服务器发送照片到人脸算法服务，人脸算法服务直接从照片中提取人脸特征值，随后将提取到的人脸特征值返回。

```
URL: http://人脸算法IP:[端口]/match
Method: POST
Content-Type: multipart/form-data
```

请求参数：

key	value	描述
photo	照片文件	照片文件的大小不得超过 1M

示例：

```
curl -L -X POST 'http://人脸算法IP:[端口]/match' \-F 'photo=@"/face.gif"'
```

返回参数：

参数	类型	解释
code	Integer	成功： 200 输入文件无效： 400 从照片提取特征值失败：500
message	String	成功： 'success' 失败： 描述错误原因
data	JsonArray	成功：返回人脸照片对应的特征值信息 失败： 空数组

示例：

```
{
  'code': 200,
  'message':'success',
  'data': [
    -0.009695335291326046,
    0.04982885345816612
  ]
}
```

3.5.1.3. 3 门禁控制器对接方案

3.5.1.3.1. 3.1 门禁控制器对接方案介绍

1. 概述

1.1 编写目的

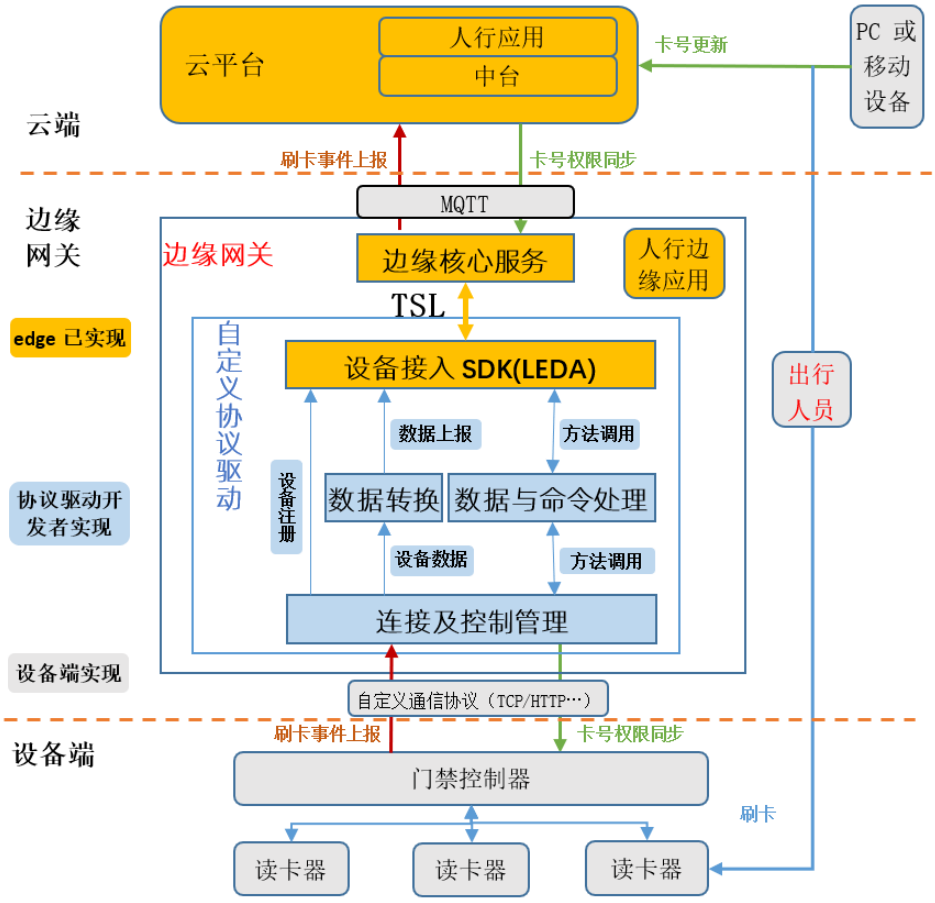
用于指导人行设备厂家的技术人员，基于自定义协议进行方案评估和开发工作，从而实现人行设备接入。目前自定义协议的人行设备主要是**刷卡门禁控制器**。本文档同时指导自定义协议驱动的开发流程。

1.2名词解释

名词	解释

Link IoT Edge	物联网边缘计算产品（Link IoT Edge，简称LE），即阿里云物联网平台（IoT）中的边缘计算产品。提供安全可靠的数据计算能力，可供本地处理设备数据，减少上传云端的成本。
网关	运行Link IoT Edge软件的计算设备统称为边缘网关或边缘服务器，简称网关。
子设备	指通过一定的协议或接口接入到Link IoT Edge网关上的设备（即设备接入到网关后称为子设备），网关代理该子设备与云端进行通信。
驱动	Link IoT Edge中的设备接入模块称为驱动（driver）或设备接入驱动。所有连接到Link IoT Edge的设备都需要通过驱动实现接入。

2．系统架构图



图解说明：

①云端：即云平台以及使用人行应用和中台服务接口开发的上层应用软件，应用软件运行在PC或移动设备上，进行卡号查询及权限更新等操作。

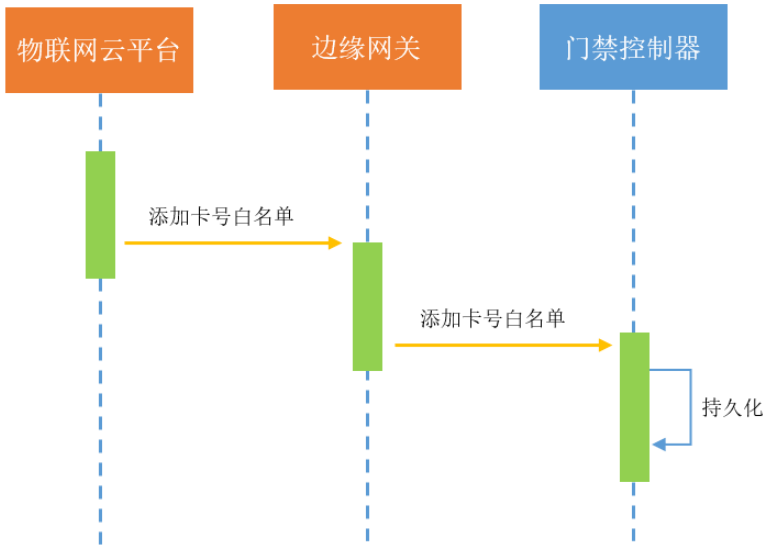
②边缘网关：运行LE组件及人行边缘应用，LE组件内运行自定义协议驱动，驱动内包含设备接入SDK（LEDA）和开发者实现两个部分。驱动负责设备的连接管理、事件上报及服务调用。

③设备端：即门禁控制器和读卡器等硬件设备。设备供应商应提供设备通信协议或设备访问SDK，供驱动访问或控制设备使用。设备供应商还应提供硬件配置工具，可以进行设备初始化、设备参数配置及故障定位等功能。

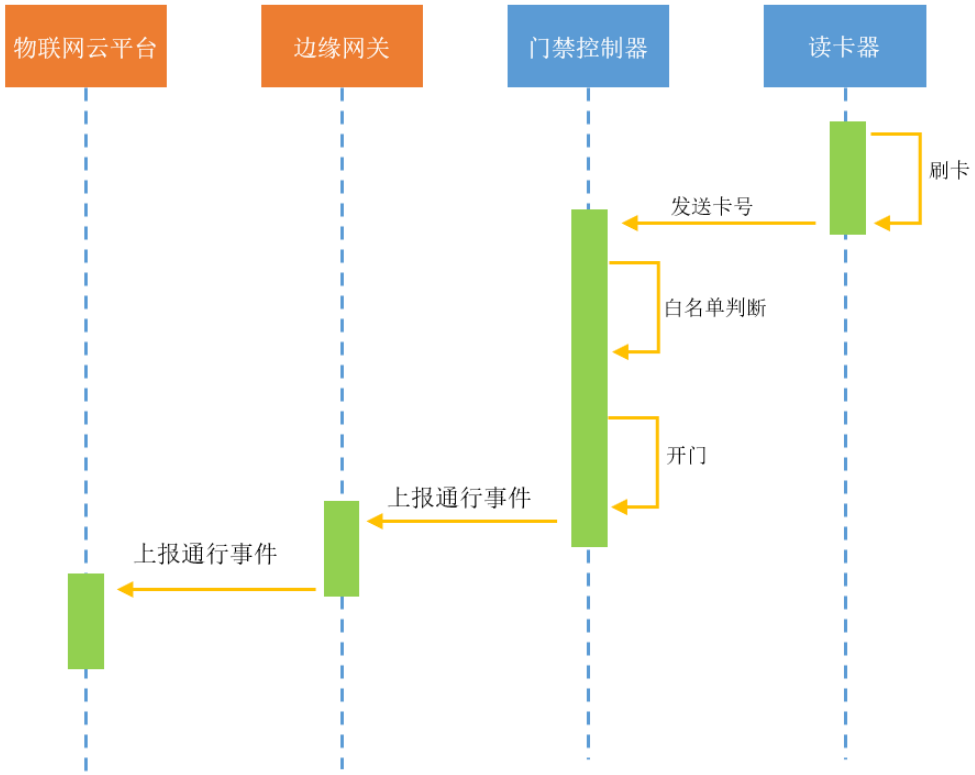
3．交互流程

3.1 刷卡门禁控制器

3.1.1 卡号同步流程



3.1.2 刷卡流程



4．对接流程

4.1 明确项目分工

- 根据系统架构图的内容，理解自定义协议驱动所处的位置。
- 明确各环节项目分工——需要设备端技术人员和协议驱动开发人员明确各自负责内容。
- 明确自定义协议驱动需要实现的功能，参考对应设备的物模型（刷卡门禁设备请参考《3.3 人行设备物模型-Rev0.1》文档的“第3章【智慧社区-人员通行-刷卡门禁】”物模型介绍。）。
- 明确自定义协议驱动与设备的通信方式，需要设备厂商提供设备端SDK，使自定义协议驱动可以基于该SDK实现访问设备数据和下发控制命令的功能。

4.2 信息收集

刷卡门禁设备信息：

- 提供设备的硬件型号、设备ID、设备IP及端口号

- 提供用于调试的卡号
- 射频卡读取时间
- 是否支持手机NFC
- 手机NFC读取时间
- 设备保存卡权限的时间

4.3 驱动开发及自测

- 参考《3.2 自定义协议驱动开发指导》文档，了解物联网设备接入，熟悉LEDA接口，熟悉实现驱动的基本方式，开发自定义协议驱动。
- 自定义协议驱动需通过驱动日志确认设备向云平台注册上线的接口被调用，并保证设备持续稳定地处于在线状态。
- 自定义协议驱动需通过驱动日志确认设备上报事件的接口被调用，通过驱动日志确认服务方法接口被调用。

4.3.1 驱动命名规则

在物联网应用服务平台，新创建的驱动英文名称定义有下面的规范建议：

- 驱动名称包含内容：

厂商名称_厂商产品型号_适用领域_驱动功能

如果这个驱动是通用驱动，适用所有厂商，则驱动名称里可以去掉厂商名称和厂商产品型号；

如果这个驱动支持某个厂商的所有产品型号，则驱动名称里可以去掉厂商产品型号。

- 适用领域说明：

人行: PerAccess (personal access)

车行: VehAccess (vehicle access)

EBA: EBA

安防：SecSystem（security system）

- 驱动功能说明：

例如EBA领域，Modbus功能，例如人行领域，可视对讲功能。

- 驱动名称使用大驼峰命名规范，即每个单词的首字母都要求是大写
- 举例说明：

一个驱动，可以支持能效通公司的所有型号的EBA Modbus设备，驱动名称可定义为：Nxtone_EBA_Modbus

4.4 测试验收

- 设备厂商按照技术对接要求完成开发后，厂商需要使用智慧社区设备认证实验室准入测试的测试用例进行测试，并满足通过标准。测试用例请参见附件：《人行测试用例》
- 厂商的设备通过智慧社区实验室准入测试后，需要设备寄送给智慧社区设备认证实验室进行测试认证。通过该认证后，厂商的设备则可以在项目中落地使用

5. 驱动开发

请参考《3.2 自定义协议驱动开发指导》文档。

3.5.1.3.2. 3.2 自定义协议驱动开发指导

概述

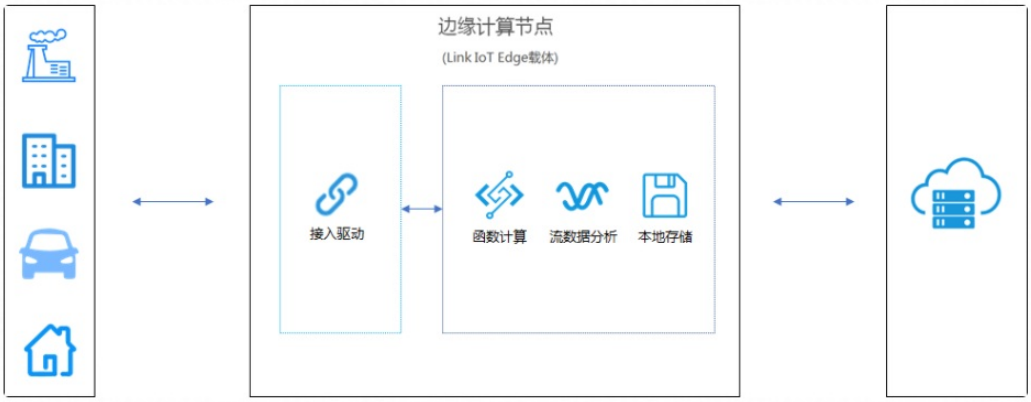
本文档指导自定义协议驱动开发人员，了解物联网设备接入相关知识，并且可以根据设备接入SDK开发驱动，实现设备物模型定义的功能。

1．了解物联网设备接入

1.1 设备接入概念

设备接入是Link IoT Edge提供的基础能力，设备接入模块在Link IoT Edge中称为驱动（driver）或设备接入驱动。所有连接到Link IoT Edge的设备都需要通过驱动实现接入。

设备接入驱动在Link IoT Edge框架位置如图所示。



1.2 设备接入驱动

1.2.1 设备接入驱动的功能组成

一个完整的驱动（设备接入模块）由设备的连接管理、设备的数据（协议）转换和设备的数据与命令处理三个模块组成。

1.2.1.1连接管理

指设备与网关建立通信连接。Link IoT Edge不限制建立通信连接的协议，可根据业务需求灵活选择。

1.2.1.2数据转换

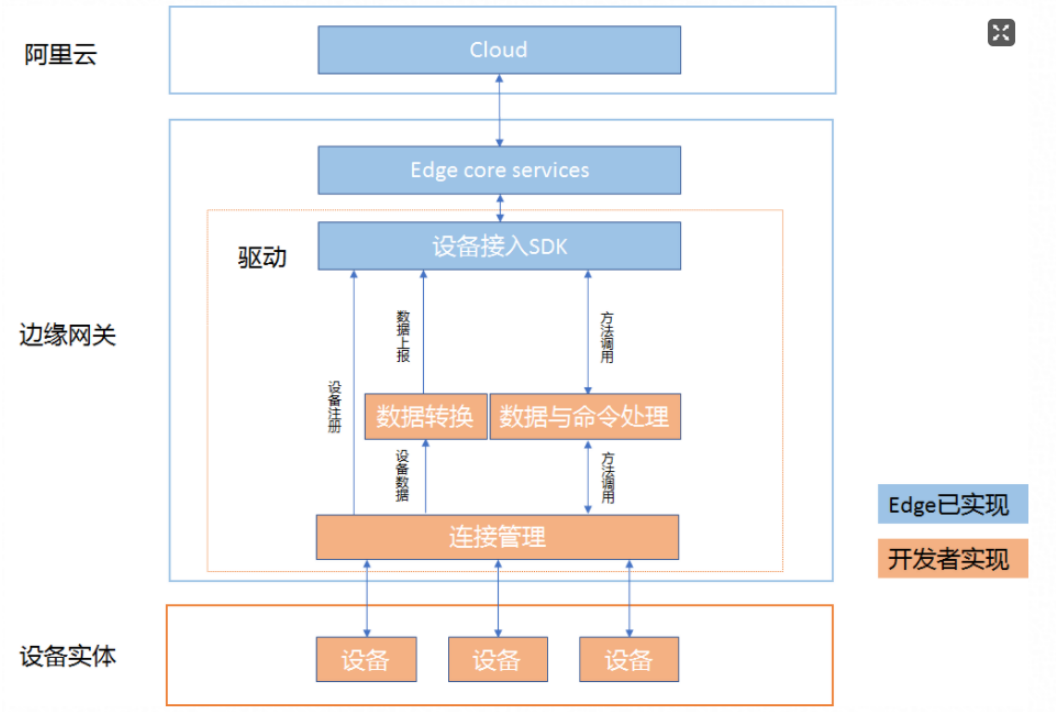
指设备接入驱动将获取到的终端设备数据转换为符合阿里云IoT物模型规范的数据格式，并上报到阿里云IoT Cloud。阿里云物联网平台物模型规范请参考物模型。

1.2.1.3数据与命令处理

指驱动可以处理云端对于设备的操作请求，并完成对设备的服务调用和处理调用结果，最终将结果返回到阿里云物联网平台。

1.2.2 设备接入驱动的开发工作

设备接入驱动是Link IoT Edge中独立的服务模块，您可以根据业务协议需求开发自定义设备接入驱动。下图展示了自定义驱动的功能和数据流向，并指出了开发一个自定义驱动需要做的开发工作。



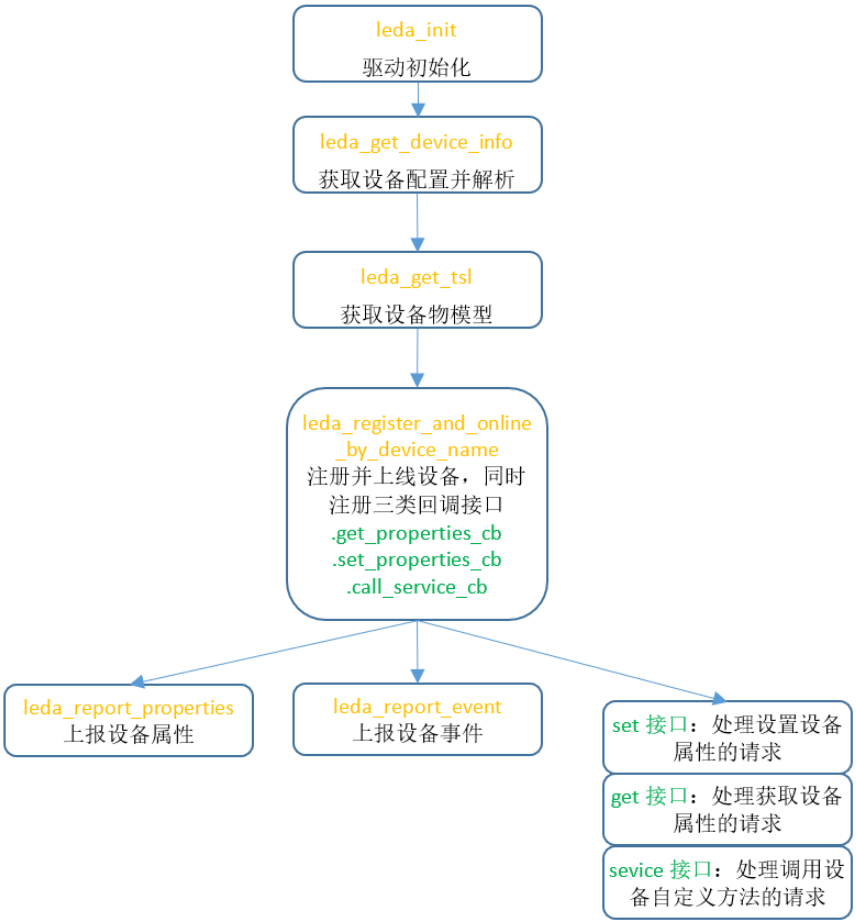
2．驱动开发

Link IoT Edge提供设备接入SDK（Link Edge Device Access，简称LEDA）方便您开发自己的驱动，SDK目前支持C、Node.js、Java和Python四种版本的语言。

2.1 LEDA接口介绍

LEDA支持的四种开发语言详细内容，请参见C、Node.js、Java和Python。

虽然SDK版本不同，但提供的功能是一样的，LEDA接口调用关系如下图所示：



2.2 驱动编码示例

在驱动开发过程中进行驱动编码时，需遵循物联网边缘计算的驱动编码规范和步骤。

2.2.1 驱动和设备配置

进行驱动编码前，需要了解Link IoT Edge的设备信息配置和驱动信息配置相关内容。

2.2.1.1 驱动配置

驱动信息配置在阿里云物联网平台进行配置。部署边缘实例时，驱动配置信息会被部署到边缘网关，其内容以json格式存储在Link IoT Edge配置中心，可以通过**leda_get_driver_info**接口获取。

驱动信息配置为JSON格式：

```
{
    "json":{
        "ip":"127.0.0.1",
        "port":54321
    }
}
```

格式参数说明如下：

参数名称	说明
json	驱动配置的格式为JSON格式配置。配置内容为自定义内容。

驱动配置示例：



2.2.1.2 设备配置

设备信息配置在阿里云物联网平台控制台配置。部署边缘实例时，设备信息配置会被部署到边缘网关，其内容以json格式存储，可以通过**leda_get_device_info**接口获取。

设备信息配置格式定义：

```
{
    "deviceList": [{
        "custom": {
             "ip":"127.0.0.1",
             "port":22322
        }, // 设备自定义配置
        "productKey": "xxxxxxxxxxx", // 产品ProductKey，在创建产品时生成
        "deviceName": "demo_led",    // 设备DeviceName，在创建设备时设置
    }]
}
```

设备信息配置参数说明：

配置名称	配置解释
deviceList	当前驱动下所有已进行设备配置的设备列表。
custom	设备自定义配置。
productKey	设备所在产品唯一标识符。
deviceName	设备名称。

实际设备配置示例：



2.2.2 驱动代码示例

完成驱动编码，可参考以下4个示例。本示例的完整工程源码请参见Github源码库LED设备驱动。

2.2.2.1 初始化驱动资源

调用leda_init接口完成资源初始化。

```
int main(int argc, char** argv)
{
    ...

    /* init driver */
    if (LE_SUCCESS != (ret = leda_init(WORKER_THREAD_NUMS)))
    {
        log_e(TAG_NAME_LED_DRIVER, "leda_init failed\n");
        return ret;
    }

    ...

  return LE_SUCCESS;
}
```

2.2.2.2 解析驱动配置，完成设备上线
通过调用leda_get_driver_info接口获取驱动配置，解析设备的连接信息，并根据解析结果连接设备。设备连接成功后，调用leda_get_device_info接口获取设备配置并解析，根据解析结果验证设备功能。功能验证通过后，调用leda_register_and_online_by_device_name接口完成设备注册并上线到阿里云物联网平台。

```
static int online_devices()
{
  ...

    /* 获取驱动和设备配置 */
    size = leda_get_device_info_size();
    if (size >0)
    {
        device_config = (char*)malloc(size);
        if (NULL == device_config)
        {
            log_e(TAG_DEMO_LED, "allocate memory failed\n");
            return LE_ERROR_INVAILD_PARAM;
        }

        if (LE_SUCCESS != (ret = leda_get_device_info(device_config, size)))
        {
            log_e(TAG_DEMO_LED, "get device config failed\n");
            return ret;
        }
    }

    /* 解析驱动和设备配置 */
    devices = cJSON_Parse(device_config);
    if (NULL == devices)
    {
        log_e(TAG_DEMO_LED, "device config parser failed\n");
        return LE_ERROR_INVAILD_PARAM;
    }

    cJSON_ArrayForEach(item, devices)
    {
        if (cJSON_Object == item->type)
        {
            /* 解析配置内容 */
            result      = cJSON_GetObjectItem(item, "productKey");
            productKey  = result->valuestring;

            result      = cJSON_GetObjectItem(item, "deviceName");
            deviceName  = result->valuestring;

            result      = cJSON_GetObjectItem(item, "custom");
            if (NULL != result)
            {
                log_i(TAG_DEMO_LED, "custom content: %s\n", cJSON_Print(result));
            }

            /* 注册并上线设备 */
            device_cb.get_properties_cb            = get_properties_callback_cb;
            device_cb.set_properties_cb            = set_properties_callback_cb;
            device_cb.call_service_cb              = call_service_callback_cb;
            device_cb.service_output_max_count     = 5;

            dev_handle = leda_register_and_online_by_device_name(productKey, deviceName, &device_cb, NULL);
            if (dev_handle < 0)
            {
                log_e(TAG_DEMO_LED, "product:%s device:%s register failed\n", productKey, deviceName);
                continue;
            }

            g_dev_handle = dev_handle;
            log_i(TAG_DEMO_LED, "product:%s device:%s register success\n", productKey, deviceName);
        }
    }

  ...

    return LE_SUCCESS;
}
```

2.2.2.3 上报数据

将收到的设备数据转换为阿里云IoT物模型格式并上报到物联网平台，调用leda_report_properties接口上报设备属性数据，调用leda_report_event接口上报设备事件。

```
/* 上报数据 */
while (1)
{
    /* 上报属性 */
    leda_device_data_t dev_proper_data[1] =
    {
        {
            .type  = LEDA_TYPE_INT,
            .key   = {"temperature"},
            .value = {0}
        }
    };
    sprintf(dev_proper_data[0].value, "%d", g_dev_temperature);
    leda_report_properties(g_dev_handle, dev_proper_data, 1);

    /* 上报事件 */
    if (g_dev_temperature > 50)
    {
        leda_device_data_t dev_event_data[1] =
        {
            {
                .type  = LEDA_TYPE_INT,
                .key   = {"temperature"},
                .value = {0}
            }
        };
        sprintf(dev_event_data[0].value, "%d", g_dev_temperature);
        leda_report_event(g_dev_handle, "high_temperature", dev_event_data, 1);
    }

    sleep(5);
}
```

2.2.2.4 处理云端服务请求

实现服务请求的三个回调函数如下所示。

get接口：处理获取设备属性的请求。

set接口：处理设置设备属性的请求。

service接口：处理调用设备自定义方法的请求。

```
static int get_properties_callback_cb(device_handle_t device_handle,
                                      leda_device_data_t properties[],
                                      int properties_count,
                                      void *usr_data)
{
    ...

    return ret;
}

static int set_properties_callback_cb(device_handle_t device_handle,
                                      const leda_device_data_t properties[],
                                      int properties_count,
                                      void *usr_data)
{
    ...

    return ret;
}

static int call_service_callback_cb(device_handle_t device_handle,
                                    const char *service_name,
                                    const leda_device_data_t data[],
                                    int data_count,
                                    leda_device_data_t output_data[],
                                    void *usr_data)
{
    ...

    return ret;
}
```

2.3 驱动产出方式

2.3.1 驱动依赖注意事项

设备接入驱动根据协议和业务场景的不同，可能会涉及第三方库依赖。Link IoT Edge针对开发设备接入驱动所用不同开发语言，分别制定了第三方库依赖规则。

- C版本SDK：C语言属于编译型语言，如果编译目标环境和运行环境不一致，则很可能导致不可运行。所以对于使用设备接入C版本SDK开发驱动，需要保证开发编译目标环境和运行环境相同。 驱动包中包含驱动程序和依赖动态库。如果该驱动依赖于第三方库，则需要将动态库和驱动程序一起打包生成最终驱动程序包。
- Node.js版本SDK：使用设备接入SDK Node.js版本开发驱动时，若依赖第三方库，需要到Link IoT Edge运行环境上开发驱动，并在驱动目录中使用如下命令安装依赖。

```
npm install 第三方库名
```

- Python版本SDK：使用设备接入SDK Python版本开发驱动时，若依赖第三方库，需要到Link IoT Edge运行环境上开发驱动，并在驱动目录中使用如下命令安装依赖。

```
pip3 install -t . 第三方库名
```

2.3.2 驱动打包方式

基于Link IoT Edge提供的SDK开发驱动并完成调试后，需将产物打包为.zip包，并确保驱动Binary或index源文件在.zip包第一级目录。

每个版本SDK开发的驱动在打包时，有不同的打包规则。

- 基于C SDK开发的驱动对于C语言开发的驱动，驱动包中包含驱动程序和驱动依赖的动态库。如果驱动程序包含依赖库，则需要将依赖库放置指定的位置，即在驱动程序当前路径下的lib文件夹下。具体操作步骤如下：
  i. 规定驱动程序需命名为main。
  ii. 在main当前路径下创建lib文件夹。
  iii. 将main依赖的动态库全部拷贝到lib文件夹下。
  iv. 使用zip命令对当前路径下的main和lib进行压缩处理生成zip包。

```
zip -r your_driver_name.zip main lib
```

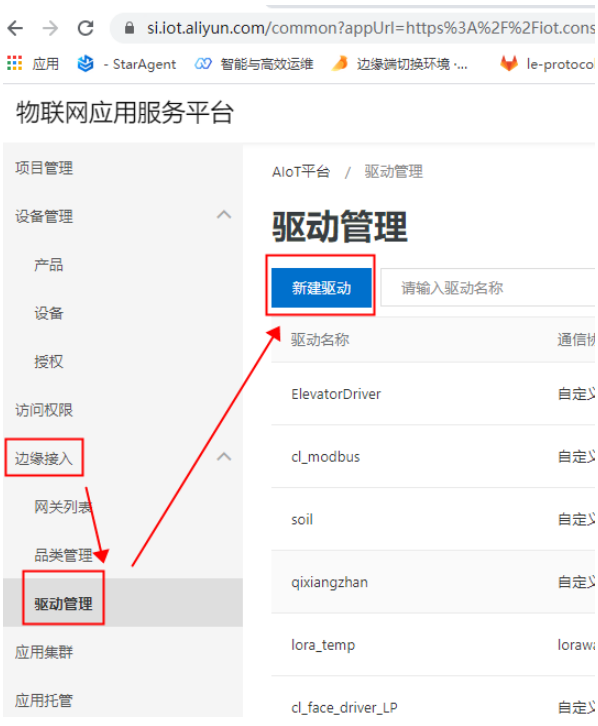
- 基于Python SDK开发的驱动驱动包文件中须包含index.py，并且在该文件中定义handler函数。驱动是一个在函数计算应用引擎中持续运行的函数，所以在驱动包中须包含index.py文件，并且在该文件中定义handler函数。 驱动运行时，会加载index.py文件。而该文件中，函数计算应用定义的handler函数是不会被调用，因此驱动代码须放在handler函数外，保证加载index.py文件时能直接执行。详情请参考Python版本SDK
- 基于Node.js SDK开发的驱动驱动包文件中须包含index.js，并且在该文件中定义handler函数。 驱动运行时，会加载index.js文件。而该文件中，函数计算应用定义的handler函数是不会被调用，因此驱动代码须放在handler函数外，保证加载index.js文件时能直接执行。详情请参考Nodejs版本SDK

2.4 驱动使用示例

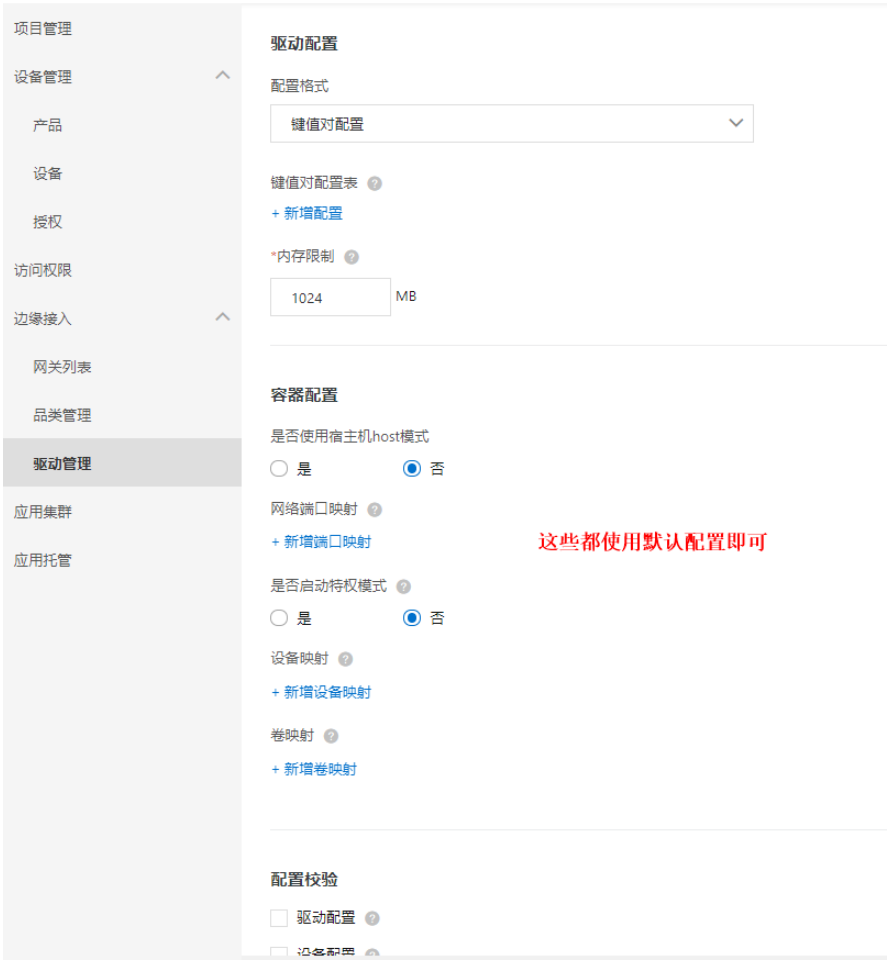
2.4.1 上传及发布驱动

- 在SI集成工作台si.iot.aliyun.com上，选择**边缘接入->驱动管理**，选择**新增驱动**

  根据提示设置参数，需要注意的是通信协议类型要选择【自定义】




物联网应用服务平台
项目管理
设备管理
产品
设备
授权
访问权限
边缘接入
网关列表
品类管理
驱动管理
应用集群
应用托管
AIoT平台 / 驱动管理 / 新建驱动
驱动管理
驱动信息
* 驱动名称
demodriver
* 通信协议类型
自定义
* 语言类型
C
* CPU架构
x86-64
* 驱动适配的边缘版本
2.4.7 版本及以上
* 驱动版本
1.0
驱动版本描述
请输入驱动版本描述
驱动是否内置
是
否
* 驱动文件
上传文件



- 上传已完成编译并打包的led_driver.zip文件。
- 完成参数的设置并上传成功驱动文件后，单击**确定**。您可以在**自研驱动列表**中看到刚刚创建的驱动

2.4.2分配驱动和设备到边缘实例

2.4.2.1 添加驱动到边缘实例

- 登录SI集成工作台si.iot.aliyun.com，在网关管理设备集成页面，点击网关右侧的设置图标，首先，在弹出菜单中选择**添加协议**：

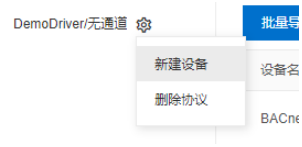


- 然后，在下拉选择的协议列表中选择“DemoDriver”，选择“确定”。

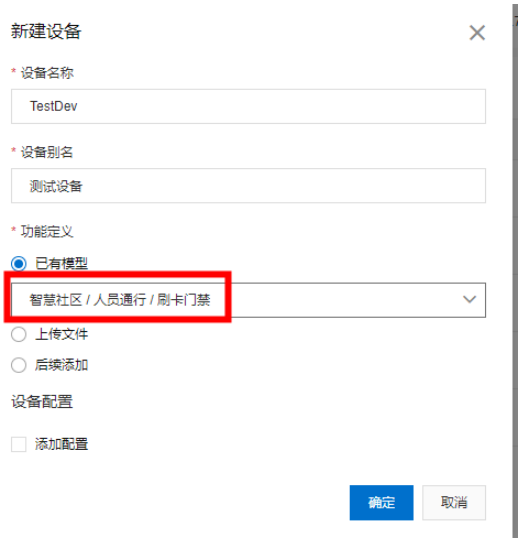


2.4.2.2 新建设备

- 如下图所示，通过点击通道右侧的**“设置”**图标，选择**“新建设备”**：



- 在弹出的新建设备页面中按照如下方式选择刷卡门禁



- 点击**确定**按钮即可完成设备创建，创建好的设备如下如所示：



2.4.3 部署边缘实例

- 在**实例详情**页面，右上角单击**部署**，部署边缘实例。部署成功后边缘实例名称后显示**部署成功**。



2.4.4 查看驱动日志

在边缘网关宿主机命令行中通过docker命令进入LE网关运行环境，命令如下：

```
sudo docker exec -it $(sudo docker ps | grep k8s_linkedge  | awk '{print $1}') bash
```

进入网关容器内之后可以在本地查看驱动日志。

a. 通过 `./fctl show` 命令，可以查看到部署驱动的具体位置。

```
cd /linkedge/gateway/build/bin/
./fctl show
```

系统显示类似如下图所示：



参数说明：

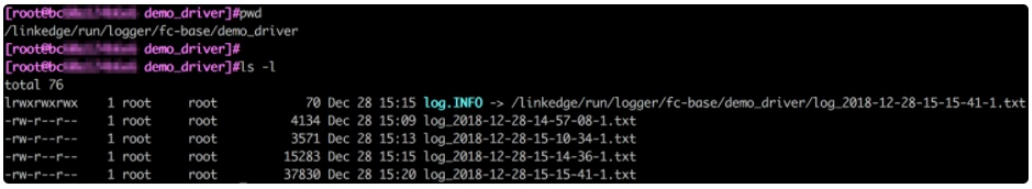
字段名称	字段解释
DriverName	驱动名称，该名称为在**上传及发布驱动**步骤中设置的驱动名称。
CodePath	驱动部署到边缘网关中的位置路径。
Process PID	驱动启动后的进程ID。

b. 驱动在运行过程中会产生运行日志，通过查看运行日志可以了解驱动运行状态，每个自定义驱动生成的日志文件统一放在/linkedge/run/logger/fc-base路径下。每个驱动日志文件路径为/linkedge/run/logger/fc-base/xxxx/log_xxxx.txt。

说明驱动日志文件路径中的xxxx为在上传驱动时填写的名称。

```
cd /linkedge/run/logger/fc-base/demo_driver && ls -l
```

详情如下图所示：



驱动运行日志可以查看大部分驱动的运行状态，但是有时还需要查看其它日志配合了解当前驱动的运行情况，例如设备上线异常时可以查看 `dimu` 日志，日志文件路径为/linkedge/run/logger/dimu/log_xxxxx.txt。

```
cd /linkedge/run/logger/dimu
ls -l
```

详情如下图所示：

```
[root@bc dimu]#pwd
/linkedge/run/logger/dimu
[root@bc dimu]#
[root@bc dimu]#ls -l
total 32
lrwxrwxrwx    1 root     root            55 Dec 28 15:15 log.INFO -> /linkedge/run/logger/dimu/log_2018-12-28-15-15-40-1.txt
-rw-r--r--    1 root     root          1459 Dec 28 11:11 log_2018-12-28-11-10-26-1.txt
-rw-r--r--    1 root     root          3793 Dec 28 14:47 log_2018-12-28-14-47-04-1.txt
-rw-r--r--    1 root     root          3638 Dec 28 14:55 log_2018-12-28-14-47-57-1.txt
-rw-r--r--    1 root     root          3899 Dec 28 15:10 log_2018-12-28-14-57-07-1.txt
-rw-r--r--    1 root     root          3807 Dec 28 15:13 log_2018-12-28-15-10-33-1.txt
-rw-r--r--    1 root     root          5475 Dec 28 15:15 log_2018-12-28-15-14-35-1.txt
-rw-r--r--    1 root     root          3204 Dec 28 15:15 log_2018-12-28-15-15-40-1.txt
```

设备上线成功后，如果设备数据上报有异常，则可以查看 `cloud_proxy` 日志。cloud_proxy产生的运行日志文件路径格式为/linkedge/run/logger/cloud-proxy/log_xxxxx.txt。

```
cd /linkedge/run/logger/cloud-proxy
ls -l
```

详情如下图所示：

```
[root@bc cloud-proxy]#pwd
/linkedge/run/logger/cloud-proxy
[root@bc cloud-proxy]#
[root@bc cloud-proxy]#ls -l
total 1128
lrwxrwxrwx    1 root     root            62 Dec 28 15:15 log.INFO -> /linkedge/run/logger/cloud-proxy/log_2018-12-28-15-15-40-1.txt
-rw-r--r--    1 root     root          7299 Dec 28 11:11 log_2018-12-28-11-10-25-1.txt
-rw-r--r--    1 root     root         30451 Dec 28 14:47 log_2018-12-28-14-47-04-1.txt
-rw-r--r--    1 root     root        165951 Dec 28 14:55 log_2018-12-28-14-47-57-1.txt
-rw-r--r--    1 root     root        472630 Dec 28 15:10 log_2018-12-28-14-57-07-1.txt
-rw-r--r--    1 root     root        136951 Dec 28 15:13 log_2018-12-28-15-10-33-1.txt
-rw-r--r--    1 root     root         76663 Dec 28 15:15 log_2018-12-28-15-14-35-1.txt
-rw-r--r--    1 root     root        247473 Dec 28 15:21 log_2018-12-28-15-15-40-1.txt
```

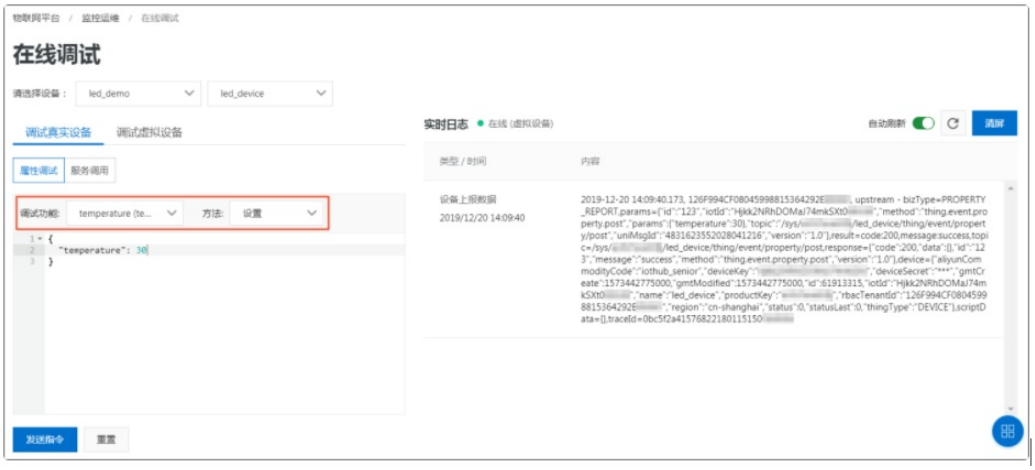
2.5 驱动调试方式

2.5.1 云端在线调试

如果设备上线成功，则可以使用阿里云物联网平台**在线调试**功能调试驱动和设备，该功能页面还可以实时查看设备上报的数据信息，也可以触发对设备服务的调用请求。

a.在物联网平台控制台，选择**监控运维 > 在线调试**，在**在线调试**页面，选择调试产品和调试设备。

b.选择调试设备待调功能和服务，进行调试，并查看设备实时运行日志

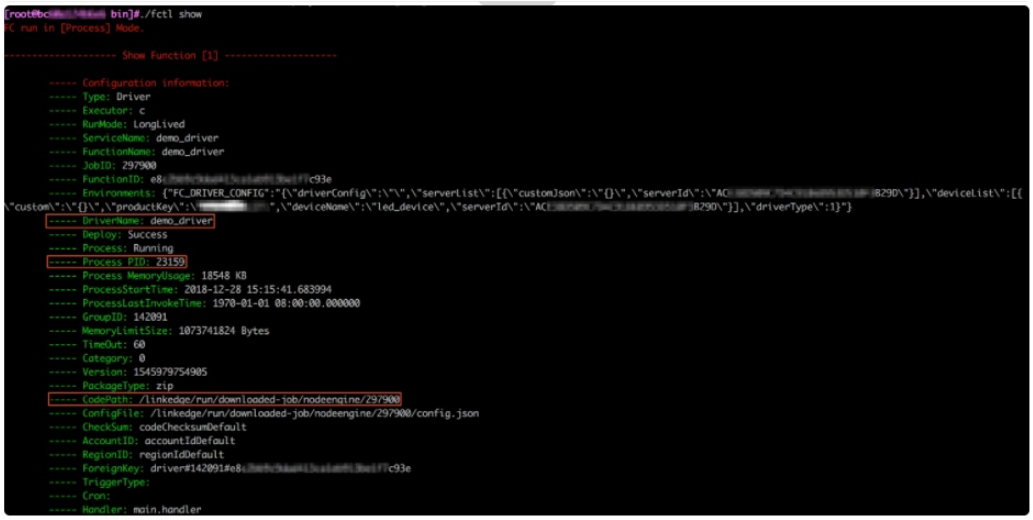


2.5.2 边缘端替换驱动

如果在驱动调试过程中发现问题，需要修改驱动代码重新生成新的驱动进行调试，这时只需要在本地替换修改编译后的驱动即可。

a. 找到驱动的位置。可以通过**fctl**命令进行查找，具体字段解释查看调试信息。

```
cd /linkedge/gateway/build/bin/
./fctl show
```



b. 根据CodePath找到驱动所在位置路径，完成替换。

c. 根据Process PID得到驱动进程ID，使用**kill -9 Pid**命令重新启动驱动。

```
kill -9 Pid #Pid为驱动进程ID，即通过fctl命令查看得到的Process PID
```

3.5.2. 车行标准

3.5.2.1. 车行道闸对接方案

3.5.2.1.1. 道闸一体机对接方案介绍

1．概述

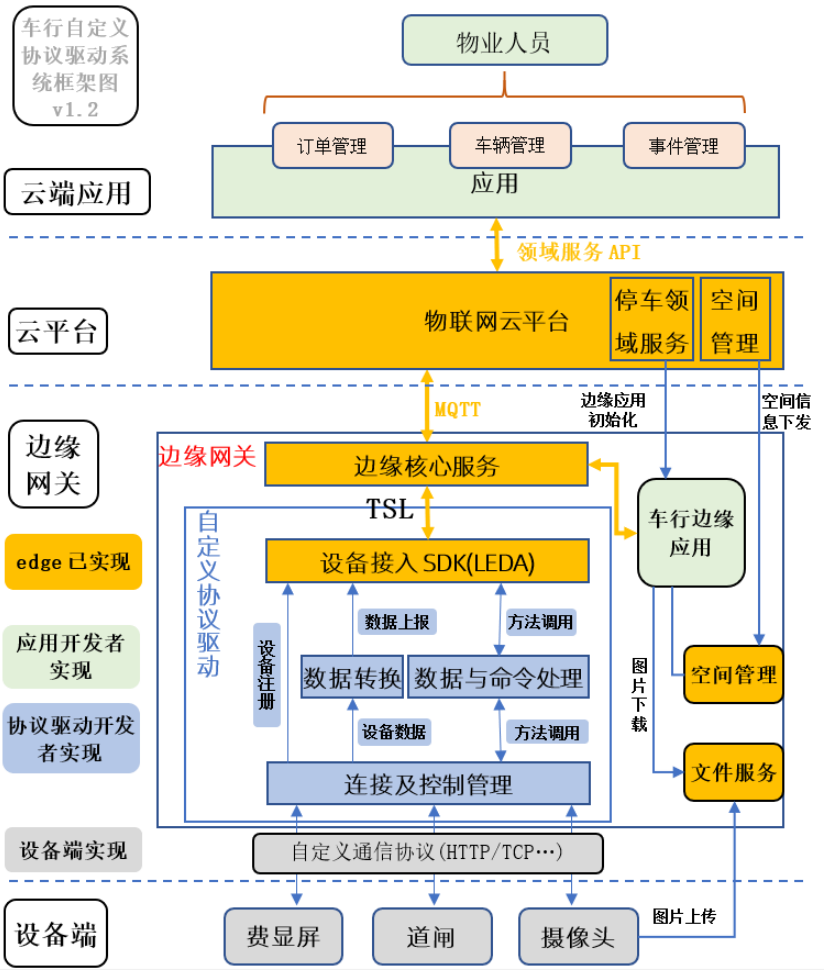
1.1 编写目的

本文档介绍道闸一体机（包括费用显示屏、车辆道闸、车辆识别摄像头）接入到边缘服务器的对接方案和设备驱动的开发流程。

1.2名词解释

名词	解释
Link IoT Edge	物联网边缘计算产品（Link IoT Edge，简称LE），即阿里云物联网平台（IoT）中的边缘计算产品。提供安全可靠的数据计算能力，可供本地处理设备数据，减少上传云端的成本。
网关	运行Link IoT Edge软件的计算设备统称为边缘网关或边缘服务器，简称网关。
子设备	指通过一定的协议或接口接入到Link IoT Edge网关上的设备（即设备接入到网关后称为子设备），网关代理该子设备与云端进行通信。
驱动	Link IoT Edge中的设备接入模块称为驱动（driver）或设备接入驱动。所有连接到Link IoT Edge的设备都需要通过驱动实现接入。

2．系统架构图



图解说明：

① 云端应用：停车应用软件。功能包括订单管理、车辆管理、事件管理等，该软件基于云平台提供的领域服务API开发，实现对设备上报事件的处理及对设备的控制。

② 云平台：物联网云平台。向托管在边缘网关内的停车应用软件提供空间管理及停车领域服务。向自定义驱动提供物模型。

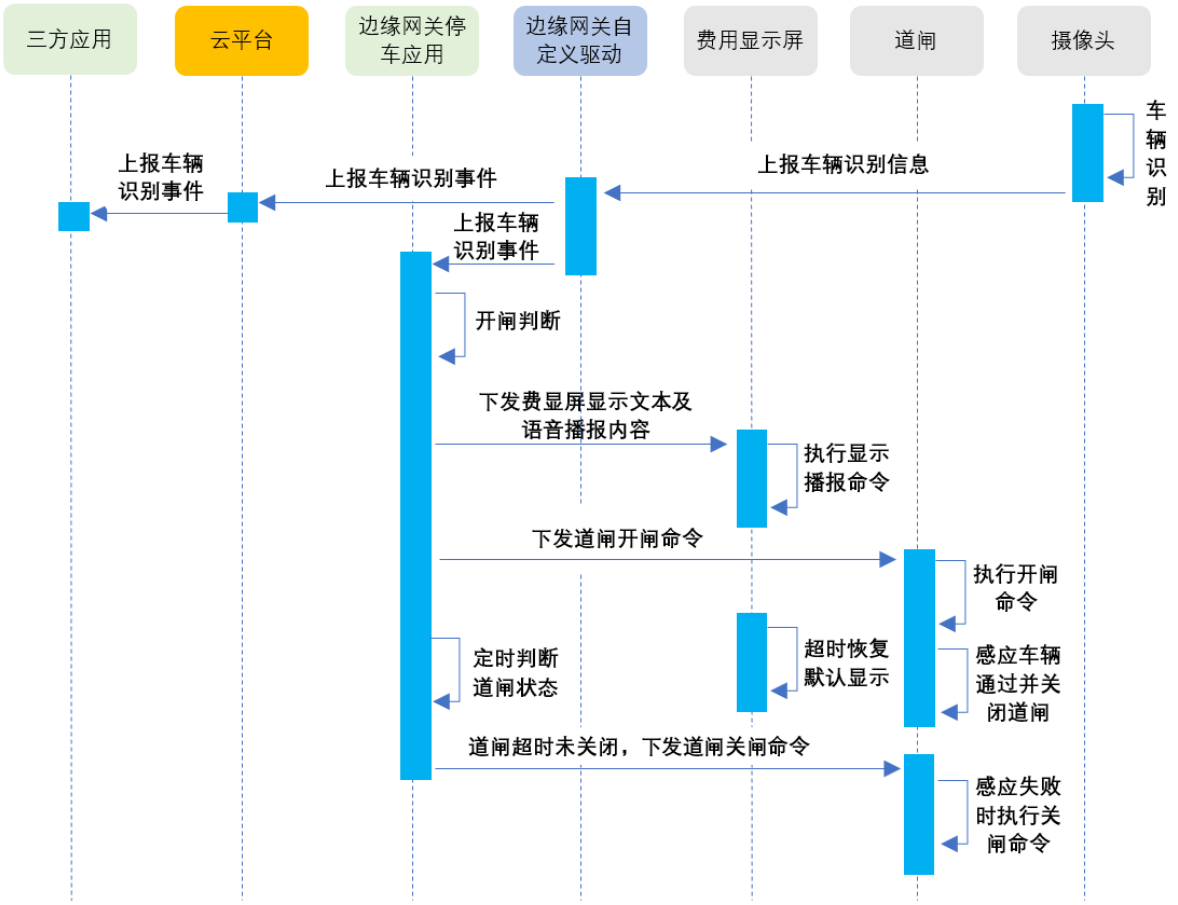
③ 边缘网关：边缘网关内主要角色分自定义协议驱动和停车边缘应用，驱动运行在LE组件中，负责管理及控制设备，其中包含阿里云提供的设备接入SDK（LEDA）部分和开发者实现部分。停车边缘应用负责各场景应用实现。

④ 设备端：道闸一体机，包括摄像头、道闸、费显屏，设备供应商应提供设备通信协议，供驱动访问或控制设备使用。设备供应商应提供硬件配置工具，可以进行设备初始化、设备参数配置及故障定位功能。

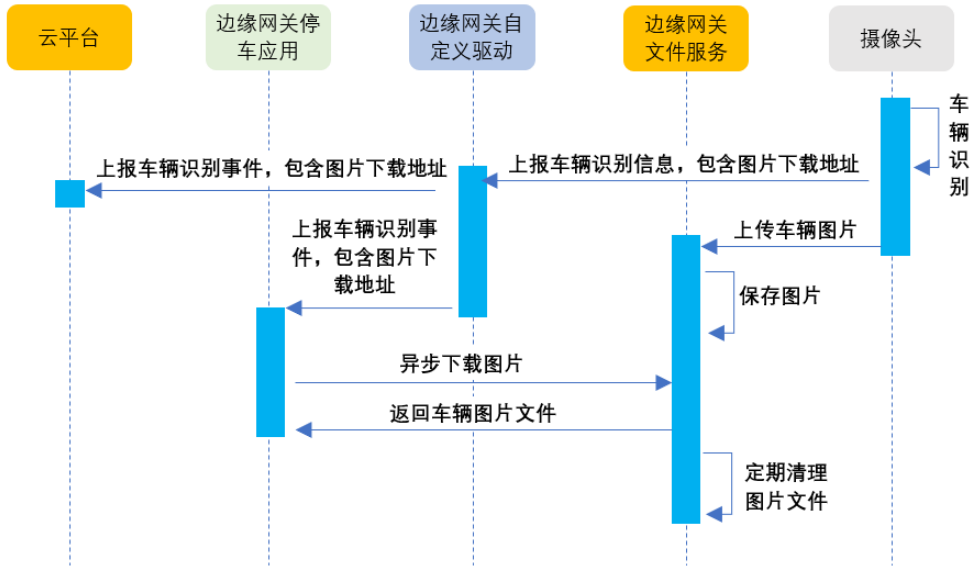
⑤ 图片上传：摄像头应通过边缘网关文件服务提供的访问端口，使用HTTP协议上传车辆及车牌图片。

3. 交互流程

3.1 车辆入场/出场



3.2 上传车辆图片



4．对接流程

4.1 明确方案分工

- 根据系统架构图的内容，进行方案可行性评估，理解自定义协议驱动所处的位置。
- 明确各环节项目分工——需要设备端技术人员和协议驱动开发人员明确各自负责内容。
- 明确自定义协议驱动需要实现的功能，参考对应设备的物模型（请参考《3.3 道闸一体机设备物模型》文档的物模型介绍）。
- 明确自定义协议驱动与设备的通信方式，需要设备厂商提供设备端SDK，使自定义协议驱动可以基于该SDK实现访问设备数据和下发控制命令的功能。

4.2 信息收集

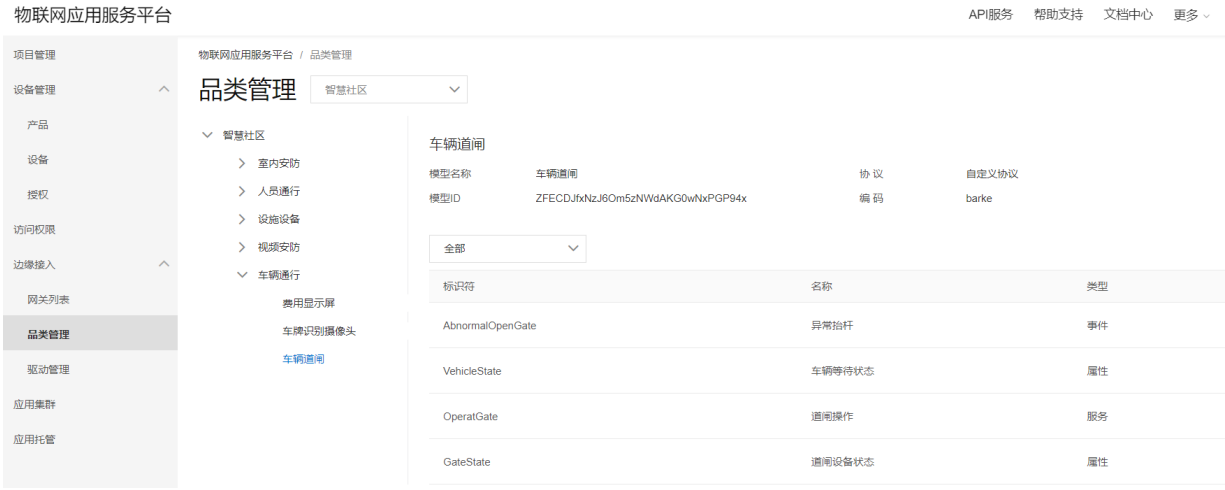
- 提供设备的硬件型号、设备ID

4.3 驱动开发

- 参考《3.2 道闸一体机驱动开发指导》文档，了解物联网设备接入，熟悉LEDA接口，熟悉实现驱动的基本方式，开发自定义协议驱动。
- 参考《3.4 边缘文件服务接口说明》文档，实现车辆图片上传功能。
- 自定义协议驱动需通过驱动日志确认设备向云平台注册上线的接口被调用，并保证设备持续稳定地处于在线状态。
- 自定义协议驱动需通过驱动日志确认设备上报事件的接口被调用，通过驱动日志确认服务方法接口被调用。

5. 物模型

请参考物联网应用服务平台里**品类管理**-车辆通行，如下图所示：



附录

驱动命名规则

在物联网应用服务平台，新创建的驱动英文名称定义有下面的规范建议：

- 驱动名称包含内容：

厂商名称_厂商产品型号_适用领域_驱动功能

如果这个驱动是通用驱动，适用所有厂商，则驱动名称里可以去掉厂商名称和厂商产品型号；

如果这个驱动支持某个厂商的所有产品型号，则驱动名称里可以去掉厂商产品型号。

- 适用领域说明：

人行: PerAccess (personal access)

车行: VehAccess (vehicle access)

EBA: EBA

安防：SecSystem（security system）

- 驱动功能说明：

例如EBA领域，Modbus功能，例如人行领域，可视对讲功能。

- 驱动名称使用大驼峰命名规范，即每个单词的首字母都要求是大写
- 举例说明：

有一个驱动，可以支持能效通公司的所有型号的EBA Modbus设备，驱动名称可定义为：Nxtone_EBA_Modbus.

3.5.2.1.2. 道闸一体机驱动开发指导

请参考 自定义协议驱动开发指导

3.5.2.1.3. 边缘文件服务接口说明

1. 概述

边缘文件服务，提供人行、车行通行图片存储功能，使用HTTP restful API。

2. 接口定义（HTTP）

主机地址-根据实际情况配置，容器内可使用homelinkedge服务域名访问。

主机内访问端口- 9102；主机外访问端口- 32012。

2.1文件增加

接口定义

	path	Content-Type
POST	/home/paas/file/add	form-data

入参

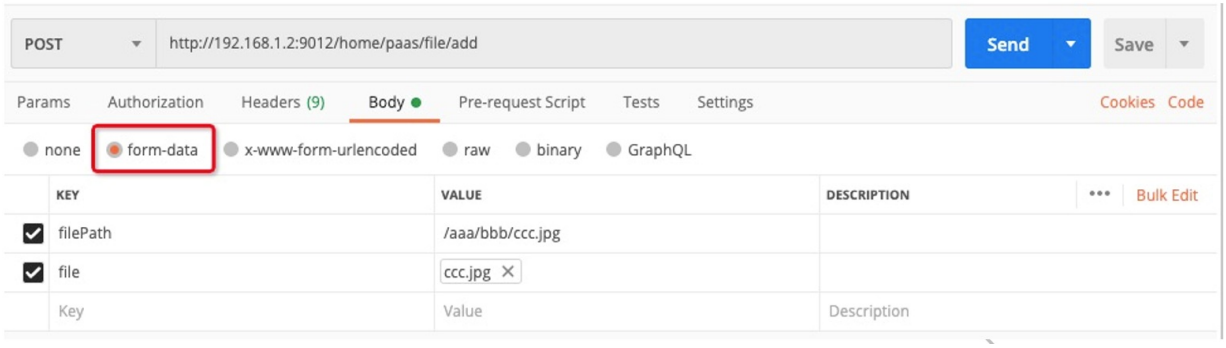
字段	类型	是否必传	备注
filePath	Text	是	文件写入路径
file	File	是	文件流
expire	Long	可选	文件最短保存时间，单位是秒，默认是2小时。可设置的范围是[0, 604800]

出参

字段	类型	备注
code	Integer	返回码
message	String	返回信息
data	String	文件下载url

示例

请求示例



返回示例

```
{
  "code": 200,
  "message": "success",
  "data": "http://192.168.1.2:9102/oss/aaa/bbb/ccc.jpg"
}
```

2.2文件删除

接口定义

	path	Content-Type
POST	/home/paas/file/delete	json

入参

字段	类型	是否必传	备注
filePath	Text	是	写入的文件路径

出参

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

示例

请求示例

```
{
  "filePath":"aaa/bbb/ccc/jpg"
}
```

返回示例

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

2.3文件获取

接口定义

	path
GET	oss/文件路径，例如: oss/aaa/bbb/ccc.jpg

出参

文件流

示例

请求示例

http://ip:port/oss/aaa/bbb/ccc.jpg

3.5.2.2. 停车系统对接方案

3.5.2.2.1. 边缘应用开发手册

1 基本概念说明

本文提到的边缘应用，是指部署在物业管理一体机的边缘应用，是基于IoT的领域服务对接方案实现的边缘应用。领域服务对接方案里有服务模型定义和数据模型定义，下面介绍一些相关的基本概念：

服务模型：

是一组可提供完整业务功能的HTTP/HTTPS接口。边缘应用开发者可以根据对应的场景需求和业务需求进行这套接口定义。

服务提供方：

服务提供方即是可以提供服务模型里定义的服务功能的应用。服务提供方即可以是云端应用，也可以是边缘端应用

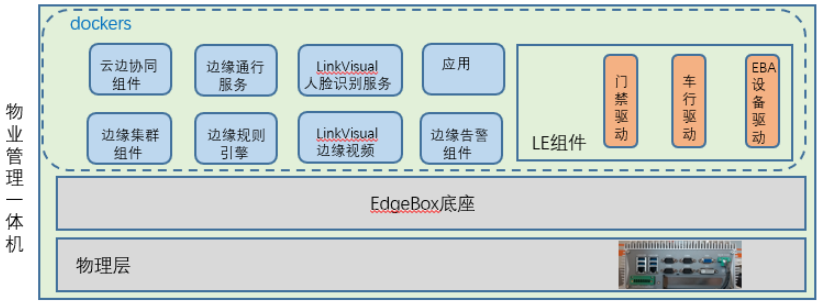
服务依赖方：

服务依赖方即是使用服务模型里定义的服务功能的应用。服务依赖方也可以是云端应用或边缘端应用

数据模型：

- 使用场景：本地系统/设备上报的消息可以基于数据模型进行定义，例如人脸通行事件
- 通过数据模型和IoT数据总线机制，可实现应用间的数据信息流转
- IoT平台提供了对数据进行增删改查的4个API，以及HTTP2方式的消息订阅机制
- (数据提供方)应用可以通过数据添加的API接口，将本地系统或设备的消息送入数据总线。
- (数据消费方)应用可以通过查询数据模型的API接口获取消息，也可以通过数据变更消息订阅方式获取消息。
- 数据模型支持图片文件上传，例如人脸通行事件的人脸照片

物业管理一体机框架：



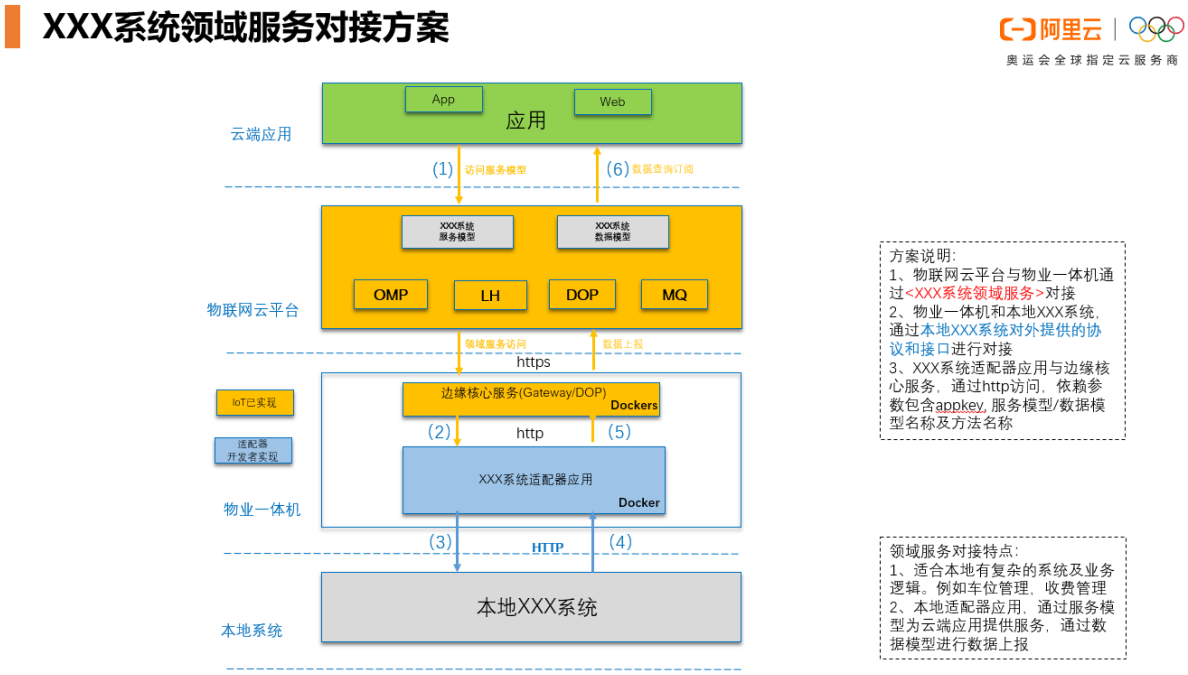
物业管理一体机是基于K8S框架实现的，底层是EdgeBox底座，之上是基于docker的各种应用和服务。

LE组件是基于docker的服务程序，里面包含了支持各种设备接入驱动，例如门禁驱动，车行驱动，EBA设备驱动等

边缘应用也是基于docker的应用程序。边缘应用可通过编译出的jar包，打包成镜像，然后通过IoT云端平台将应用下发到指定的物业管理一体机。边缘应用的启动入口，可以通过应用jar包里的docker file指定。

2 整体架构

下面将介绍**边缘端适配器应用**在整体架构里的位置以及上下游模块的关系，便于更好的理解边缘应用开发方法



2.1 核心模块功能说明

云端应用：即SaaS应用，一般由ISV提供，是服务模型依赖方，负责服务模型的调用和数据订阅。

物联网云平台：即IoT云端平台。在领域服务对接方案里，与边缘端核心服务一起提供服务总线和数据总线框架服务。

适配器应用：即运行在物业管理一体机的边缘应用，是服务模型的提供方。该应用一方面接受云端应用通过IoT平台的服务模型调用，然后将调用转换成本地系统支持的接口调用，另外一方面该应用接受本地系统的事件上报，然后通过边缘数据总线，将消息安装数据模型格式要求，上传到IoT云端平台

本地系统：即项目现场的本地各种系统，例如立方停车场系统。

2.2 核心流程说明

基于边缘适配器应用的领域服务对接方案，核心流程包括两个：至上而下的服务模型调用，至下而上的数据上报。下面分别说明。下面的序号，与框架图中的序号一一对应。

服务调用：

(1)云端SaaS应用调用服务模型提供的HTTP服务。 发起服务调用时，需提供项目Appkey, 路径名称(path)里需包含服务模型ID+接口方法名称。

(2)边缘适配器应用，侦听到对应的服务调用后，进行适当的适配转换，再调用本地系统提供的Http服务

(3)本地系统有一套对外开放的Http服务。侦听到来自边缘适配器应用调用后，完成相应的功能并将结果返回。

数据上报：

(4)本地系统发生事件，可以将事件内容通过Http接口发送给边缘适配器应用。接口的url可以双方(适配器应用/本地系统)约定

(5)边缘适配器应用侦听到事件请求后，可以将事件内容转化成数据模型要求的格式，然后根据(边缘)IoT平台提供的数据插入接口，上报数据到边缘数据总线，然后内部流转到云端数据总线。

(6)云端SaaS应用，可以通过IoT平台提供的查询数据模型的API接口获取数据，也可以通过数据变更消息订阅方式获取数据

3 边缘应用开发指导

这里介绍的边缘应用，是服务提供方应用

3.1 边缘应用对接服务模型

边缘应用侦听到对应的服务调用后，进行适当的适配转换，再调用本地系统提供的Http服务。

边缘应用对接服务模型，可以参考IoT公开链接：服务总线：服务提供的开发示例

3.2 边缘应用对接数据模型

具体对接可参考IoT公开链接：边缘应用数据总线对接

这里只补充说明一些需要关注的内容：

通过本地系统环境变量获取appkey and appSecrect：

1. public static final String appkey =System.getenv("iot.hosting.appKey");
2. public static final StringappSecret =System.getenv("iot.hosting.appSecret");
3. //边缘端数据模型服务路由

private static finalStringDATA_EDGE_PATH =System.getenv("iot.hosting.api.domain");

请求参数：

请求参数里的modelId，就是对应的数据模型Id。

1. request.putParam("modelId","value1");

对于边缘端应用，下面两个请求参数可以忽略掉

1. request.putParam("scopeId","value2");
2. request.putParam("appId","value3");

上传文件数据模型接口说明：

下面链接文档提供的接口只是获得了需要上传的文件名称和URL: 获取文件上传地址接口

如返回接口示例：

```
{
"id":"6fr2c332-c1db-417c-aa15-8c5trg3r5d92",
"code":200,
"message":null,
"localizedMsg":null,
"data":{
"fileName":"5269712352e5.jpg",
"url":"https://xxxxx.xxx.xx.com/xxx/file/5269712352e5.jpg?Expires=1557902379&OSSAccessKeyId=xxxx&Signature=xxxx"
}
```

开发者还需将真正需要上传的文件，上传到接口返回值里指定的fileName和url，示例代码如下：

```
// response为上传文件数据模型接口的返回
result = new String(response.getBody(), "UTF-8");
UploadResult uploadResult = JSON.parseObject(result, UploadResult.class);
data nameAndPath = uploadResult.getData();
String url = nameAndPath.getUrl();
HttpClient httpClient = HttpClients.createDefault();
HttpResponse response;
HttpPut put = new HttpPut(url);
//fileBytes：为准备要上传的图片文件
HttpEntity reqEntity = EntityBuilder.create().setBinary(fileByte).build();
put.setEntity(reqEntity);
response = httpClient.execute(put);
```

4 服务依赖方应用开发指导

服务依赖方，例如云端SaaS应用，需要进行服务调用和订阅数据。这里也是列出需要特别关注的点。详细内容请参考：附录1 参考链接：1：边缘应用服务总线对接；3：服务总线

4.1 APPkey and AppSecrect

云端SaaS应用访问服务模型时，请求参数里的Appkey and AppSecrect，可以从物联网应用服务平台的项目详情的开发配置里查看得到，如下图所示



4.2 下载文件数据模型接口：

可以参考边缘应用数据总线对接的下载文件数据模型接口：获取文件下载地址接口

需要注意下面几点：

a: Appkey and Appsecrect：

请参考4.1 里的说明获取

b: filename:

入参列表里缺失了filename，实际上是需要的：

request.putParam("fileName","value1");

filename是从订阅的数据模型消息里获取到的

c: scopeid:

scopeid就是项目id。在物联网应用服务平台里选择打开某个项目后，浏览器地址栏里projectId就是scopeid，如下图所示：



下载文件数据模型接口返回参数里，提供了需下载的文件url:

获取下载的图片路径的demo：

```
public static void main(String[] args) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
                new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey("333464769");
        ioTApiClientBuilderParams.setAppSecret("532ecf2cb6554d46b195e6d240aaf30d");
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);

        IoTApiRequest request = new IoTApiRequest();
        //设置api的版本
        request.setApiVer("0.0.1");
        request.setId("42423423");
        request.putParam("fileName", "03d5999cb80b4cff94cfb8eec1723e54.jpg");
        request.putParam("scopeId", "a124LsoSscaXN9qb"); //项目id
        request.putParam("modelId", "iot_park_pass_record");
        request.putParam("attrName", "plateNumberImage");


        ApiResponse response = syncClient.postBody("api.link.aliyun.com",
                "/data/model/data/download", request, true);
        System.out.println("response code = " + response.getCode()
                + " response = " + new String(response.getBody(), "UTF-8"));


    }
```

```
{
   "id": "6fr2c332-c1db-417c-aa15-8c5trg3r5d92",
    "code": 200,
    "message": null,
    "localizedMsg": null,
    "data": {
        "url": "https://xxxxx.xxx.xx.com/xxx/file/5269712352e5.jpg?Expires=1557902379&OSSAccessKeyId=uyedjYLHDH3DF&Signature=sotMFFIq4RP
%2BWJSDScE8SxvOlvo%3D"
    }
}
```

开发者还需要根据这个url，进行文件下载操作。

4.3 数据查询/数据订阅

前面提到过，云端应用可以通过IoT平台提供的查询数据模型的API接口获取数据，也可以通过数据变更消息订阅方式获取数据。

可以分别参考下面内容实现，只是注意Appkey and Appsecrect需要参考4.1 里的说明得到

数据查询：请参考：查询数据接口

数据订阅：请参考：数据变更消息订阅

5 边缘端应用自测

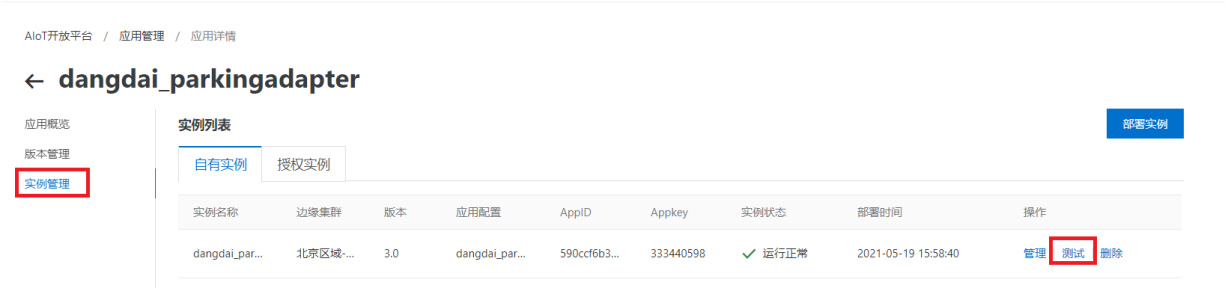
5.1 物联网应用服务平台应用调试接口

应用通过AIoT开放平台创建部署成功后，可以基于平台的应用调试接口，进行服务接口的调用验证：

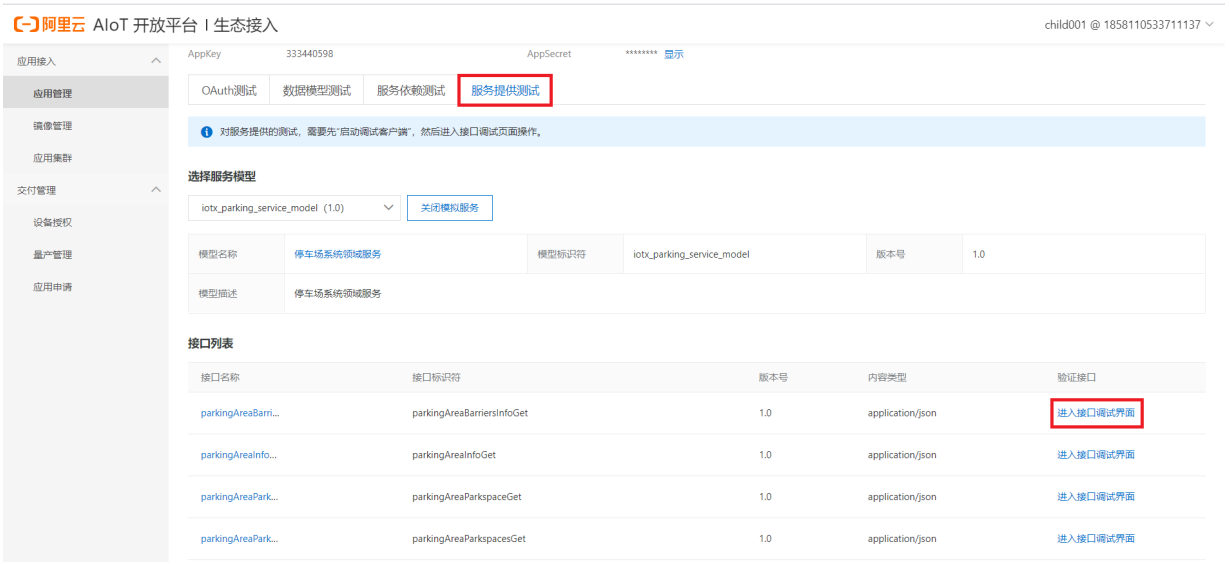
进入应用管理，点击需调试的应用。注意这个应用需要处于已发布状态，如下图所示



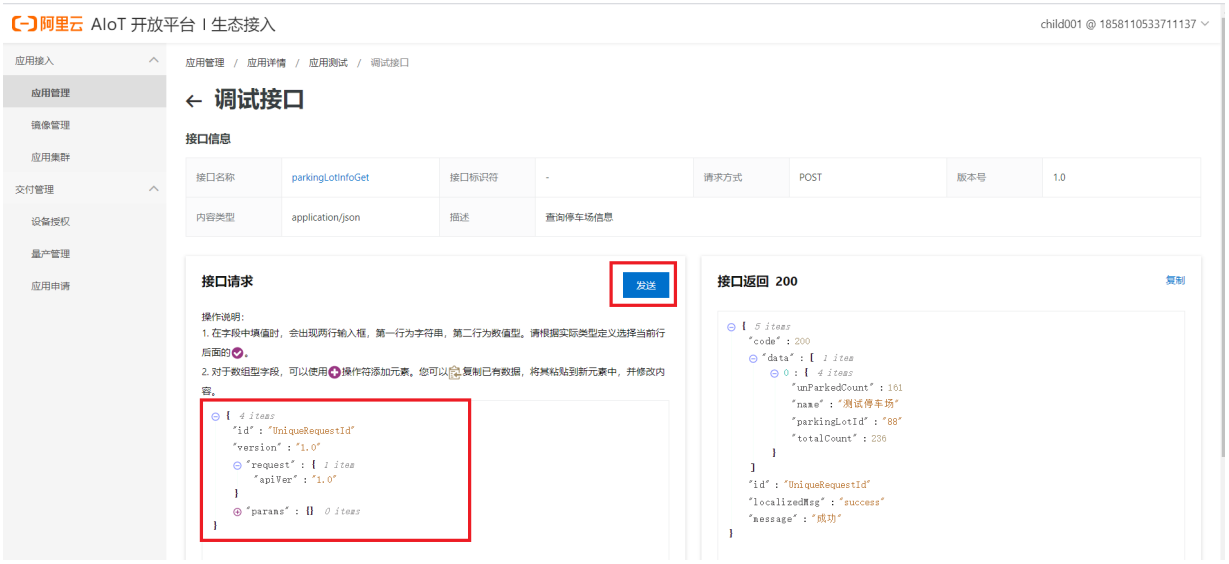
点击**实例管理-测试**：



点击**服务提供测试**，选择一个要调试验证的接口，点击右侧的**进入接口调试界面**



接口请求里，可以根据输入参数要求补充相关参数，然后点击**发送**，然后查看**接口返回**信息，如下图所示：



5.2 模拟本地系统事件上报：Postman

前置条件：

- PC机安装了Postman软件
- 物业管理一体机部署了适配器应用
- PC机需要与物业管理一体机在相同的局域网内

模拟事件上报：

下面示例，以《停车场系统领域模型V3.1-数据模型定义》-车辆通行为例

路径：物业一体机ip:port/具体路径

请求Body内容示例如下：

```
{
    "carCode": "浙A5Cxxx",
    "inTime": "2016-10-18 16:44:44",
    "passTime": "2016-10-28 16:44:44",
    "parkID": "88",
    "inOrOut": "1",
    "GUID": "134589c1d68d44d38dcb7f084b9cf8a1",
    "channelID": "1",
    "channelName": "北大门出口",
    "imagePath": "https://ss1.bdstatic.com\\70cFuXSh_xxx\it\\u=3854694535,624476780&fm=11&gp=0.jpg"
}
```


5.3 模拟本地服务调用：Postman

前置条件：

- PC机安装了Postman软件
- 物业管理一体机部署了适配器应用
- PC机需要与物业管理一体机在相同的局域网内
- 本地部署了相应的系统，例如部署了xxx停车系统

模拟服务调用：

下面示例，以《停车场系统领域服务V3.1-服务模型定义》-1.1 查询停车场信息为例

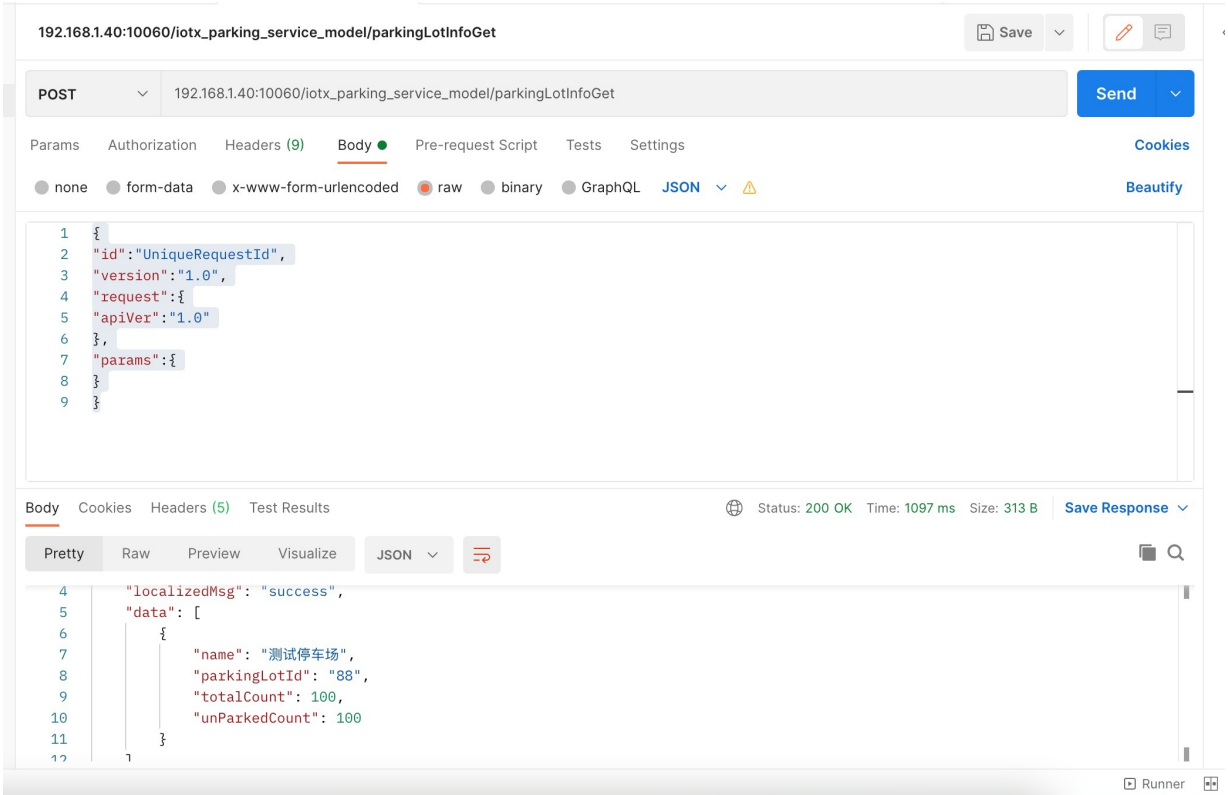
路径： 物业一体机IP:port/服务模型ID/服务接口path

特别注意Header内容：定义 Content-Type 为 application/octet-stream

请求Body内容示例如下：

```
{
 "id":"UniqueRequestId",
 "version":"1.0",
 "request":{
  "apiVer":"1.0"
  },
 "params":{
  }
}
```

Postman示例截图如下。其中192.168.1.40是物业一体机的内网IP地址，需根据实际情况填写，10060是固定的端口号。



6 示例代码

6.1 边缘端适配器应用对接服务模型

以**车辆加入安全黑/白名单**服务为例

1、入口层

```
package com.aliyun.iotx.parkinglot.adapter.web.servicemodelcontroller;

import com.alibaba.fastjson.JSON;


/**
 * 黑白名单
 */
@Slf4j
@RestController
@RequestMapping(value = "/iotx_parking_service_model", method = RequestMethod.POST)
public class BlackWhiteListController {

    @Autowired
    private BlackWhiteListService blackWhiteListService;

    /**
     * 7.1车辆加入安全黑/白名单
     * @param request
     * @return
     */
    @RequestMapping(value = "/parkingLotVehicleListAdd")
    public IoTxResult vehicleAddList(HttpServletRequest request) throws Exception {

        String json="";
        json= new String(readInputStream(request.getInputStream()),"UTF-8");

        //将json数据解析成vo对象接收
        BWListVo bwListVo = JSON.parseObject(json, BWListVo.class);
        BlackWhiteListVo blackWhiteList = bwListVo.getParams();

        //自己业务的处理逻辑
        blackWhiteListService.vehicleAddList(blackWhiteList);

        IoTxResult<Object> ioTxResult = new IoTxResult<>();
        ioTxResult.setData(null);
        IoTxResultUtils.ioTxResultSet(ParkingLotAdapterEnum.SUCCDESS_VEHICLEADDLIST, ioTxResult);
        return ioTxResult;
    }
}

/**
依赖类:
*/

@NoArgsConstructor
@AllArgsConstructor
@Data
class BWListVo implements Serializable {

    private static final long serialVersionUID = -8067179280515471493L;

    /**
     * request里的全局唯一id透传
     */
    private String id;

    /**
     * 请求协议版本
     */
    private String version;

    private Map<String, Object> request;

    private BlackWhiteListVo params;

}


/**
 * 车辆加入安全
 * 黑白名单入参
 */
@NoArgsConstructor
```

```
@AllArgsConstructor
@Data
@ValidateBean
class BlackWhiteListVo implements Serializable {

    private static final long serialVersionUID = 8317834756309144547L;

    @NotBlank(message = "parkingLotId不能为空")
    private String parkingLotId;

    @JsonFormat(with = ACCEPT_SINGLE_VALUE_AS_ARRAY)
    @EachValidate(constraint = NotBlank.class,message = "车牌号不能为空,不能有空字符串")
    private List<String> plateNumber;

    @JsonFormat(with = ACCEPT_SINGLE_VALUE_AS_ARRAY)
    private List<String> areaId;

    @NotBlank(message = "type不能为空")
    private String type;

    private String effectiveDate;
    private String expiryDate;
}
```

6.2 边缘端适配器应用对接数据模型

以**车辆通行**数据模型为例：

车辆通行接口入口

```
package com.aliyun.iotx.parkinglot.adapter.web.datamodelcontroller;

import com.alibaba.cloudapi.sdk.model.ApiResponse;


/**
 * 数据模型
 * 车辆通行
 */
@Slf4j
@RestController
public class IoTParkPassRecordController {

    @Autowired
    private IoTParkPassRecordService parkPassRecordService;

    /**
     * 上传进出记录
     * 图片地址格式
     * @param inOutRecordVo
     * @return
     */
    @RequestMapping(value = "/reportInAndOutRecord", method = RequestMethod.POST)
    public LFResultVo inOutRecordReport(@RequestBody InOutRecordVo inOutRecordVo) {

        String newUrl = UrlFormatUtil.change(inOutRecordVo.getImagePath());
        inOutRecordVo.setImagePath(newUrl);

        ApiResponse apiResponse = parkPassRecordService.inOutRecordReport(inOutRecordVo);

        LFResultVo lfResultVo = new LFResultVo();
        if (apiResponse.getCode() == 200) {
            lfResultVo.setResCode(0);
            lfResultVo.setResMsg("数据上报成功");
        } else {
            lfResultVo.setResCode(1);
            lfResultVo.setResMsg("数据上报失败");
        }
        return lfResultVo;
    }
}
```

可以参考边缘应用数据总线对接

边缘应用数据总线参考使用：

```
public class BlackWhiteList {
    /**
     * 车辆通行上传
```

```
     * @param request
     * @return
     */
    public static void main(String[] args) {
        IoTApiRequest ioTApiRequest = new IoTApiRequest();
        String uuid = UUID.randomUUID().toString();
        String uuidOne = uuid.replace("-", "");

        ioTApiRequest.setId(uuidOne);
        ioTApiRequest.setApiVer("1.0");
        ioTApiRequest.putParam("modelId", "iot_park_pass_record");

        JSONObject properties = new JSONObject();
        properties.put("direction", "获取入参的对应的值");
        properties.put("openType", "获取入参的对应的值");
        properties.put("plateNumber","获取入参的对应的值");

        //首先获取到要上传的文件名和上传的路径
        ApiResponse response = getUploadFileNameAndPath();
        String result = null;
        try {
            result = new String(response.getBody(), "UTF-8");
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }

        //将JSON字符串转化成对象
        UploadResult uploadResult = JSON.parseObject(result, UploadResult.class);
        data nameAndPath = uploadResult.getData();

        properties.put("plateNumberImage", nameAndPath.getFileName());

        //根据底层停车系统上报的文件路径下载文件
        String lfFileUrl = inOutRecordVo.getImagePath();
        //下载后文件的名字
        String name = nameAndPath.getFileName();
        //文件保存路径
        String savePath = fsp.getSavePathOne();

        if(!StringUtil.isEmpty(lfFileUrl)){
            String url = FileUtil.getURL(lfFileUrl);
            FileUtil.downLoadFromUrl(url,name,savePath);
        }

        //根据dop接口获取到的path将文件上传
        //上传文件的路径
        String url = nameAndPath.getUrl();
        //要上传的文件
        File file = new File(savePath + name);
        if(file.exists()){
            byte[] fileBytes = FileUtil.getFileBytes(file);
            boolean b1 = FileUtil.uploadFile(url, fileBytes);
            if (!b1) {
                log.info("文件上传失败");
                throw new ParkingLotAdapterException(502, "文件上传失败", "File upload failed");
            }
        }
        //文件上传成功后,清除本地缓存的文件
        FileUtil.deleteFile(savePath, name);

        properties.put("typePermission", "获取入参的对应的值");
        properties.put("plateColor", "获取入参的对应的值");
        properties.put("plateType", "获取入参的对应的值");
        properties.put("vehicleColor", "获取入参的对应的值");
        properties.put("vehicleType", "获取入参的对应的值");
        properties.put("barrierId", "获取入参的对应的值");

        if (StringUtils.isNotBlank(inOutRecordVo.getChannelName())) {
            properties.put("barrierName", "获取入参的对应的值");
        }

        properties.put("parkingLotId","获取入参的对应的值");
        properties.put("areaId", "未知");
        properties.put("orderNumber", "获取入参的对应的值");
        properties.put("recordId", "未知");

        //数据上报的时间
        Date date = new Date();
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
```

```
        String eventTime = dateFormat.format(date);
        properties.put("eventTime",eventTime);

        ioTApiRequest.putParam("properties", properties);
        ApiResponse apiResponse = syncApiClient.postBody(host, path, ioTApiRequest, "https".equalsIgnoreCase(schema));

    }

    public static ApiResponse getUploadFileNameAndPath() {

        IoTApiRequest request = new IoTApiRequest();

        //设置api的版本
        request.setApiVer("0.0.1");

        // 接口参数
        String uuid = UUID.randomUUID().toString();
        request.setId(uuid.replace("-", ""));

        JSONObject param = new JSONObject();
        param.put("appId",  "应用id");
        param.put("version", "应用版本");

        request.putParam("properties", param);

        //这个参数对应于数据模型中需要上传文件的字段
        request.putParam("attrName", "plateNumberImage");

        //这个参数对应于数据模型的模型id
        request.putParam("modelId", "iot_park_pass_record");

        request.putParam("fileType", "文件类型");
        request.putParam("version", "版本");
        request.putParam("fileSize", "文件大小");

        try {
            ApiResponse apiResponse = syncApiClient.postBody(DATA_EDGE_PATH,
                    "/data/model/data/upload", request, false);
            return apiResponse;
        } catch (IOException e) {
            log.info("上传文件获取文件名和上传路径接口出现异常:{}", e.getMessage());
        }
        return null;
    }

}
```

pom依赖：

```
<dependency>
            <groupId>com.aliyun.iotx</groupId>
            <artifactId>iotx-api-gateway-client</artifactId>
        </dependency>

    <dependency>
            <groupId>com.aliyun.iotx</groupId>
            <artifactId>common-base</artifactId>
        </dependency>
```

6.3 云端应用

服务模型调用示例代码：

```
import java.io.UnsupportedEncodingException;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;

import com.alibaba.cloudapi.sdk.model.ApiResponse;

*/
public class RequestDemo {

    public static void main(String[] args) throws UnsupportedEncodingException {
        postRequestDemo();
    }

    public static void postRequestDemo() throws UnsupportedEncodingException {

        IoTApiClientBuilderParams builderParams = new IoTApiClientBuilderParams();

        builderParams.setAppKey("xxxxxxx");
        builderParams.setAppSecret("xxxxxx");

        SyncApiClient syncClient = new SyncApiClient(builderParams);

        IoTApiRequest request = new IoTApiRequest();

        //设置api的版本
        request.setApiVer("1.0");
        request.setId("42423423");

        //请求参数域名、path、request
        String host = "service-mesh.api-iot.cn-shanghai.aliyuncs.com";
        String path = "/iotx_parking_service_model/parkingLotInfoGet";

        ApiResponse response = syncClient.postBody(host, path, request);
        System.out.println(
            "response code = " + response.getCode() + " response content = " + new String(response.getBody(),
                "utf-8"));
    }
}
```

数据模型订阅消费示例代码：

```
public class Subscribe {


    public static void main(String[] args) throws InterruptedException {

        EventBus eventBus = new EventBus("li");

        //注册所有的订阅
        eventBus.register(new HelloEventListener());

        String appKey = "您的AppKey";
        String appSecret = "您的Secret";
        String endpoint = String.format("https://%s.iot-as-http2.cn-shanghai.aliyuncs.com:443", appKey);
        Profile profile = Profile.getAppKeyProfile(endpoint, appKey, appSecret);
        MessageClient client = MessageClientFactory.messageClient(profile);
        MessageCallback messageCallback = new MessageCallback() {
            @Override
            public Action consume(MessageToken messageToken) {
                Message m = messageToken.getMessage();
                System.out.println("receive : " + new String(messageToken.getMessage().getPayload()));
                JSONObject jsonObject = JSON.parseObject(new String(messageToken.getMessage().getPayload()));
                JSONArray dataIds = jsonObject.getJSONArray("dataIds");
                eventBus.post(dataIds);
                return MessageCallback.Action.CommitSuccess;
            }
        };
        String topic = String.format("/sys/appkey/%s/dop/model/data/change", appKey);
        client.setMessageListener(topic, messageCallback);
        client.connect(messageToken -> {
            System.out.println(messageToken.getMessage());
            return MessageCallback.Action.CommitSuccess;
        });
    }

}

public class HelloEventListener {

    @Subscribe
    public void listen(JSONArray dataIds) {

        System.out.println("receive3 msg:" + dataIds);
        List<Long> objects = dataIds.toJavaList(Long.class);
        System.out.println(objects);
        for (Long object : objects) {
            try {
                getData(String.valueOf(object));
            } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
            }

        }
    }

    public static void getData(String id) throws UnsupportedEncodingException {
        IoTApiClientBuilderParams ioTApiClientBuilderParams =
                new IoTApiClientBuilderParams();
        ioTApiClientBuilderParams.setAppKey("您的AppKey");
        ioTApiClientBuilderParams.setAppSecret("您的Secret");
        SyncApiClient syncClient = new SyncApiClient(ioTApiClientBuilderParams);
        IoTApiRequest request = new IoTApiRequest();
        request.setApiVer("0.0.3");
        request.putParam("modelId", "iot_park_pass_record");
        List<String> returnFields = Lists.newArrayList("plateNumber");
        request.putParam("returnFields", returnFields);
        request.putParam("conditions", JSON.parseArray("[{\"fieldName\": \"id\",\"operate\": \"eq\",\"value\": " + id + "}]"));
        ApiResponse response = syncClient.postBody("api.link.aliyun.com",
                "/data/model/data/query", request, true);
        System.out.println("response code = " + response.getCode()
                + " response = " + new String(response.getBody(), "UTF-8"));
    }

}
```

pom依赖：

```
<dependency>
            <groupId>com.aliyun.openservices</groupId>
            <artifactId>iot-client-message</artifactId>
            <version>1.1.3</version>
        </dependency>

        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-core</artifactId>
            <version>3.7.1</version>
        </dependency>

        <dependency>
            <groupId>commons-codec</groupId>
            <artifactId>commons-codec</artifactId>
            <version>1.11</version>
        </dependency>
```

监听后的返回结果示例：

```
SJ: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
tive : {"dataIds":[49656],"logicalModelIsoId":"oxs_iso_id","modelId":"iot_park_pass_record","operateType":"insert","scopeId":"a124JhMcdAHC8wsy"}
```

getData方法返回的结果：就可以得到刚刚上传的数据了

```
response code = 200 response = {"code":200,"data":"{\"count\":1,\"hasNext\":false,\"items\":[{\"plateNumber\":\"京777777\"}],\"pageNum\":1,\"pageSize\":20}",
"id":"97002182-5bb7-4edd-a361-58776acd0895"}
```

重复消费的问题：

本地客户端的业务代码处理消息的时间过长，大于等于30s。服务端认为你消费失败了，就会向客户端重复推送数据。

附录 如何查看边缘应用日志

应用开发完成并部署到物业一体机进行调试，可以登录到物业一体机，查看对应的应用日志：

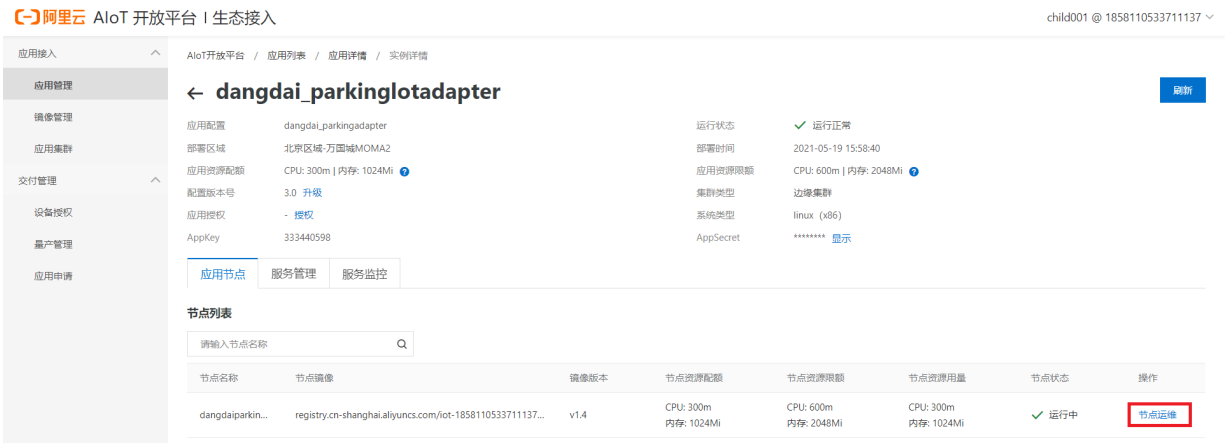
进入应用管理，点击需调试的适配器应用。注意这个应用需要处于已发布状态，如下图所示



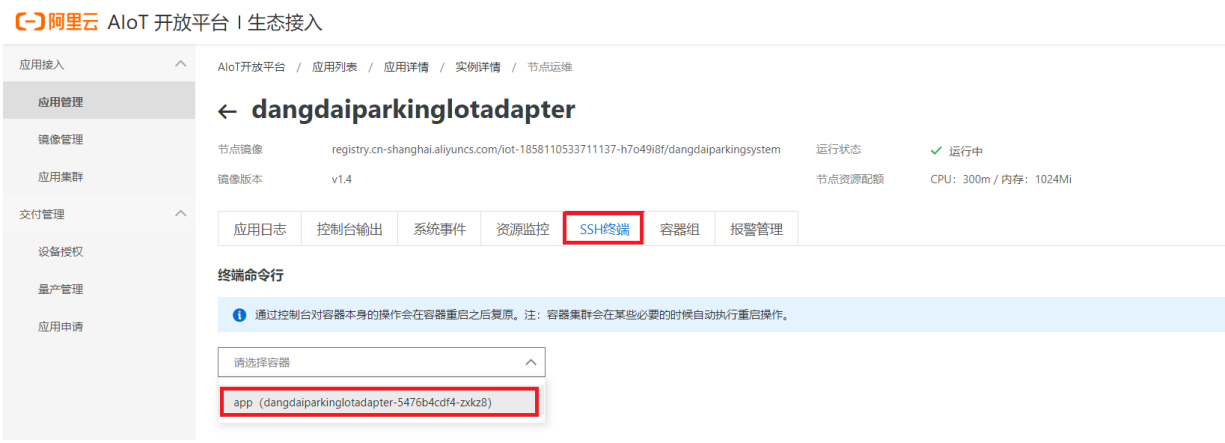
选择**实例管理-管理**



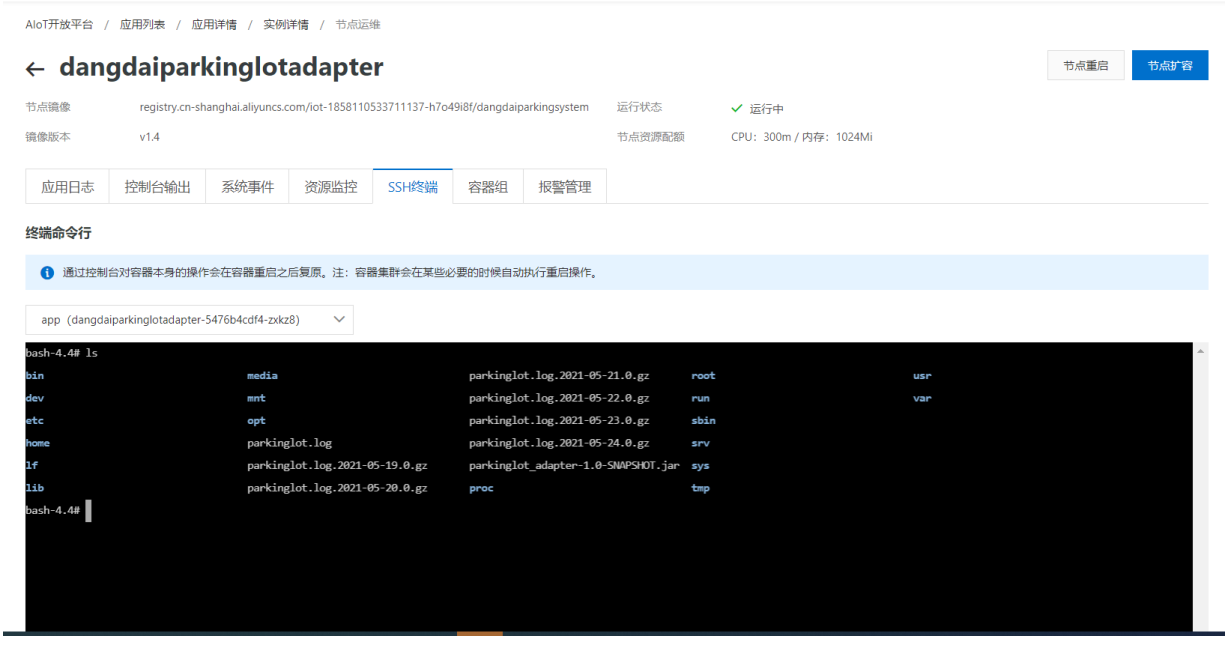
选择**节点运维**



点击SSH终端，选择容器（一般只有一个，点击选择即可）：



选择完容器，将自动登录到物业一体机。当前目录即保持有该应用的日志



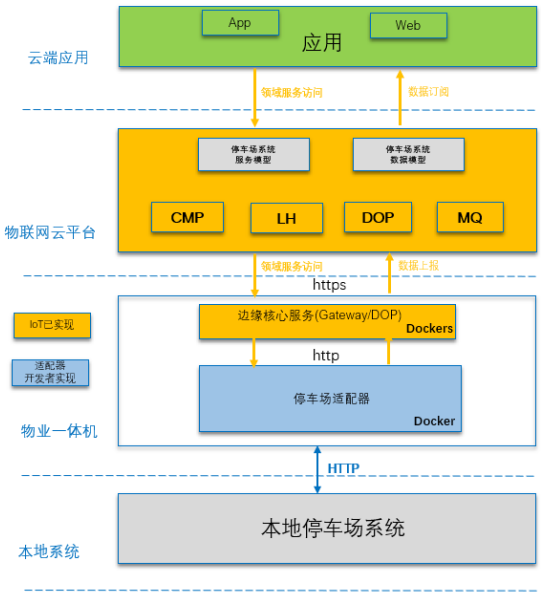
3.5.2.2.2. 停车场管理软件对接方案介绍

1、概述

本文档针对停车场领域服务开发人员，通过本方案集成本地停车场系统，把本地系统、老旧存量系统接入阿里云IoT平台。

2、系统架构图

停车场领域服务对接方案(领域服务对接)



方案说明:
1、物联网云平台与物业一体机通过<停车场系统领域模型>对接
2、物业一体机和本地停车系统，通过本地停车系统对外提供的协议和接口进行对接
3、停车场领域服务提供的功能包括停车场/停车区域查询、道闸控制、停车位管理、车辆及权限管理等功能
4、停车场适配器与边缘核心服务，通过http访问，依赖参数包含appkey, 服务模型/数据模型名称及方法名称

领域服务对接特点:
1、适合本地有复杂的系统及业务逻辑。例如车位管理，收费管理
2、本地适配器应用，通过服务模型为云端应用提供服务，通过数据模型进行数据上报

图解说明:

① 云端应用：停车场系统应用软件。功能包括停车场/停车区域查询，停车位管理，车辆权限管理，黑白名单管理，以及订阅停车场本地系统上报的车辆通行、车位变化、车辆信息变化等事件。该软件基于物联网云平台提供的停车场领域服务模型和数据模型实现。

② 物联网云平台：定义了停车场领域服务模型和数据模型，并将边缘侧提供的停车领域服务和数据消息，向上提供给云端应用。

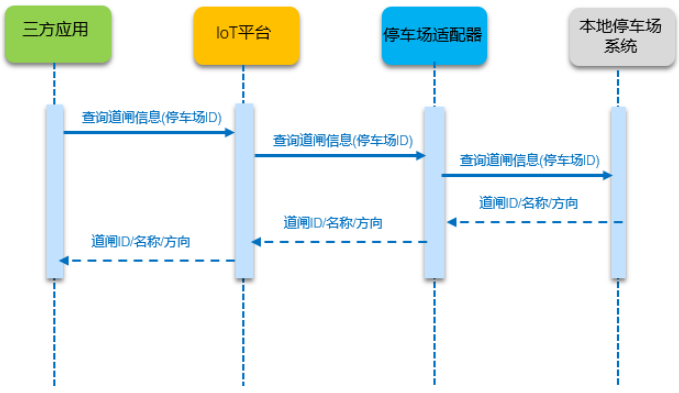
③ 物业一体机：停车场适配器，以docker应用的方式，通过与本地停车场系统对接，可以提供具体的停车场领域服务和数据消息，最后通过物联网平台，提供给云端应用。

④ 本地系统：本地停车场系统。物业一体机里的停车场适配器，需要根据本地停车场系统对外开放的接口，实现与本地停车场系统对接

3、交互流程

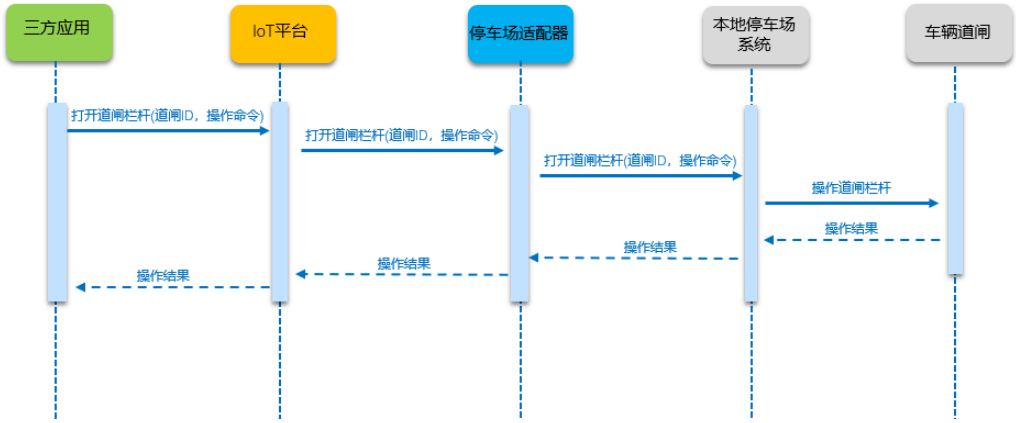
下面以一些典型的场景为例，说明该方案模块间的交互流程

3.1 查询停车场道闸信息

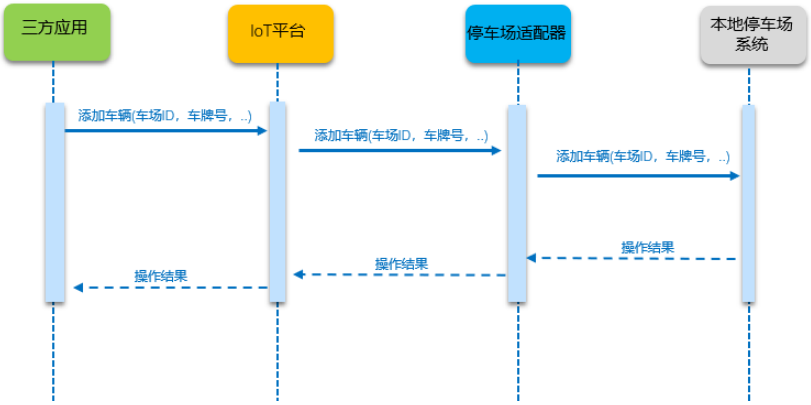


此外，查询区域车位列表，查询停车场信息，查询车辆信息等场景与上面流程类似。

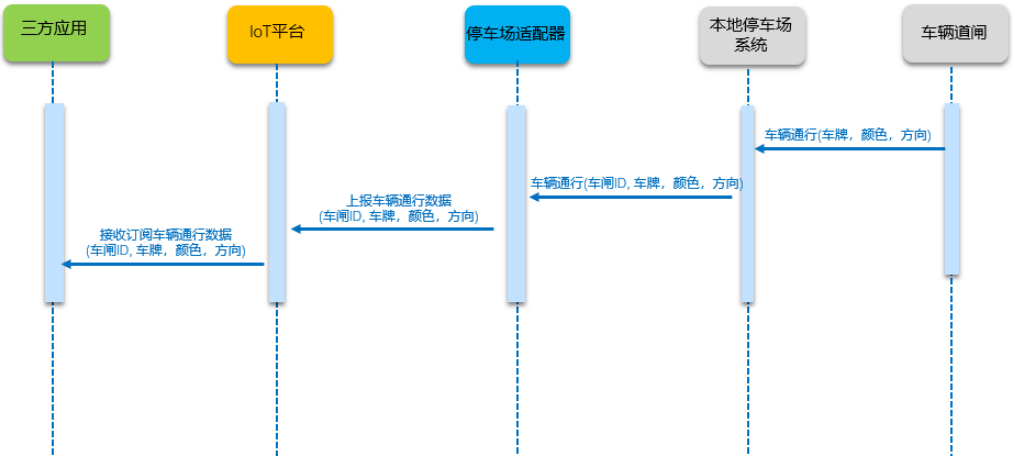
3.2 手动打开道闸栏杆



3.3 添加车辆



3.4 车辆通行事件上报



4、对接流程

4.1 对接说明

运行在物业一体机Docker容器里的停车场适配器应用，需要提供《**停车场系统领域模型**》要求的接口，同时与本地停车场子系统完成对接

4.2 对接前提

本地停车场系统开放接口或者提供对接SDK

4.3 对接开发

- 参考了解数据总线，以及如何向平台操作数据

3.5.2.2.3. 停车场系统领域服务-服务模型定义

请参考 停车场系统领域服务-服务模型定义

3.5.2.2.4. 停车场系统领域模型-数据模型定义

请参考 停车场系统领域模型-数据模型定义

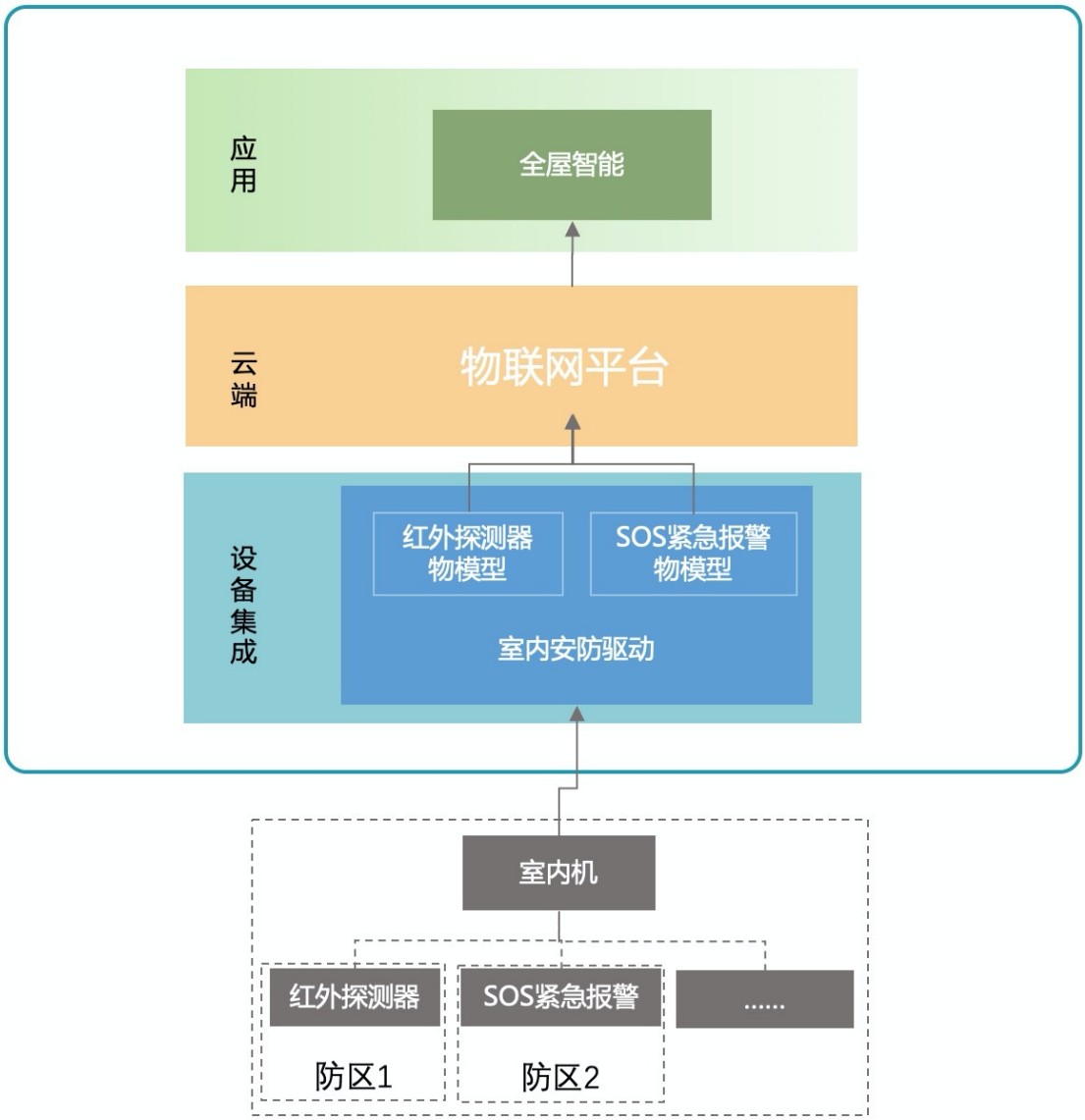
3.5.3. 安防标准

3.5.3.1. 室内安防报警

3.5.3.1.1. 室内安防设备对接边缘服务器接口定义

1 方案介绍

室内各传感器的数据都汇总到室内机上，室内机通过小区局域网直连物业管理一体机，即边缘服务器。传感器产生的报警事件由室内机上报到边缘服务器。边缘服务生成物模型事件并同步到物联网平台



2 接口说明

室内机对接边缘服务器，访问10070端口。

边缘服务器访问室内机的端口由设备厂商自定义，定义后固定不变。

所有HTTP 报文的首部信息中，不能带有Connection: Keep-Alive字段，可以设置为Connection:Close。

物业管理一体机，以下简称“边缘服务器”。

3 服务接口

3.1 服务检测

调用此接口检查服务是否可用。

```
URL：http://[边缘服务器IP 或 室内机IP]:[端口]/alarm/ping
Method：GET
Content-Type：application/json
```

说明

服务启动后开始监听端口数据，当收到ping消息后，立刻回复pong，以表示服务状态正常，否则无消息返回。

请求参数

无。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。固定内容。
message	String	pong	调用成功时，返回“pong”。固定内容。

示例

正常返回示例

```
{
    "code": 0,
    "message": "pong"
}
```

3.2 建立连接

室内机向边缘服务器发送连接请求。

```
URL：http://[边缘服务器IP]:[端口]/alarm/connect
Method：POST
Content-Type：application/json
```

说明

前置条件：

室内机先预置安全密钥（RSA公钥），再进行连接。

建立连接交互流程：

1. 室内机携带下面的请求参数向边缘服务器发起连接请求。
2. 边缘服务器校验ip参数是否已经配置，如未配置，拒绝连接请求。设备信息配置需要预先在云端配置好，并部署到边缘。
3. 如果配置校验通过，边缘服务器返回成功消息给室内机。响应消息中包含一个有效的token。
4. 边缘服务器向室内机发送“服务检测”ping请求，室内机返回pong消息给边缘服务器，连接建立完成。
5. 如果室内机没有收到成功消息和token，等待一分钟后，再次发起连接请求。

connect请求里，如果参数encrypted为1，那么参数ciphertext的内容为使用RSA公钥加密的密文。下面是生成密文和发送密文的方法：

1. 室内机先随机生成一个长度为16字节的AES Secret Key，再使用预置的RSA公钥对AESSecret Key进行加密，把生成的密文发送给边缘服务器。
2. 边缘服务器使用RSA私钥解密出AES Secret Key并保存，在后续通信中，用它对数据进行加解密。
3. 连接建立成功后，室内机发送的数据都使用AES Secret Key进行加密，室内机收到的数据也是使用AES Secret Key加密的。

说明：

*RSA*的加密模式为：*ECB*；填充模式为：*PKCS1Padding*。（JAVA为：RSA/ECB/PKCS1Padding）

*AES*长度128位，加密模式为：*ECB*；填充模式为：*PKCS7Padding*。（JAVA为：AES/ECB/PKCS5Padding）

限制

调用该接口的每秒请求数（QPS）最大限制为10。

请求参数

名称	类型	是否必选	示例值	描述

version	String	是	2021-01-01	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-01-01`。接口可能存在多个版本。
protocol	String	是	2021-01-01	设备的通信协议的版本号，为日期形式：`YYYY-MM-DD`，固定值为`2021-01-01`。 边缘服务器发起请求时，使用此参数作为version，即version为`2021-01-01`。
sn	String	是	123456789	设备的唯一编码。长度为4-32个字符，可以包含英文字母、数字和特殊字符：连字符（-）、下划线（_）、at符号（@）、点号（.）、和英文冒号（:）。
vendor	String	是	DEV	设备厂商名称。
model	String	否	DEV-HW-210	设备硬件型号。
time	Integer	是	1589472000	请求的时间戳。以秒为单位，需要使用UTC时间。
ip	String	是	192.168.1.128	连接发起方的IPv4地址。
mac	String	是	00-16-EA-AE-3D-40	连接发起方的MAC地址。
port	int	是	10000	边缘服务器访问设备服务的端口，如果不提供，填写-1。
keepalive	Integer	是	30	心跳最大间隔。单位：秒，合法取值范围[10，120]。
encrypted	Integer	是	1	是否采用数据加密模式。 • 0：不加密。 • 1：加密。
ciphertext	String	否	mrmZ2Amm5TNDjlxA+ssNt23QTcbcQNKxzMc+YDoewHkR1eOT2+fk=	如果encrypted为1，需要传递此参数。先随机生成一个长度为16字节的AES Secret Key，再使用预置的RSA公钥对Secret Key进行加密，再把密文进行Base64编码得到的结果就是ciphertext的内容。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

data	JSONObject		调用成功时，返回内容定义如下。

名称	类型	示例值	描述
token	String	6f2a-***-1-128	连接成功后，由接收连接的server端颁发的连接成功标识，代表一个有效连接，最大长度位64字节。在后续client的请求中需要携带，如“心跳保活”。

示例

请求示例

```
{
    "version": "2021-01-01",
    "protocol": "2021-01-01",
    "sn": "123456789",
    "vendor": "DEV",
    "model": "DEV-HW-210",
    "time": 1589472000,
    "ip": "192.168.1.128",
    "mac": "00-16-EA-AE-3D-40",
    "port": 10000,
    "keepalive": 30,
    "encrypted": 1,
    "ciphertext": "hlbtFix0/wrcca8SqUcy+5Q8J76xOM29fp4itQhstZhAQRRi+RITQPupSSnrZCXAGzPHYUdbKWijcMa0FGwAoPqw4m0Mot1HT3qKxMBHTvSGuOEayozZJ
Mj18FL0tceGZ2V5BxkethBDsRTfn2qaDEMCaS9+go0dqj1aP0IjrYg="
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success",
    "data":{
        "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128"
    }
}
```

3.3 心跳保活

室内机向边缘服务器定期发送心跳消息，以确保消息通道的连通性，维持设备的在线状态。

```
URL：http://[边缘服务器IP]:[端口]/alarm/keepalive
Method：POST
Content-Type：application/json
```

说明

建立连接后，服务器会向客户端颁发一个token，心跳消息中需要携带此token。当发送心跳时，返回token无效，那么需要重新建立连接，获取有效token。

成功时，返回的消息体中包含一个时间参数，可以用来同步双方时间。

限制

调用该接口的每秒请求数（QPS）最大限制为100。

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-01-01	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-01-01`。接口可能存在多个版本。
token	String	是	6f2a-***-1-128	有效连接的标识。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。
data	JSONObject		调用成功时，返回内容定义如下。

名称	类型	示例值	描述
time	String	1591069176000	响应消息生成的时间，时间戳以毫秒为单位，需要使用UTC时间。

示例

请求示例

```
{
    "version": "2021-01-01",
    "token": "6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128"
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success",
    "data":{
        "time": "1591069176000"
    }
}
```

3.4 上报防区信息和状态

室内机上报主机下的防区信息和防区状态给边缘服务器。

```
URL：http://[边缘服务器IP]:[端口]/alarm/reportZoneInfo
Method：POST
Content-Type：application/json
```

说明

每次建立连接后，上报一次防区信息和状态。当防区信息或状态变化时，立即上报实时的防区信息和状态。

限制

调用该接口的每秒请求数（QPS）最大限制为100。

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-01-01	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-01-01`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。
zoneInfo	JSONArray	是		防区信息。
• zoneID	String	是	01	防区的唯一标识。

• zoneDeviceType	Integer	是	1	防区下的探测器类型，枚举值见附录。
• zoneState	Integer	是	1	防区布防状态，枚举如下： • 0：未知 • 1：撤防 • 2：布防

加密的

名称	类型	是否必选	示例值	描述
version	String	是	2021-01-01	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-01-01`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。
ciphertext	String	是	ewogICAgJ2EiJzoiYiIKfQ==	先将zoneInfo对应的内容转换为json字符串，再使用AES密钥进行加密，最后把密文Base64编码。 例如： "zoneInfo":abc，abc就是需要加密的内容。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "version":"2021-01-01",
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "zoneInfo":[
        {
            "zoneID":"01",
            "zoneDeviceType":1,
            "zoneState":1
        },
        {
            "zoneID":"02",
            "zoneDeviceType":3,
            "zoneState":2
        }
    ]
}
```

加密的

```
{
    "version":"2021-01-01",
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "ciphertext":"ewogICAgJ2EiJzoiYiIKfQ=="
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

3.5 设置防区布防状态

边缘服务器对室内机下的指定防区进行设防或撤防操作。

```
URL: http://[室内机IP]:[端口]/alarm/setZoneAlertState
Method: POST
Content-Type: application/json
```

说明

可以对一个或多个防区进行布防或撤防操作。

限制

调用该接口的每秒请求数（QPS）最大限制为100。

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-01-01	为connect接口传递的protocol参数值。
zoneInfo	JSONArray	是		防区信息。
• zoneID	String	是	01	防区的唯一标识。
• zoneState	Integer	是	1	防区状态，枚举如下： • 1：撤防 • 2：布防
• time	String	是	1591069176000	设置操作的时间，时间戳以毫秒为单位，需要使用UTC时间。 为了防止重放攻击，室内机需要记录这个时间，如果收到请求的时间参数等于或小于最后记录的时间，则抛弃这个请求。

加密的

名称	类型	是否必选	示例值	描述
version	String	是	2021-01-01	为connect接口传递的protocol参数值。
ciphertext	String	是	ewogICAgJ2EiJzoiYiIKfQ==	先将zoneInfo对应的内容转换为json字符串，再使用AES密钥进行加密，最后把密文Base64编码。 例如： "zoneInfo":abc，abc就是需要加密的内容。

返回参数

名称	类型	示例值	描述

code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "version":"2021-01-01",
    "zoneInfo":[
        {
            "zoneID":"01",
            "zoneState":1,
            "time":"1591069176000"
        },
        {
            "zoneID":"02",
            "zoneState":2,
            "time":"1591069176000"
        }
    ]
}
```

加密的

```
{
    "version":"2021-01-01",
    "ciphertext":"ewogICAgJ2EiJzoiYiIKfQ=="
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

3.6 防区报警

室内机发送防区设备报警信息给边缘服务器。

```
URL：http://[边缘服务器IP]:[端口]/alarm/reportAlarm
Method：POST
Content-Type：application/json
```

说明

当防区设备触发报警时，根据报警触发规则，室内机上报报警信息给边缘服务器。

限制

调用该接口的每秒请求数（QPS）最大限制为100。

请求参数

名称	类型	是否必选	示例值	描述
version	String	是	2021-01-01	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-01-01`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。
alarmInfo	JSONArray	是		报警信息。
• zoneID	String	是	01	防区的唯一标识。

• alarmTime	String	否	1591069176000	报警事件发生的历史时间，如果无此参数，表示事件发生在当前时间。 时间戳以毫秒为单位，需要使用UTC时间。

加密的

名称	类型	是否必选	示例值	描述
version	String	是	2021-01-01	API版本号，为日期形式：`YYYY-MM-DD`，最新版本为`2021-01-01`。接口可能存在多个版本。
token	String	是	6f2a-***-128	有效连接标识。
ciphertext	String	是	ewogICAgJ2EiJzoiYiIKfQ==	先将alarmInfo对象转换为json字符串，再使用AES密钥进行加密，最后把密文Base64编码。 例如： "alarmInfo":abc，abc就是需要加密的内容。

返回参数

名称	类型	示例值	描述
code	Integer	0	调用成功时，返回0。失败时，返回的错误码。详见错误码表。
message	String	success	调用结果描述。

示例

请求示例

```
{
    "version":"2021-01-01",
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "alarmInfo":[
        {
            "zoneID":"01"
        },
        {
            "zoneID":"02",
            "alarmTime":"1591069176000"
        }
    ]
}
```

加密的

```
{
    "version":"2021-01-01",
    "token":"6f2ae65e-d7f3-11ea-9a16-6c96cfdfd355-192-168-1-128",
    "ciphertext":"ewogICAgJ2EiJzoiYiIKfQ=="
}
```

正常返回示例

```
{
    "code": 0,
    "message": "success"
}
```

附录

探测器类型

```
1.SOS紧急按钮
2.红外
3.门磁
4.窗磁
5.瓦斯
6.烟雾
7.水浸
8.震动
9.气压
```

3.5.3.2. 周界报警

3.5.3.2.1. 周界报警主机对接方案介绍

1、概述

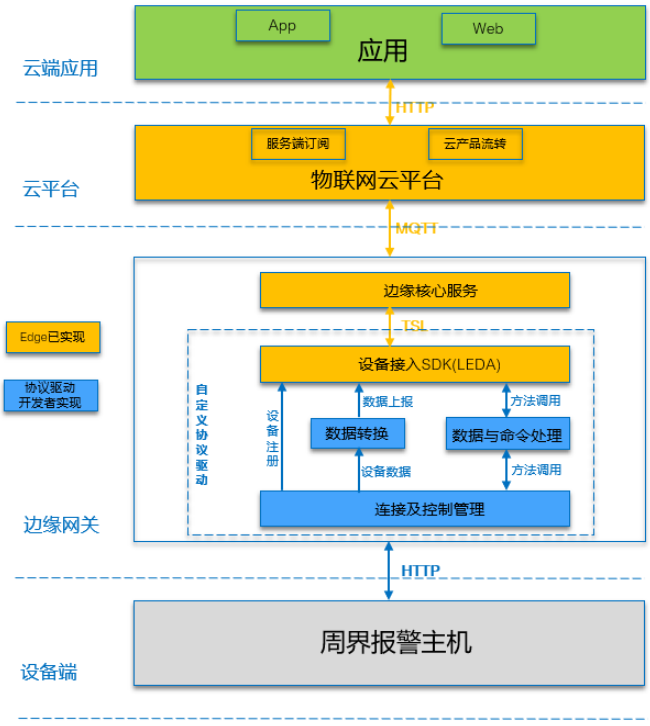
1.1 编写目的

本文档为周界报警主机设备厂商的技术人员，介绍周界报警主机接入到边缘网关(物业一体机)的对接方案和设备驱动的开发流程。

1.2 名词解释

名词	解释
Link IoT Edge	物联网边缘计算产品（Link IoT Edge，简称LE），即阿里云物联网平台（IoT）中的边缘计算产品。提供安全可靠的数据计算能力，可供本地处理设备数据，减少上传云端的成本。
网关	运行Link IoT Edge软件的计算设备统称为边缘网关或边缘服务器，简称网关。
子设备	指通过一定的协议或接口接入到Link IoT Edge网关上的设备（即设备接入到网关后称为子设备），网关代理该子设备与云端进行通信。
驱动	Link IoT Edge中的设备接入模块称为驱动（driver）或设备接入驱动。所有连接到Link IoT Edge的设备都需要通过驱动实现接入。

2、系统架构图



图解说明：

① 云端应用：周界报警应用软件。功能包括通道设置、通道状态查询，通道告警事件接收和处理等，该软件基于云平台提供的物模型相关的服务端订阅和云产品流转，实现对设备上报事件的处理及对设备的控制。

② 云平台：物联网云平台，向自定义驱动提供物模型。

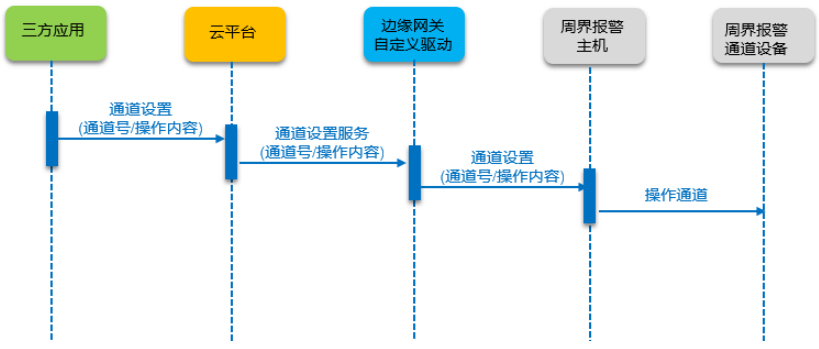
③ 边缘网关：边缘网关内含有自定义协议驱动。驱动运行在LE组件中，负责管理及控制设备，其中包含阿里云提供的设备接入SDK（LEDA）部分和开发者实现部分。

④ 设备端：周界报警主机及下面接入的通道设备。设备供应商应提供设备通信协议，供驱动访问或控制设备使用。设备供应商应提供硬件配置工具，可以进行设备初始化、设备参数配置及故障定位功能。

3、交互流程

3.1 通道设置

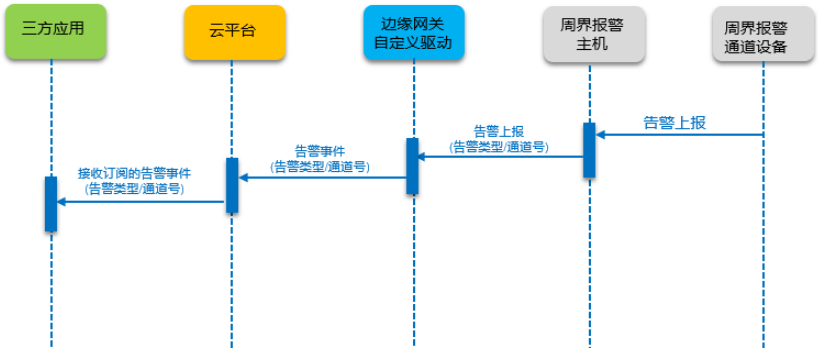
云端应用对指定的通道进行布防或撤防通道设置时，交互流程如下图所示



此外，通道列表查询，通道状态查询，消除报警的流程，与上面类似

3.2 通道告警

周界报警的通道设备出现告警时，例如入侵告警、触网告警等，将上报给周界报警主机。周界报警主机将出现告警的通道号及告警类型通知边缘网关驱动，最后上报到应用，交互流程示意如下：



此外，告警解除事件的上报流程，与上图的通道告警事件上报流程类似

4、对接流程

4.1 明确方案分工

- 根据系统架构图的内容，进行方案可行性评估，理解自定义协议驱动所处的位置。
- 明确各环节项目分工——需要设备端技术人员和协议驱动开发人员明确各自负责内容。
- 明确自定义协议驱动需要实现的功能，参考对应设备的物模型（请参考《周界报警设备物模型》文档的物模型介绍）。
- 明确自定义协议驱动与设备的通信方式，需要设备厂商提供设备端SDK，使自定义协议驱动可以基于该SDK实现访问设备数据和下发控制命令的功能。

4.2 信息收集

- 提供设备的硬件型号、设备ID

4.3 驱动开发

- 参考《周界报警主机驱动开发指导》文档，了解物联网设备接入，熟悉LEDA接口，熟悉实现驱动的基本方式，开发自定义协议驱动。
- 自定义协议驱动需通过驱动日志确认设备向云平台注册上线的接口被调用，并保证设备持续稳定地处于在线状态。
- 自定义协议驱动需通过驱动日志确认设备上报事件的接口被调用，通过驱动日志确认服务方法接口被调用。

在物联网应用服务平台，新创建的驱动英文名称定义有下面的规范建议：

- 驱动名称包含内容：

厂商名称_厂商产品型号_适用领域_驱动功能

如果这个驱动是通用驱动，适用所有厂商，则驱动名称里可以去掉厂商名称和厂商产品型号；

如果这个驱动支持某个厂商的所有产品型号，则驱动名称里可以去掉厂商产品型号。

- **适用领域**

  说明：

人行：PerAccess (personal access)

车行：VehAccess (vehicle access)

EBA：EBA

安防：SecSystem（security system）

- **驱动功能**

  说明：

例如EBA领域的Modbus功能，例如人行领域的可视对讲功能。

- **驱动名称**

  使用大驼峰命名规范，即每个单词的首字母都要求是大写

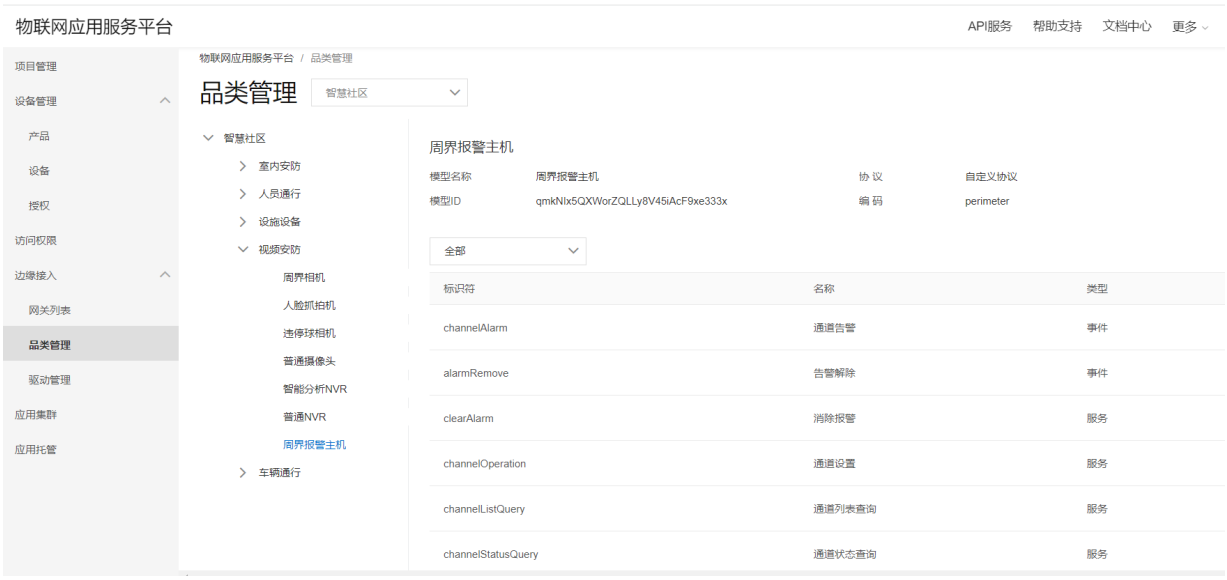
- 举例说明：

有一个驱动，可以支持广拓公司（Goto）的A7型号的周界报警主机设备，驱动名称可定义为：

Goto_A7_SecSystem_Perimeter

5. 物模型

请参考物联网应用服务平台里**品类管理-周界报警主机**，如下图所示



3.5.3.2.2. 周界报警主机驱动开发指导

请参考 自定义协议驱动开发指导

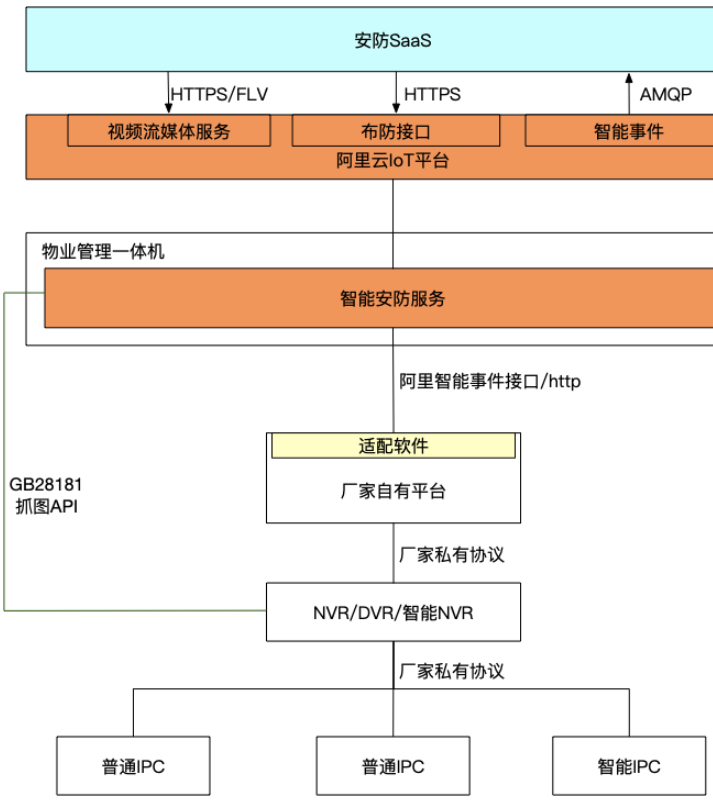
3.5.3.3. 监控设备

3.5.3.3.1. 安防设备接入方案

1 物模型说明

1. 方案介绍

本方案的目的是将不同品牌的安防设备进行统一管理，接入到阿里云平台。接入功能包括安防基础功能和智能事件。基础功能包括：设备管理、设备抓拍、云台控制、视频直播、录像点播。智能事件包括：规则布防、人脸识别、区域入侵、电动车进电梯、垃圾桶满溢、高空抛物、消防通道占用、热成像人体测温、热成像温度异常、电动车进出单元门、违停检测、人员徘徊、人员摔倒。事件支持动态扩展。



2. 设备接入

2.1 基础功能

2.1.1 非抓拍功能

功能：设备管理、云台控制、视频直播、录像点播

接入方式：GB28181

协议版本：GB/T28181-2016

默认参数：

GB28181 SIP服务器编号：34020000002000000001

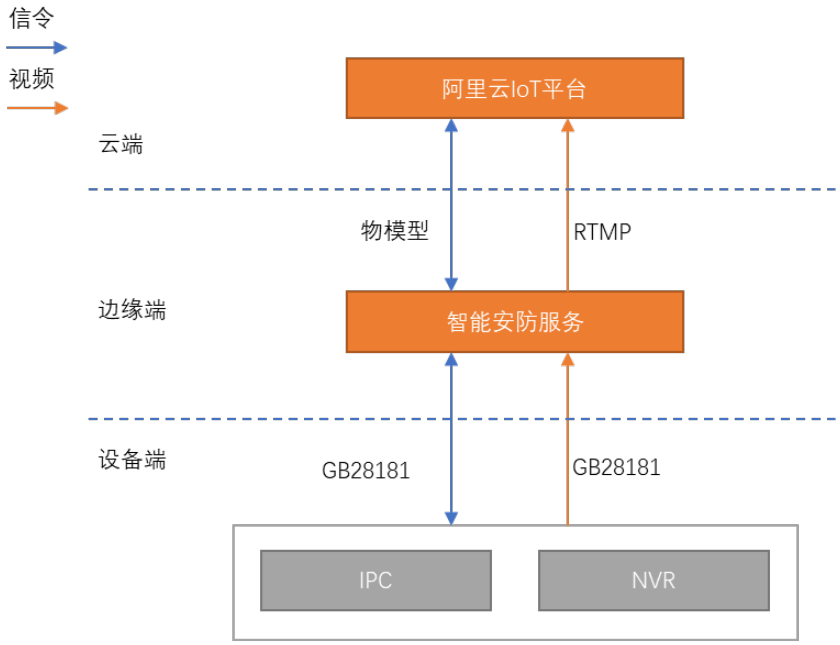
GB28181 SIP域：3402000000

GB28181 SIP服务器端口：5060

接入流程：

i. 边缘视频网关的智能安防服务中开启了GB28181的server服务，安防设备以GB28181 client的方式进行接入。

ii. 在本地配置安防设备的GB28181 SIP配置参数，智能安防服务的默认参数见上述“默认参数”。



2.1.2 抓拍功能

功能：设备抓拍

接入方式：私有SDK

接入流程：

a. 智能安防服务通过设备厂商的私有SDK进行图片抓拍。设备端需保证支持抓拍功能，无需做适配和修改。

2.2 智能事件

接入方式：HTTP API

默认参数：

HTTP端口：5080

接入流程：

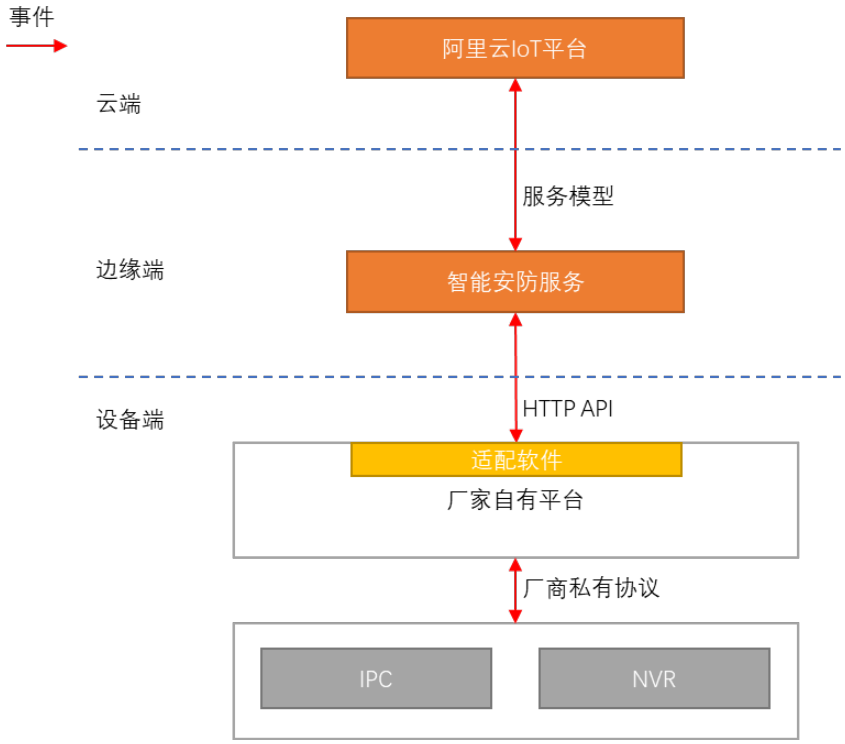
(1) 边缘视频网关的智能安防服务开启了本地HTTP server，智能事件通过HTTP API进行上报。事件API定义见《智能安防服务-边缘事件管理API》文档。

(2) 设备厂商负责实现“适配软件”，管理事件规则和上报事件。

(3) 适配软件向智能安防服务主动发起认证，认证通过后可以进行API交互，如果API交互返回“认证失败”或“未认证”的错误码，需重新发起认证。

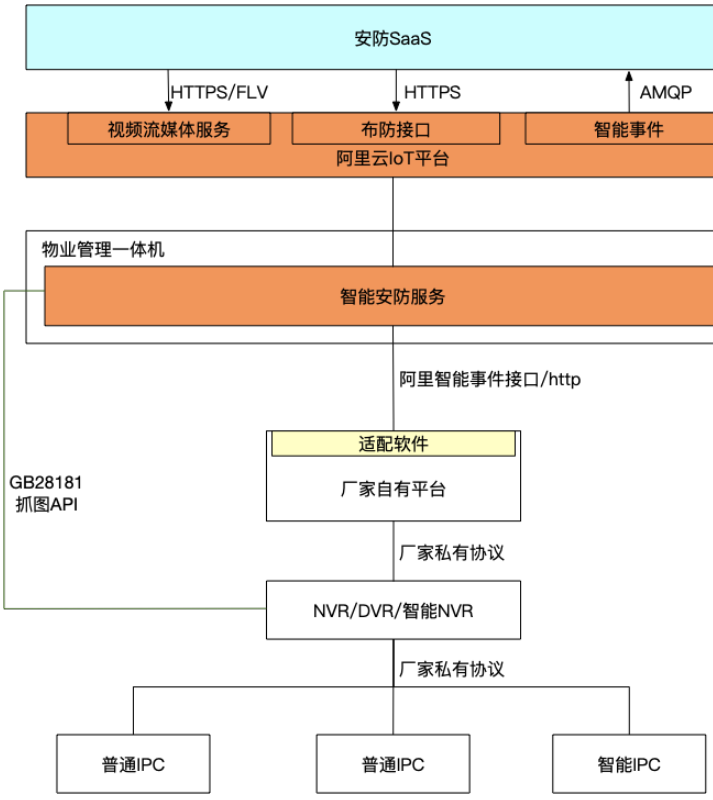
(4) 事件上报：安防IPC/NVR产生事件后，通过私有协议上报至适配软件，适配软件通过HTTP API上报至边缘视频网关。

(5) 规则下发：云端saas的事件规则，通过阿里云IoT平台的服务模型下发到边侧智能安防服务，智能安防服务通过HTTP API下发给适配软件。



3.5.3.3.2. 安防设备实施手册

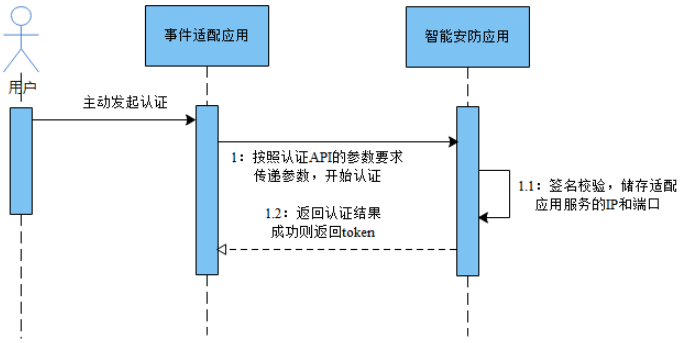
1. 架构图



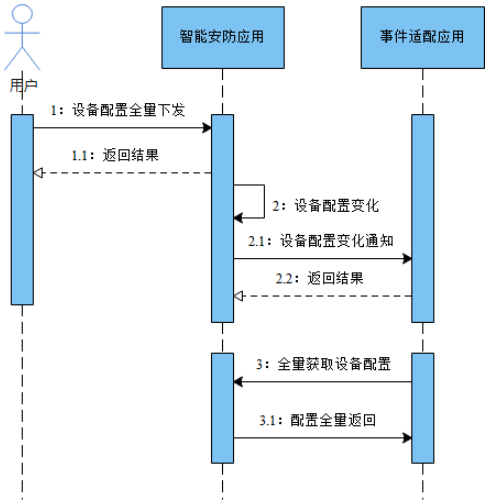
2. 流程图

2.1 应用认证

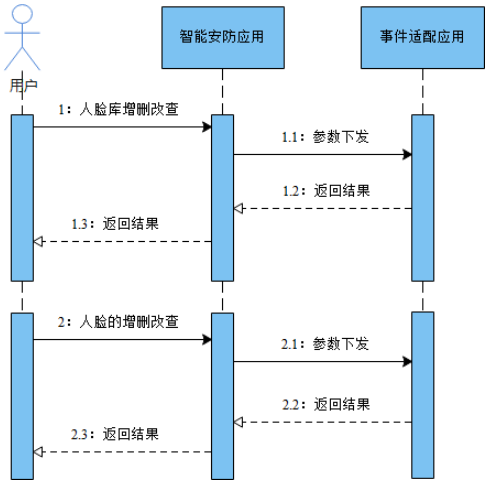
说明：事件适配应用向智能安防应用主动发起认证，认证成功后，智能安防应用返回令牌信息（token），智能安防应用接受带有此token的HTTP请求，同时，事件适配应用也接受来自智能安防应用带有此token的HTTP请求。



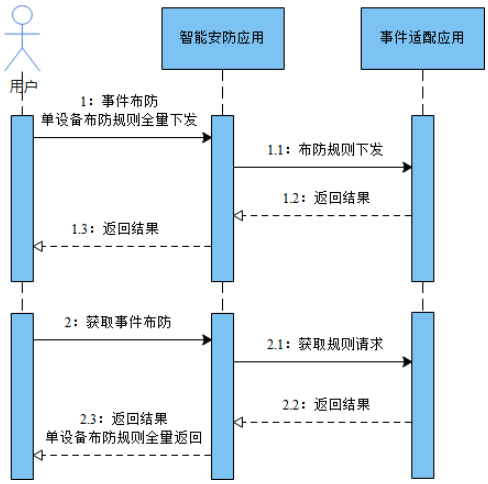
2.2 设备配置



2.3 人脸管理

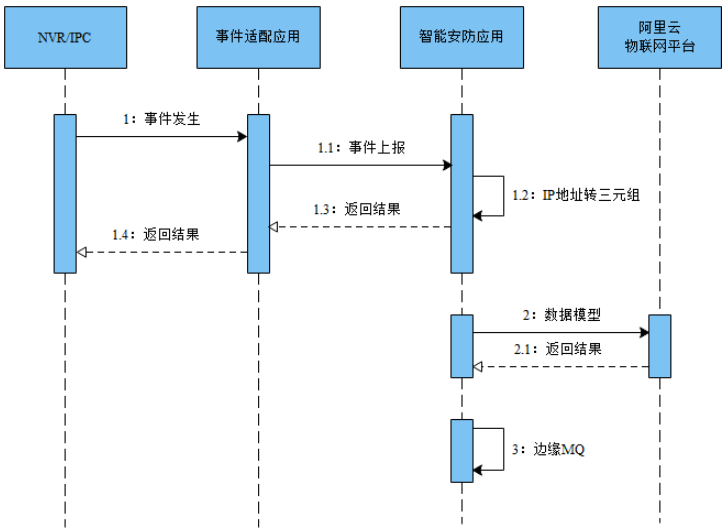


2.4 事件布防

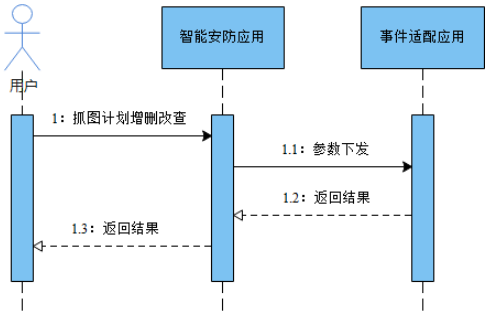


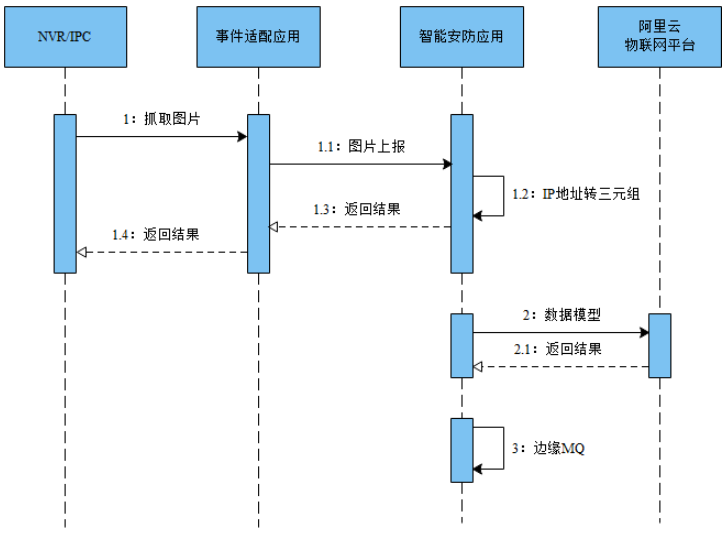
说明：

NVR/IPC产生告警事件后，安防应用负责管理告警事件。

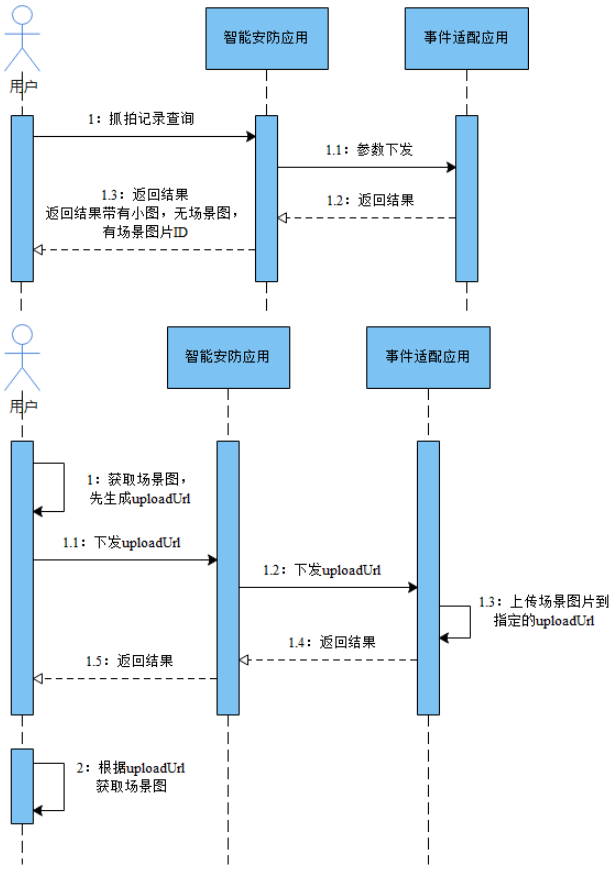


2.5 抓图计划





2.6 抓拍记录查询



3. API

智能安防应用端口号：5080

事件适配应用IP地址和端口号：事件适配应用使用前，需向阿里云IOT后台小二提交端口审核，审核通过后端口方可使用。IP地址和端口号通过认证API的参数传递给智能安防应用。

3.1 接入认证

- 名称：edgeSecurityEvent/appAuthentication
- 方向：事件适配应用->智能安防应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
ip	必填	String	连接发起方的IPv4地址
mac	必填	String	连接发起方的MAC地址 00-**-3D-40
time	必填	Long	请求的时间戳。以秒为单位，需要使用UTC时间。
manufacture	必填	String	品牌 DH：大华 HIK：海康 UNIVIEW：宇视 TIANDY：天地伟业
signMethod	必填	String	签名方式，目前支持MD5 取值范围 MD5 SHA1 SHA256
sign	必填	String	需要签名的内容封装格式： "TIME{time}MAC{mac}IP{ip}MANUFACTURE{DH}"大括号内的是具体内容，其外面的是名称。备注：time要转换成string类型。
alarmAdapterPort	必填	Integer	事件适配应用的端口号 智能安防应用作为发起方，此参数省略

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	Json	数据
+token	String	令牌信息

- 示例代码

请求

```
{
  "ip":"192.168.1.2",
  "mac":"00-**-3D-40",
  "time":1589472000,
  "manufacture":"DH",
  "signMethod":"md5",
  "sign":"4db511**42",
  "alarmAdapterPort":10080
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "token":"cc85****0001"
  }
}
```

3.2 设备管理

- 名称：edgeSecurityEvent/deviceManage/getConfig
- 方向：事件适配应用->智能安防应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
manufacture	必填	String	品牌 DH：大华 HIK：海康 UNIVIEW：宇视 TIANDY：天地伟业

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSONArray	数据
+manufacture	String	品牌 DH：大华 HIK：海康 UNIVIEW：宇视 TIANDY：天地伟业
+deviceType	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC
+ip	String	设备的IP地址，deviceType为3时，为NVR的IP
+port	Integer	设备的端口号，deviceType为3时，为NVR的PORT
+username	String	登录用户名，deviceType为3时，为NVR的username
+password	String	登录密码，deviceType为3时，为NVR的password
+channel	Integer	通道号，从0开始，deviceType为3时有效

+childIp	String	deviceType为3时有效，子设备IP
+childPort	Integer	deviceType为3时有效，子设备PORT
+childUsername	String	deviceType为3时有效，子设备username
+childPassword	String	deviceType为3时有效，子设备password
+extInfo	String	扩展字段

- 示例代码

请求

```
{
  "token":"cc85****0001",
  "manufacture":"DH"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":[{
    "manufacture":"DH",
    "deviceType":1,
    "ip":"192.168.1.2",
    "port":20080,
    "u********e":"*********",
    "p********d":"*********"
  },{
    "manufacture":"DH",
    "deviceType":2,
    "ip":"192.168.1.3",
    "port":20080,
    "u********e":"*********",
    "p********d":"*********"
  },{
    "manufacture":"DH",
    "deviceType":3,
    "ip":"192.168.1.3",
    "port":20080,
    "u********e":"*********",
    "p********d":"*********",
    "childIp":"192.168.1.3",
    "childPort":20080,
    "childU********e":"*********",
    "childP********d":"*********",
    "channel":0
  }]
}
```

- 名称：edgeEventAdapter/deviceManage/configChange
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息

- 响应参数

参数	类型	说明

code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
}
```

返回

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

- 名称：edgeEventAdapter/deviceManage/getStatus
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JsonArray	数据
+ip	String	设备地址，deviceType为3时，ip为NVR的IP
+channel	Integer	通道号，从0开始，deviceType为3时有效
+deviceType	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC
+status	Integer	连接状态 0：连接异常 1：连接正常
+unnormalInfo	String	异常信息，自定义异常信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":[{
    "ip":"192.168.1.2",
    "deviceType":2,
    "status":1
  },{
    "ip":"192.168.1.3",
    "deviceType":2,
    "status":1
  },{
    "ip":"192.168.1.4",
    "channel":1,
    "deviceType":3,
    "status":0,
    "unnormalInfo":"password error"
  }]
}
```

3.3 人脸管理

- 名称：edgeEventAdapter/face/createGroup
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC（NVR下IPC）
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
groupName	必填	String	人脸库名称（最大字符长度63）
groupType	必填	Integer	人脸库类型 1：黑名单 2：白名单 3：员工 4：访客 100：其他
extInfo	可选	String	扩展信息

- 响应参数

参数	类型	说明

code	Integer	错误码
message	String	信息
data	JSON	数据
+groupId	String	人脸库ID

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupName":"test_name"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "groupId":"12"
  }
}
```

- 名称：edgeEventAdapter/face/removeGroup
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库ID（最大字符长度63）

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12"
}
```

返回

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

- 名称：edgeEventAdapter/face/modifyGroup
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC（NVR下IPC）
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
groupId	必填	String	人脸库ID（最大字符长度63）
groupName	必填	String	人脸库名称（最大字符长度63）
extInfo	可选	String	扩展信息

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12",
  "groupName":"test_name2"
}
```

返回

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

- 名称：edgeEventAdapter/face/getGroupDetail
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库Id

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSON	数据
+groupId	String	人脸库ID
+groupName	String	人脸库名称
+groupFaceNum	Integer	人脸库入库数量
+extInfo	String	扩展信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "groupId":"12",
    "groupName":"test_name",
    "groupFaceNum":100
  }
}
```

- 名称：edgeEventAdapter/face/getAllGroups
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSON	数据
+groups	JsonArry	
++groupId	String	人脸库ID（最大字符长度63）
++groupName	String	人脸库名称（最大字符长度63）
++groupFaceNum	Integer	人脸库入库数量
++extInfo	String	扩展信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":[{
    "groupId":"12",
    "groupName":"test_name",
    "groupFaceNum":10
  },{
    "groupId":"13",
    "groupName":"test_name2",
    "groupFaceNum":10
  }]
}
```

- 名称：edgeEventAdapter/face/create
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库ID（最大字符长度63）
facePicUrl	必填	String	人脸图片url
name	可选	String	姓名（最大字符长度63）
extInfo	可选	String	扩展信息

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSON	数据
+faceId	String	人脸ID（最大字符长度63）

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12",
  "name":"test_name",
  "facePicUrl":"****"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "faceId":"12"
  }
}
```

- 名称：edgeEventAdapter/face/remove
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库ID（最大字符长度63）
faceId	必填	String	人脸Id（最大字符长度31）

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12",
  "faceId":"12"
}
```

返回

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

- 名称：edgeEventAdapter/face/modify
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库ID（最大字符长度63）
faceId	必填	String	人脸Id（最大字符长度31），索引值。不支持修改
facePicUrl	可选	String	人脸图片url
name	可选	String	姓名（最大字符长度63）
extInfo	可选	String	扩展信息

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12",
  "faceId",12,
  "name":"test_name2",
  "facePicUrl":"****"
}
```

返回

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

- 名称：edgeEventAdapter/face/search
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息

ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库ID
name	可选	String	姓名（最大字符长度63）
extInfo	可选	String	扩展信息
pageNo	可选	Integer	分页查询，页码，从1开始 默认为1
pageSize	可选	Integer	分页查询，页尺寸 默认为10

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSON	数据
+faces	JsonArray	
++groupId	String	人脸库ID（最大字符长度63）
++groupName	String	人脸库名称（最大字符长度127）
++faceId	String	人脸Id（最大字符长度31）
++name	String	姓名（最大字符长度63）
++facePicBase64	String	人脸图片base64编码后的数据
++facePicType	String	图片类型 JPG PNG BMP
++extInfo	String	扩展信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12",
  "name":"test_name2",
  "pageNo":1,
  "getCount":10
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "faces":[{
      "groupId":"12",
      "groupName":"test_grp_name",
      "faceId":"12",
      "name":"test_name",
      "facePicBase64":"AABB****1122",
      "facePicType":"JPG"
    },{
      "groupId":"12",
      "groupName":"test_grp_name",
      "faceId":"13",
      "name":"test_name2",
      "facePicBase64":"AABB****1122",
      "facePicType":"JPG"
    }]
  }
}
```

3.4 事件布防

- 名称：edgeEventAdapter/alarmDeploy/setRule
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
deviceType	必填	Integer	设备类型 2：直连IPC 3：非直连IPC（NVR下IPC）

eventType	必填	Integer	事件类型 1：人脸识别 2：区域入侵 3：电动车进电梯 4：垃圾桶满溢 5：高空抛物 6：消防通道占用 7：热成像-温度异常 8：热成像-人体测温 100：其他
alarmRule	必填	JsonArray	布防计划，size
+weekDay	必填	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
+timeSlot	必填	JsonArray	时间段
++time	必填	String	时间范围 0:00-2:00
++interval	必填	Integer	时间间隔，单位秒
++timeExtInfo	可选	String	时间段扩展信息
+bindGroup	必填	JSONArray	人脸底库绑定 eventType == 1 时有效
++groupId	必填	String	eventType == 1 时有效
+extInfo	可选	String	规则扩展信息

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":2,
  "eventType":1,
  "alarmRule":[{
    "weekDay":"SUNDAY",
    "timeSlot":[{
      "time":"00:00-2:00",
      "interval":600
    },{
      "time":"02:00-4:00",
      "interval":600
    }]
  },{
    "weekDay":"MONDAY",
    "timeSlot":[{
      "time":"00:00-23:59",
      "interval":600
    }]
  }]
}
```

返回

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

- 名称：edgeEventAdapter/alarmDeploy/getRule
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
deviceType	必填	Integer	设备类型 2：直连IPC 3：非直连IPC（NVR下IPC）
eventType	必填	Integer	事件类型 1：人脸识别 2：区域入侵 3：电动车进电梯 4：垃圾桶满溢 5：高空抛物 6：消防通道占用 7：热成像-温度异常 8：热成像-人体测温 100：其他

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSONArray	
+weekDay	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
+timeSlot	JsonArray	时间段
++time	String	时间范围 0:00-2:00
++interval	Integer	时间间隔，单位秒
++timeExtInfo	String	时间段扩展信息
+bindGroup	JSONArray	人脸底库绑定 deviceType == 1 时有效
++groupId	String	deviceType == 1 时有效
+extInfo	String	规则扩展信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":2,
  "eventType":1
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":[{
    "weekDay":"SUNDAY",
    "timeSlot":[{
      "time":"00:00-2:00",
      "interval":600
    },{
      "time":"02:00-4:00",
      "interval":600
    }]
  },{
    "weekDay":"MONDAY",
    "timeSlot":[{
      "time":"00:00-23:59",
      "interval":600
    }]
  }]
}
```

- 名称：edgeSecurityEvent/reportAlarmEvent
- 方向：事件适配应用->智能安防应用
- 说明：
  - 事件上报接口包括四部分：事件信息、目标信息、事件扩展信息、目标扩展信息。其中事件信息和目标信息为固定部分，所有事件都可以按照格式填充这两部分。根据不同的事件，通过事件扩展信息可扩展事件信息中没有的属性，通过目标扩展信息可扩展目标信息中没有的属性。
  - 一次事件上报只能有1个目标，如果某个事件有多个目标，则上报多个事件，多个事件的eventId和eventTime相同。
  - 关于可选字段，要根据现场需求判断是否需要
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC

eventType	必填	Integer	事件类型 1：人脸识别 2：区域入侵 3：电动车进电梯 4：垃圾桶满溢 5：高空抛物 6：消防通道占用 7：热成像-温度异常 8：热成像-人体测温 9：测温规则报警 10：电动车进出单元门 11：违停 12：人员徘徊 13：人员摔倒监测 100：其他
eventTime	必填	Long	事件的触发时间，UTC时间，精确到s
eventId	必填	String	事件ID，事件的唯一标识
eventStatus	可选	Integer	事件状态 0：瞬时 1：开始 2：停止
confidence	可选	Integer	事件置信度，0~255，值越大置信度越高
scenePicBase64	可选	String	全景图片base64，如果是智能事件，此字段为必填
scenePicType	可选	String	全景图片类型，如果是智能事件，此字段为必填 JPG PNG BMP
objectInfo	可选	Json	目标INFO
+objPicBase64	可选	String	目标图片base64，如果是智能事件，此字段为必填
+objPicType	可选	String	目标图片类型，如果是智能事件，此字段为必填 JPG PNG BMP
+letfTopX	可选	Integer	左上角x坐标(0, 8192)
+leftTopY	可选	Integer	左上角y坐标(0, 8192)
+rightBottomX	可选	Integer	右下角x坐标(0, 8192)
+rightBottomY	可选	Integer	右下角y坐标(0, 8192)

+objectExtInfo	可选	Json	目标自定义扩展信息
extInfo	可选	Json	其他扩展信息

objectExtInfo字段：

objectExtInfo	可选	Json	eventType: 1 人脸识别
+similarity	必填	Float	相似度 (0, 1)
+groupId	必填	String	人脸所在底库的GroupId
+faceId	必填	String	底库人脸Id
+gender	可选	Integer	性别 1：男 2：女 100：未知
+glass	可选	Integer	是否戴眼镜 0：未戴眼镜 1：戴眼镜 100：未知
+mask	可选	Integer	是否戴口罩 0：未戴口罩 1：戴口罩 100：未知
+pitch	可选	Integer	俯仰角 [-90,90]
+yow	可选	Integer	偏航角 [-90,90]
+roll	可选	Integer	翻滚角 [-90,90]
+faceQuality	可选	Integer	人脸质量 [0-100]
+facePicBase64	可选	String	底库人脸图片base64
+facePicType	可选	String	底库人脸图片类型 JPG PNG BMP

objectExtInfo	可选	Json	eventType: 2 区域入侵
+targetType	必填	Integer	目标类型 1：人 2：车 100：其他

objectExtInfo	可选	Json	eventType: 6 消防通道占用

+plateColor	必填	Integer	车牌颜色 0：未知颜色 1：蓝色 2：黄色 3：白色 4：黑色 5：绿色 6：黄绿色 100：其他颜色
+plateNum	可选	String	车牌号
+vehicleColor	必填	Integer	车身颜色 1：红色 2：绿色 3：蓝色4：黄色 5：白色 6：灰色 7：黑色 8：紫色 9：棕色 10：粉色 11：银色 12：深蓝 13：深灰 14：青色 100：其他颜色

objectExtInfo	可选	Json	eventType: 7 热成像-温度异常
+alarmContion	必填	Integer	报警条件 1：大于 2：等于 3：小于
+tempValue	必填	Float	温度值
+tempUnit	必填	Integer	温度单位 1：摄氏度 2：华氏度

objectExtInfo	可选	Json	eventType: 8 热成像-人体测温
+tempMax	必填	Float	最高温度
+tempUnit	必填	Integer	温度单位 1：摄氏度 2：华氏度

objectExtInfo	可选	Json	eventType: 10 电动车进出单元门

+type	必填	Float	1：进单元门 2：出单元门 100：未知

objectExtInfo	可选	Json	eventType: 11 违停
+type	必填	Integer	1：机动车 2：非机动车 100：未知
+plateColor	可选	Integer	车牌颜色 0：未知颜色 1：蓝色 2：黄色 3：白色 4：黑色 5：绿色 6：黄绿色 100：其他颜色
+plateNum	可选	String	车牌号
+vehicleColor	可选	Integer	车身颜色 1：红色 2：绿色 3：蓝色4：黄色 5：白色 6：灰色 7：黑色 8：紫色 9：棕色 10：粉色 11：银色 12：深蓝 13：深灰 14：青色 100：其他颜色

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求（人脸识别事件）

```
{
  "token":"cc85****0001",
  "ip":"192.168.1.2",
  "channel":1,
  "deviceType":3,
  "eventType":1,
  "eventTime":1631798285,
  "eventId":123,
  "eventStatus":0,
  "confidence":240,
  "scenePicBase64":"AABB****1122",
  "scenePicType":"JPG",
  "objectInfo":{
    "objPicBase64":"AABB****1122",
    "objPicType":"JPG",
    "leftTopY":100,
    "leftTopY":100,
    "rightBottomX":400,
    "rightBottomY":400,
    "objectExtInfo":{
     "similarity":0.65,
      "groupId":"12",
      "faceId":"1",
      "gender":1,
      "glass":1,
      "mask":0,
      "pitch":15,
      "yow":15,
      "roll":15,
      "faceQuality":95,
      "facePicBase64":"AABB****1122",
      "facePicType":"JPG",
    }
  }
}
```

返回

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

3.5 抓图计划

- 名称：edgeEventAdapter/capture/setTask
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
taskId	必填	String	任务的唯一标识
deviceList	必填	JSONArray	关联的监控设备
+ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
+channel	必填	Integer	通道号，从0开始，deviceType为3时有效
+deviceType	必填	Integer	设备类型 2：直连IPC 3：非直连IPC（NVR下IPC）

startDate	必填	String	计划开始时间 格式：YYYY-MM-DD
endDate	必填	String	计划结束时间 格式：YYYY-MM-DD
captureWeekDay	可选	JsonArray	抓图周期 captureWeekDay和cron二选一必填
+weekDay	必填	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
+captureTime	必填	List	抓图时分 24时制，格式：hh:mm
cron	可选	List	cron表达式 captureWeekDay和cron二选一必填
status	必填	Integer	启用状态 1：启用 0：禁用
extInfo	可选	String	扩展字段

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "taskId":"123",
  "deviceList":[{
    "ip":"192.168.1.2",
    "channel":1,
    "deviceType":3
  },{
    "ip":"192.168.1.2",
    "channel":2,
    "deviceType":3
  }],
  "startDate":"2010-01-01",
  "endDate":"2010-04-01",
  "captureWeekDay":[{
    "weekDay":"SUNDAY",
    "captureTime":"10:00,12:00"
  },{
    "weekDay":"MONDAY",
    "captureTime":"10:00,12:00"
  }],
  "status":1
}
```

返回

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

- 名称：edgeEventAdapter/capture/getTask
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
taskId	必填	String	任务的唯一标识

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息
data	必填	JSON	数据
+deviceList	必填	JSONArray	关联的监控设备
++ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
++channel	必填	Integer	通道号，从0开始，deviceType为3时有效

++deviceType	必填	Integer	设备类型 2：直连IPC 3：非直连IPC（NVR下IPC）
+startDate	必填	String	计划开始时间 格式：YYYY-MM-DD
+endDate	必填	String	计划结束时间 格式：YYYY-MM-DD
+captureWeekDay	可选	JsonArray	抓图周期 captureWeekDay和cron二选一必填
++weekDay	必填	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
++captureTime	必填	List	抓图时分 24时制，格式：hh:mm
+cron	可选	List	cron表达式 captureWeekDay和cron二选一必填
+status	必填	Integer	启用状态 1：启用 0：禁用
+extInfo	可选	String	扩展字段

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "taskId":"123"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "deviceList":[{
      "ip":"192.168.1.2",
      "channel":1,
      "deviceType":3
    },{
      "ip":"192.168.1.2",
      "channel":2,
      "deviceType":3
    }],
    "startDate":"2010-01-01",
    "endDate":"2010-04-01",
    "captureWeekDay":[{
      "weekDay":"SUNDAY",
      "captureTime":"10:00,12:00"
    },{
      "weekDay":"MONDAY",
      "captureTime":"10:00,12:00"
    }],
    "status":1
  }
}
```

- 名称：edgeEventAdapter/capture/deleteTask
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
taskId	必填	String	任务的唯一标识

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "taskId":"123"
}
```

返回

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

- 名称：edgeSecurityEvent/capture/report
- 方向：事件适配应用->智能安防应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
deviceType	必填	Integer	设备类型 2：直连IPC 3：非直连IPC（NVR下IPC）
taskId	可选	String	任务ID
captureTime	必填	Long	抓图时间，UTC时间，精确到s
picBase64	必填	String	抓拍图片base64
picType	必填	String	图片类型 JPG PNG BMP

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.3",
  "channel":1,
  "deviceType":3,
  "captureTime":1631798285,
  "picBase64":"AABB****1122",
  "picType":"JPG"
}
```

返回

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

3.6 抓拍记录查询

- 名称：edgeEventAdapter/record/get
- 方向：智能安防应用->事件适配应用
- 版本：1.0.1
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
picBase64	必填	String	图片base64
picType	必填	String	图片类型 JPG PNG BMP
recordLimit	必填	Integer	获取记录数量限制
startTime	必填	Long	UTC时间，精确到s
endTime	必填	Long	UTC时间，精确到s endTime与startTime相差<=24h
minSimilarity	必填	Float	最小相似度(0,1)
maxSimilarity	必填	Float	最大相似度(0,1) maxSimilarity > minSimilarity
pageNo	可选	Integer	分页查询，页码，从1开始 默认为1
pageSize	可选	Integer	分页查询，页尺寸，pageSize>1&pageSize<=20 默认为10

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息
data	必填	Json	数据
+totalSize	必填	Integer	查询结果总数，注意，不是当页结果，是所有页的结果总和
+recordList	必填	JsonArray	
++ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
++channel	必填	Integer	通道号，从0开始，deviceType为3时有效
++deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC（NVR下IPC）

++smallPicBase64	必填	String	小图base64
++smallPicType	必填	String	图片类型 JPG PNG BMP
++scenePicId	必填	String	场景图id
++similarity	必填	Float	相似度(0,1)
++captureTime	必填	Long	UTC时间，精确到s
++extinfo	可选	JsonArray	眼镜，性别，年龄等

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "picBase64":"AABB****1122",
  "picType":"JPG",
  "recordLimit":10,
  "startTime":1631794685,
  "endTime":1631798285,
  "minSimilarity":0.55,
  "maxSimilarity":0.65
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "totalSize":100,
    "recordList":[{
      "ip":"192.168.1.3",
      "channel":1,
      "deviceType":3,
      "smallPicBase64":"AABB****1122",
      "smallPicType":"JPG",
      "scenePicId":"pic1234",
      "similarity":0.6,
      "captureTime":1631795685
    },{
      "ip":"192.168.1.3",
      "channel":1,
      "deviceType":3,
      "smallPicBase64":"AABB****1122",
      "smallPicType":"JPG",
      "scenePicId":"pic1235",
      "similarity":0.6,
      "captureTime":1631795685
    }]
  }
}
```

- 名称：edgeEventAdapter/record/scenePicUpload
- 方向：智能安防应用->事件适配应用
- 版本：1.0.1
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明

token	必填	String	令牌信息
scenePicId	必填	String	场景图ID

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息
data	必填	Json	数据
+picBase64	必填	String	场景图base64
+picType	必填	String	图片类型 JPG PNG BMP

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "scenePicId":"pic1234",
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "picBase64":"AABB****1122",
    "picType":"JPG"
  }
}
```

- 名称：edgeEventAdapter/record/getAll
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
picBase64	必填	String	图片base64
picType	必填	String	图片类型 JPG PNG BMP
startTime	必填	Long	UTC时间，精确到s

endTime	必填	Long	UTC时间，精确到s endTime与startTime相差<=24h
minSimilarity	必填	Float	最小相似度(0,1)
maxSimilarity	必填	Float	最大相似度(0,1) maxSimilarity > minSimilarity

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息
data	必填	Json	数据
+recordList	必填	JsonArray	
++ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
++channel	必填	Integer	通道号，从0开始，deviceType为3时有效
++deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC（NVR下IPC）
++similarity	必填	Float	相似度(0,1)
++captureTime	必填	Long	UTC时间，精确到s
++extinfo	可选	JsonArray	眼镜，性别，年龄等

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "picBase64":"AABB****1122",
  "picType":"JPG",
  "recordLimit":10,
  "startTime":1631794685,
  "endTime":1631798285,
  "minSimilarity":0.55,
  "maxSimilarity":0.65
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "recordList":[{
      "ip":"192.168.1.3",
      "channel":1,
      "deviceType":3,
      "similarity":0.6,
      "captureTime":1631795685
    },{
      "ip":"192.168.1.3",
      "channel":1,
      "deviceType":3,
      "similarity":0.6,
      "captureTime":1631795685
    }]
  }
}
```

4. 附录

4.1 错误码

code	message	说明
200	success	成功
801	adapter authentication failed	适配应用认证失败
802	adapter not authentication	适配应用未认证
803	device certificate not exist	设备证书不存在
804	param check error: /'%s/'	参数校验失败，返回失败的参数
805	memory exception	内存异常
806	json data wrong format	json数据格式错误
807	device not exist	设备不存在
808	procotol not support	不支持的协议
809	manufacture error	设备厂商错误
810	abnormal communication	模块通讯异常

4.2 应用要求

硬件要求：

- cpu（2.0 Ghz）： < 0.2core
- 内存： < 1Gb
- 磁盘： < 1Gb

服务要求：

- 本地可持久化数据：

智能安防应用的IP和端口号信息。

事件布防信息。

日志数据。

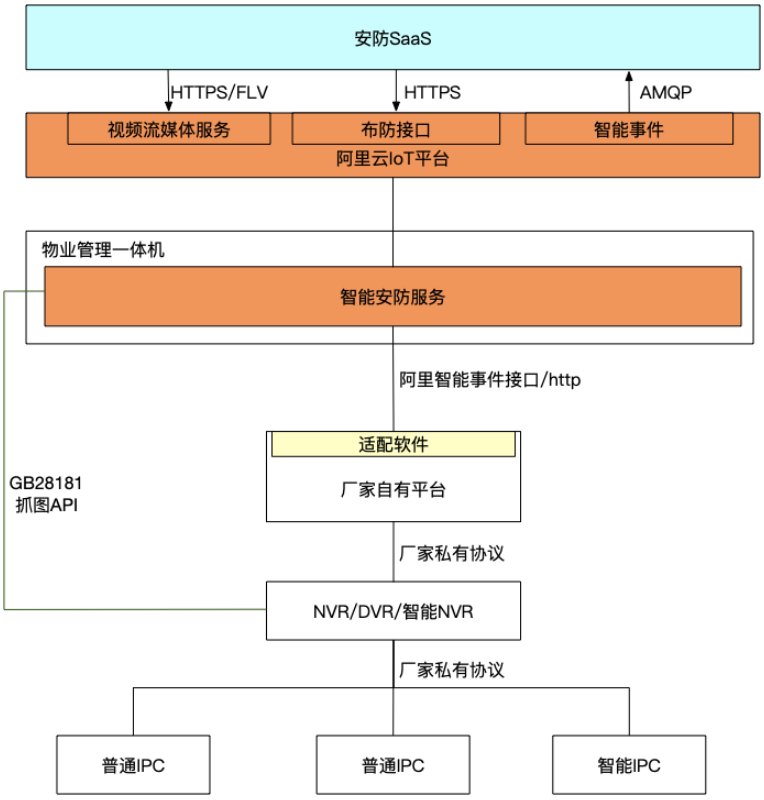
- 日志：日志中需带有级别、时间信息，且需提供以下项目：接口日志（接入、配置、告警）、设备状态日志、运行异常日志。日志可过滤。

4.3 修改记录

版本号	修改内容	修改时间	责任人
Rev0.1	文档创建	2021-09-01	诚治
Rev0.2	1. 新增、修改人脸库接口，增加通道字段 2. 修改人脸库接口，去掉groupType字段 3. 3.6.2章节增加uploadUrl说明	2022-01-18	诚治
Rev0.3	1. 修改“3.6.1 抓拍记录查询”接口，返回信息增加查询总数字段 2. 修改“3.6.2 抓拍记录场景图获取”接口，下发的uploadUrl改为返回downloadUrl	2022-02-16	诚治
Rev0.4	1. 修改“3.4.3 告警事件上报”接口，类型增加“其他”	2022-02-22	诚治
Rev0.5	1. 修改“3.2.1 获取所有设备配置”接口，子IPC增加用户名密码	2022-03-23	诚治

3.5.3.3.3. 智能安防服务-边缘事件管理API-Rev0.5

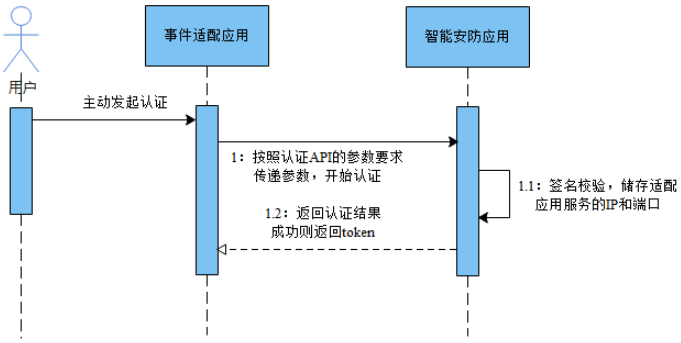
1. 架构图



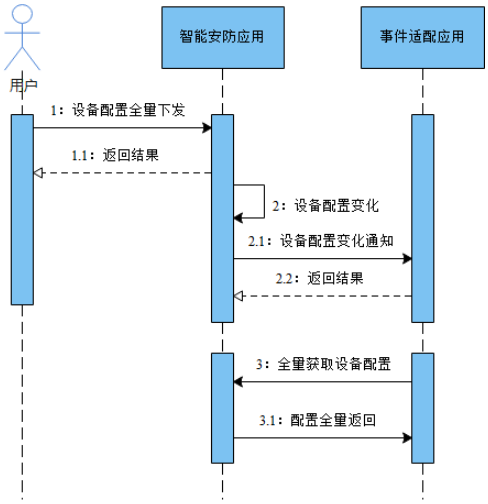
2. 流程图

2.1 应用认证

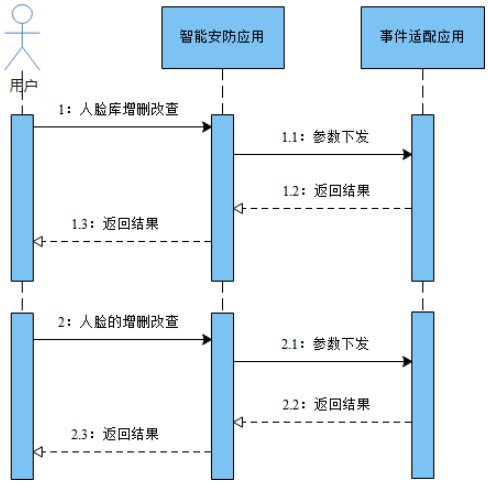
说明：事件适配应用向智能安防应用主动发起认证，认证成功后，智能安防应用返回令牌信息（token），智能安防应用接受带有此token的HTTP请求，同时，事件适配应用也接受来自智能安防应用带有此token的HTTP请求。



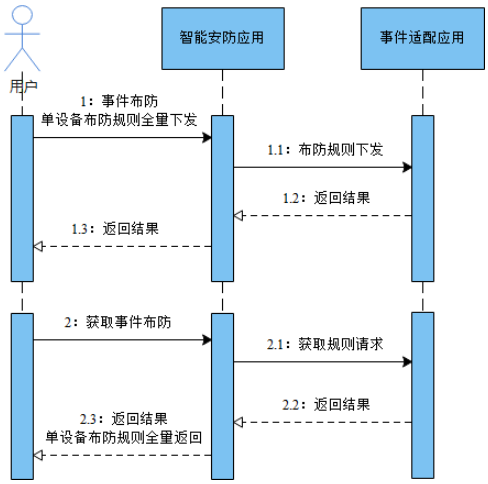
2.2 设备配置



2.3 人脸管理

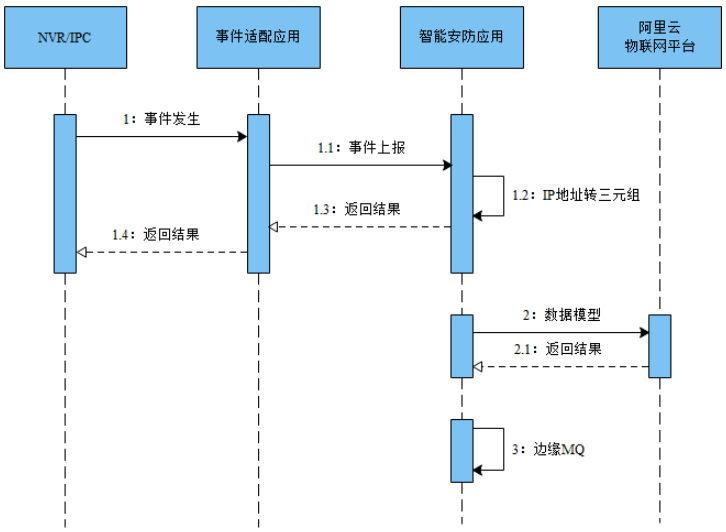


2.4 事件布防

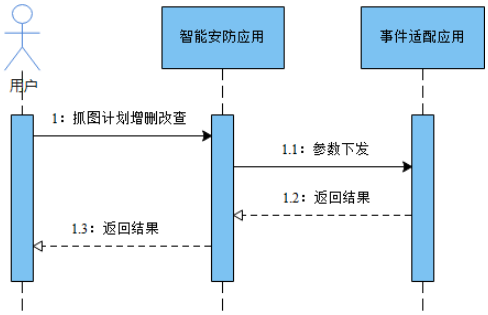


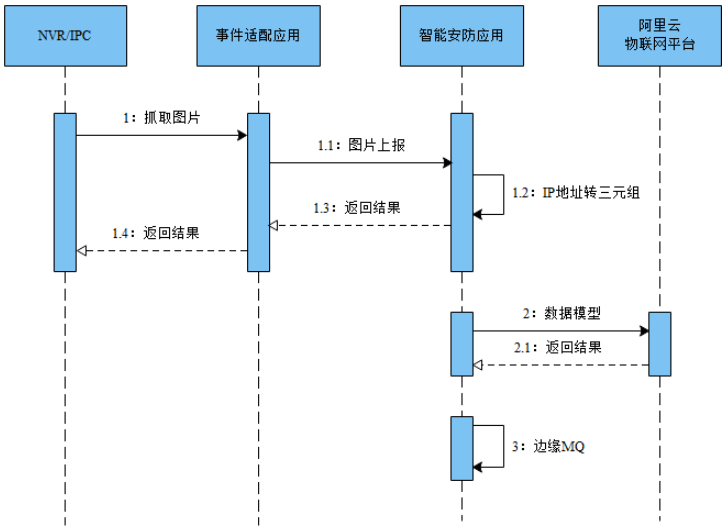
说明：

NVR/IPC产生告警事件后，安防应用负责管理告警事件。

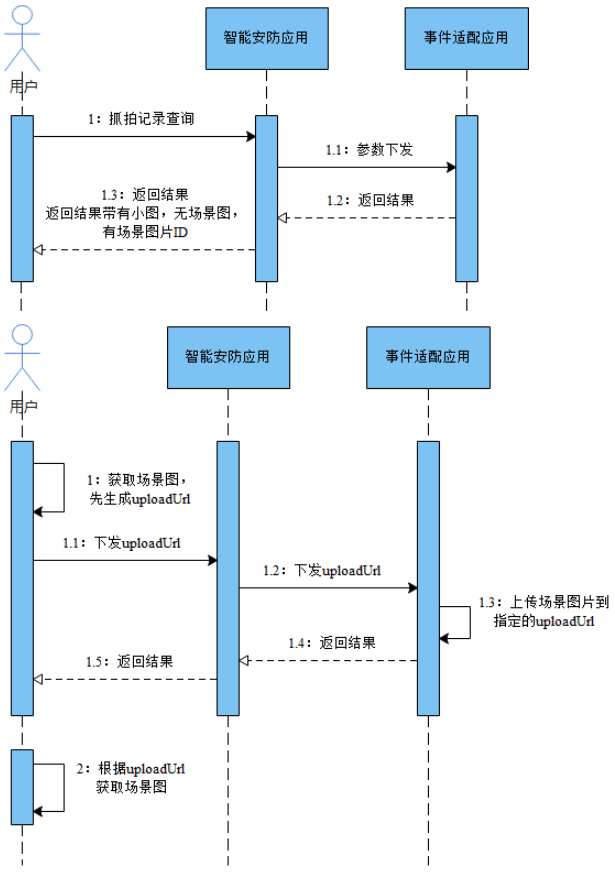


2.5 抓图计划





2.6 抓拍记录查询



3. API

智能安防应用端口号：5080

事件适配应用IP地址和端口号：事件适配应用使用前，需向阿里云IOT后台小二提交端口审核，审核通过后端口方可使用。IP地址和端口号通过认证API的参数传递给智能安防应用。

3.1 接入认证

- 名称：edgeSecurityEvent/appAuthentication
- 方向：事件适配应用->智能安防应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
ip	必填	String	连接发起方的IPv4地址
mac	必填	String	连接发起方的MAC地址 00-**-3D-40
time	必填	Long	请求的时间戳。以秒为单位，需要使用UTC时间。
manufacture	必填	String	品牌 DH：大华 HIK：海康 UNIVIEW：宇视 TIANDY：天地伟业
signMethod	必填	String	签名方式，目前支持MD5 取值范围 MD5 SHA1 SHA256
sign	必填	String	需要签名的内容封装格式： "TIME{time}MAC{mac}IP{ip}MANUFACTURE{DH}"大括号内的是具体内容，其外面的是名称。备注：time要转换成string类型。
alarmAdapterPort	必填	Integer	事件适配应用的端口号 智能安防应用作为发起方，此参数省略

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	Json	数据
+token	String	令牌信息

- 示例代码

请求

```
{
  "ip":"192.168.1.2",
  "mac":"00-**-3D-40",
  "time":1589472000,
  "manufacture":"DH",
  "signMethod":"md5",
  "sign":"4db511**42",
  "alarmAdapterPort":10080
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "token":"cc85****0001"
  }
}
```

3.2 设备管理

- 名称：edgeSecurityEvent/deviceManage/getConfig
- 方向：事件适配应用->智能安防应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
manufacture	必填	String	品牌 DH：大华 HIK：海康 UNIVIEW：宇视 TIANDY：天地伟业

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSONArray	数据
+manufacture	String	品牌 DH：大华 HIK：海康 UNIVIEW：宇视 TIANDY：天地伟业
+deviceType	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC
+ip	String	设备的IP地址，deviceType为3时，为NVR的IP
+port	Integer	设备的端口号，deviceType为3时，为NVR的PORT
+username	String	登录用户名，deviceType为3时，为NVR的username
+password	String	登录密码，deviceType为3时，为NVR的password
+channel	Integer	通道号，从0开始，deviceType为3时有效

+childIp	String	deviceType为3时有效，子设备IP
+childPort	Integer	deviceType为3时有效，子设备PORT
+childUsername	String	deviceType为3时有效，子设备username
+childPassword	String	deviceType为3时有效，子设备password
+extInfo	String	扩展字段

- 示例代码

请求

```
{
  "token":"cc85****0001",
  "manufacture":"DH"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":[{
    "manufacture":"DH",
    "deviceType":1,
    "ip":"192.168.1.2",
    "port":20080,
    "u********e":"*********",
    "p********d":"*********"
  },{
    "manufacture":"DH",
    "deviceType":2,
    "ip":"192.168.1.3",
    "port":20080,
    "u********e":"*********",
    "p********d":"*********"
  },{
    "manufacture":"DH",
    "deviceType":3,
    "ip":"192.168.1.3",
    "port":20080,
    "u********e":"*********",
    "p********d":"*********",
    "childIp":"192.168.1.3",
    "childPort":20080,
    "childU********e":"*********",
    "childP********d":"*********",
    "channel":0
  }]
}
```

- 名称：edgeEventAdapter/deviceManage/configChange
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息

- 响应参数

参数	类型	说明

code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
}
```

返回

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

- 名称：edgeEventAdapter/deviceManage/getStatus
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JsonArray	数据
+ip	String	设备地址，deviceType为3时，ip为NVR的IP
+channel	Integer	通道号，从0开始，deviceType为3时有效
+deviceType	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC
+status	Integer	连接状态 0：连接异常 1：连接正常
+unnormalInfo	String	异常信息，自定义异常信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":[{
    "ip":"192.168.1.2",
    "deviceType":2,
    "status":1
  },{
    "ip":"192.168.1.3",
    "deviceType":2,
    "status":1
  },{
    "ip":"192.168.1.4",
    "channel":1,
    "deviceType":3,
    "status":0,
    "unnormalInfo":"password error"
  }]
}
```

3.3 人脸管理

- 名称：edgeEventAdapter/face/createGroup
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC（NVR下IPC）
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
groupName	必填	String	人脸库名称（最大字符长度63）
groupType	必填	Integer	人脸库类型 1：黑名单 2：白名单 3：员工 4：访客 100：其他
extInfo	可选	String	扩展信息

- 响应参数

参数	类型	说明

code	Integer	错误码
message	String	信息
data	JSON	数据
+groupId	String	人脸库ID

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupName":"test_name"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "groupId":"12"
  }
}
```

- 名称：edgeEventAdapter/face/removeGroup
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库ID（最大字符长度63）

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12"
}
```

返回

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

- 名称：edgeEventAdapter/face/modifyGroup
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC（NVR下IPC）
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
groupId	必填	String	人脸库ID（最大字符长度63）
groupName	必填	String	人脸库名称（最大字符长度63）
extInfo	可选	String	扩展信息

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12",
  "groupName":"test_name2"
}
```

返回

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

- 名称：edgeEventAdapter/face/getGroupDetail
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库Id

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSON	数据
+groupId	String	人脸库ID
+groupName	String	人脸库名称
+groupFaceNum	Integer	人脸库入库数量
+extInfo	String	扩展信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "groupId":"12",
    "groupName":"test_name",
    "groupFaceNum":100
  }
}
```

- 名称：edgeEventAdapter/face/getAllGroups
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSON	数据
+groups	JsonArry	
++groupId	String	人脸库ID（最大字符长度63）
++groupName	String	人脸库名称（最大字符长度63）
++groupFaceNum	Integer	人脸库入库数量
++extInfo	String	扩展信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":[{
    "groupId":"12",
    "groupName":"test_name",
    "groupFaceNum":10
  },{
    "groupId":"13",
    "groupName":"test_name2",
    "groupFaceNum":10
  }]
}
```

- 名称：edgeEventAdapter/face/create
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库ID（最大字符长度63）
facePicUrl	必填	String	人脸图片url
name	可选	String	姓名（最大字符长度63）
extInfo	可选	String	扩展信息

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSON	数据
+faceId	String	人脸ID（最大字符长度63）

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12",
  "name":"test_name",
  "facePicUrl":"****"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "faceId":"12"
  }
}
```

- 名称：edgeEventAdapter/face/remove
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库ID（最大字符长度63）
faceId	必填	String	人脸Id（最大字符长度31）

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12",
  "faceId":"12"
}
```

返回

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

- 名称：edgeEventAdapter/face/modify
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库ID（最大字符长度63）
faceId	必填	String	人脸Id（最大字符长度31），索引值。不支持修改
facePicUrl	可选	String	人脸图片url
name	可选	String	姓名（最大字符长度63）
extInfo	可选	String	扩展信息

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12",
  "faceId",12,
  "name":"test_name2",
  "facePicUrl":"****"
}
```

返回

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

- 名称：edgeEventAdapter/face/search
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息

ip	必填	String	设备地址
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC
groupId	必填	String	人脸库ID
name	可选	String	姓名（最大字符长度63）
extInfo	可选	String	扩展信息
pageNo	可选	Integer	分页查询，页码，从1开始 默认为1
pageSize	可选	Integer	分页查询，页尺寸 默认为10

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSON	数据
+faces	JsonArray	
++groupId	String	人脸库ID（最大字符长度63）
++groupName	String	人脸库名称（最大字符长度127）
++faceId	String	人脸Id（最大字符长度31）
++name	String	姓名（最大字符长度63）
++facePicBase64	String	人脸图片base64编码后的数据
++facePicType	String	图片类型 JPG PNG BMP
++extInfo	String	扩展信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":1,
  "groupId":"12",
  "name":"test_name2",
  "pageNo":1,
  "getCount":10
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "faces":[{
      "groupId":"12",
      "groupName":"test_grp_name",
      "faceId":"12",
      "name":"test_name",
      "facePicBase64":"AABB****1122",
      "facePicType":"JPG"
    },{
      "groupId":"12",
      "groupName":"test_grp_name",
      "faceId":"13",
      "name":"test_name2",
      "facePicBase64":"AABB****1122",
      "facePicType":"JPG"
    }]
  }
}
```

3.4 事件布防

- 名称：edgeEventAdapter/alarmDeploy/setRule
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
deviceType	必填	Integer	设备类型 2：直连IPC 3：非直连IPC（NVR下IPC）

eventType	必填	Integer	事件类型 1：人脸识别 2：区域入侵 3：电动车进电梯 4：垃圾桶满溢 5：高空抛物 6：消防通道占用 7：热成像-温度异常 8：热成像-人体测温 100：其他
alarmRule	必填	JsonArray	布防计划，size
+weekDay	必填	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
+timeSlot	必填	JsonArray	时间段
++time	必填	String	时间范围 0:00-2:00
++interval	必填	Integer	时间间隔，单位秒
++timeExtInfo	可选	String	时间段扩展信息
+bindGroup	必填	JSONArray	人脸底库绑定 eventType == 1 时有效
++groupId	必填	String	eventType == 1 时有效
+extInfo	可选	String	规则扩展信息

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":2,
  "eventType":1,
  "alarmRule":[{
    "weekDay":"SUNDAY",
    "timeSlot":[{
      "time":"00:00-2:00",
      "interval":600
    },{
      "time":"02:00-4:00",
      "interval":600
    }]
  },{
    "weekDay":"MONDAY",
    "timeSlot":[{
      "time":"00:00-23:59",
      "interval":600
    }]
  }]
}
```

返回

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

- 名称：edgeEventAdapter/alarmDeploy/getRule
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
deviceType	必填	Integer	设备类型 2：直连IPC 3：非直连IPC（NVR下IPC）
eventType	必填	Integer	事件类型 1：人脸识别 2：区域入侵 3：电动车进电梯 4：垃圾桶满溢 5：高空抛物 6：消防通道占用 7：热成像-温度异常 8：热成像-人体测温 100：其他

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息
data	JSONArray	
+weekDay	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
+timeSlot	JsonArray	时间段
++time	String	时间范围 0:00-2:00
++interval	Integer	时间间隔，单位秒
++timeExtInfo	String	时间段扩展信息
+bindGroup	JSONArray	人脸底库绑定 deviceType == 1 时有效
++groupId	String	deviceType == 1 时有效
+extInfo	String	规则扩展信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.2",
  "deviceType":2,
  "eventType":1
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":[{
    "weekDay":"SUNDAY",
    "timeSlot":[{
      "time":"00:00-2:00",
      "interval":600
    },{
      "time":"02:00-4:00",
      "interval":600
    }]
  },{
    "weekDay":"MONDAY",
    "timeSlot":[{
      "time":"00:00-23:59",
      "interval":600
    }]
  }]
}
```

- 名称：edgeSecurityEvent/reportAlarmEvent
- 方向：事件适配应用->智能安防应用
- 说明：
  - 事件上报接口包括四部分：事件信息、目标信息、事件扩展信息、目标扩展信息。其中事件信息和目标信息为固定部分，所有事件都可以按照格式填充这两部分。根据不同的事件，通过事件扩展信息可扩展事件信息中没有的属性，通过目标扩展信息可扩展目标信息中没有的属性。
  - 一次事件上报只能有1个目标，如果某个事件有多个目标，则上报多个事件，多个事件的eventId和eventTime相同。
  - 关于可选字段，要根据现场需求判断是否需要
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC

eventType	必填	Integer	事件类型 1：人脸识别 2：区域入侵 3：电动车进电梯 4：垃圾桶满溢 5：高空抛物 6：消防通道占用 7：热成像-温度异常 8：热成像-人体测温 9：测温规则报警 10：电动车进出单元门 11：违停 12：人员徘徊 13：人员摔倒监测 100：其他
eventTime	必填	Long	事件的触发时间，UTC时间，精确到s
eventId	必填	String	事件ID，事件的唯一标识
eventStatus	可选	Integer	事件状态 0：瞬时 1：开始 2：停止
confidence	可选	Integer	事件置信度，0~255，值越大置信度越高
scenePicBase64	可选	String	全景图片base64，如果是智能事件，此字段为必填
scenePicType	可选	String	全景图片类型，如果是智能事件，此字段为必填 JPG PNG BMP
objectInfo	可选	Json	目标INFO
+objPicBase64	可选	String	目标图片base64，如果是智能事件，此字段为必填
+objPicType	可选	String	目标图片类型，如果是智能事件，此字段为必填 JPG PNG BMP
+letfTopX	可选	Integer	左上角x坐标(0, 8192)
+leftTopY	可选	Integer	左上角y坐标(0, 8192)
+rightBottomX	可选	Integer	右下角x坐标(0, 8192)
+rightBottomY	可选	Integer	右下角y坐标(0, 8192)

+objectExtInfo	可选	Json	目标自定义扩展信息
extInfo	可选	Json	其他扩展信息

objectExtInfo字段：

objectExtInfo	可选	Json	eventType: 1 人脸识别
+similarity	必填	Float	相似度 (0, 1)
+groupId	必填	String	人脸所在底库的GroupId
+faceId	必填	String	底库人脸Id
+gender	可选	Integer	性别 1：男 2：女 100：未知
+glass	可选	Integer	是否戴眼镜 0：未戴眼镜 1：戴眼镜 100：未知
+mask	可选	Integer	是否戴口罩 0：未戴口罩 1：戴口罩 100：未知
+pitch	可选	Integer	俯仰角 [-90,90]
+yow	可选	Integer	偏航角 [-90,90]
+roll	可选	Integer	翻滚角 [-90,90]
+faceQuality	可选	Integer	人脸质量 [0-100]
+facePicBase64	可选	String	底库人脸图片base64
+facePicType	可选	String	底库人脸图片类型 JPG PNG BMP

objectExtInfo	可选	Json	eventType: 2 区域入侵
+targetType	必填	Integer	目标类型 1：人 2：车 100：其他

objectExtInfo	可选	Json	eventType: 6 消防通道占用

+plateColor	必填	Integer	车牌颜色 0：未知颜色 1：蓝色 2：黄色 3：白色 4：黑色 5：绿色 6：黄绿色 100：其他颜色
+plateNum	可选	String	车牌号
+vehicleColor	必填	Integer	车身颜色 1：红色 2：绿色 3：蓝色4：黄色 5：白色 6：灰色 7：黑色 8：紫色 9：棕色 10：粉色 11：银色 12：深蓝 13：深灰 14：青色 100：其他颜色

objectExtInfo	可选	Json	eventType: 7 热成像-温度异常
+alarmContion	必填	Integer	报警条件 1：大于 2：等于 3：小于
+tempValue	必填	Float	温度值
+tempUnit	必填	Integer	温度单位 1：摄氏度 2：华氏度

objectExtInfo	可选	Json	eventType: 8 热成像-人体测温
+tempMax	必填	Float	最高温度
+tempUnit	必填	Integer	温度单位 1：摄氏度 2：华氏度

objectExtInfo	可选	Json	eventType: 10 电动车进出单元门

+type	必填	Float	1：进单元门 2：出单元门 100：未知

objectExtInfo	可选	Json	eventType: 11 违停
+type	必填	Integer	1：机动车 2：非机动车 100：未知
+plateColor	可选	Integer	车牌颜色 0：未知颜色 1：蓝色 2：黄色 3：白色 4：黑色 5：绿色 6：黄绿色 100：其他颜色
+plateNum	可选	String	车牌号
+vehicleColor	可选	Integer	车身颜色 1：红色 2：绿色 3：蓝色4：黄色 5：白色 6：灰色 7：黑色 8：紫色 9：棕色 10：粉色 11：银色 12：深蓝 13：深灰 14：青色 100：其他颜色

- 响应参数

参数	类型	说明
code	Integer	错误码
message	String	信息

- 示例代码

请求（人脸识别事件）

```
{
  "token":"cc85****0001",
  "ip":"192.168.1.2",
  "channel":1,
  "deviceType":3,
  "eventType":1,
  "eventTime":1631798285,
  "eventId":123,
  "eventStatus":0,
  "confidence":240,
  "scenePicBase64":"AABB****1122",
  "scenePicType":"JPG",
  "objectInfo":{
    "objPicBase64":"AABB****1122",
    "objPicType":"JPG",
    "leftTopY":100,
    "leftTopY":100,
    "rightBottomX":400,
    "rightBottomY":400,
    "objectExtInfo":{
     "similarity":0.65,
      "groupId":"12",
      "faceId":"1",
      "gender":1,
      "glass":1,
      "mask":0,
      "pitch":15,
      "yow":15,
      "roll":15,
      "faceQuality":95,
      "facePicBase64":"AABB****1122",
      "facePicType":"JPG",
    }
  }
}
```

返回

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

3.5 抓图计划

- 名称：edgeEventAdapter/capture/setTask
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
taskId	必填	String	任务的唯一标识
deviceList	必填	JSONArray	关联的监控设备
+ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
+channel	必填	Integer	通道号，从0开始，deviceType为3时有效
+deviceType	必填	Integer	设备类型 2：直连IPC 3：非直连IPC（NVR下IPC）

startDate	必填	String	计划开始时间 格式：YYYY-MM-DD
endDate	必填	String	计划结束时间 格式：YYYY-MM-DD
captureWeekDay	可选	JsonArray	抓图周期 captureWeekDay和cron二选一必填
+weekDay	必填	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
+captureTime	必填	List	抓图时分 24时制，格式：hh:mm
cron	可选	List	cron表达式 captureWeekDay和cron二选一必填
status	必填	Integer	启用状态 1：启用 0：禁用
extInfo	可选	String	扩展字段

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "taskId":"123",
  "deviceList":[{
    "ip":"192.168.1.2",
    "channel":1,
    "deviceType":3
  },{
    "ip":"192.168.1.2",
    "channel":2,
    "deviceType":3
  }],
  "startDate":"2010-01-01",
  "endDate":"2010-04-01",
  "captureWeekDay":[{
    "weekDay":"SUNDAY",
    "captureTime":"10:00,12:00"
  },{
    "weekDay":"MONDAY",
    "captureTime":"10:00,12:00"
  }],
  "status":1
}
```

返回

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

- 名称：edgeEventAdapter/capture/getTask
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
taskId	必填	String	任务的唯一标识

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息
data	必填	JSON	数据
+deviceList	必填	JSONArray	关联的监控设备
++ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
++channel	必填	Integer	通道号，从0开始，deviceType为3时有效

++deviceType	必填	Integer	设备类型 2：直连IPC 3：非直连IPC（NVR下IPC）
+startDate	必填	String	计划开始时间 格式：YYYY-MM-DD
+endDate	必填	String	计划结束时间 格式：YYYY-MM-DD
+captureWeekDay	可选	JsonArray	抓图周期 captureWeekDay和cron二选一必填
++weekDay	必填	String	周日期 星期日：SUNDAY 星期一：MONDAY 星期二：TUESDAY 星期三：WEDNESDAY 星期四：THURSDAY 星期五：FRIDAY 星期六：SATURDAY
++captureTime	必填	List	抓图时分 24时制，格式：hh:mm
+cron	可选	List	cron表达式 captureWeekDay和cron二选一必填
+status	必填	Integer	启用状态 1：启用 0：禁用
+extInfo	可选	String	扩展字段

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "taskId":"123"
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "deviceList":[{
      "ip":"192.168.1.2",
      "channel":1,
      "deviceType":3
    },{
      "ip":"192.168.1.2",
      "channel":2,
      "deviceType":3
    }],
    "startDate":"2010-01-01",
    "endDate":"2010-04-01",
    "captureWeekDay":[{
      "weekDay":"SUNDAY",
      "captureTime":"10:00,12:00"
    },{
      "weekDay":"MONDAY",
      "captureTime":"10:00,12:00"
    }],
    "status":1
  }
}
```

- 名称：edgeEventAdapter/capture/deleteTask
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
taskId	必填	String	任务的唯一标识

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "taskId":"123"
}
```

返回

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

- 名称：edgeSecurityEvent/capture/report
- 方向：事件适配应用->智能安防应用
- 版本：1.0.0
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
channel	必填	Integer	通道号，从0开始，deviceType为3时有效
deviceType	必填	Integer	设备类型 2：直连IPC 3：非直连IPC（NVR下IPC）
taskId	可选	String	任务ID
captureTime	必填	Long	抓图时间，UTC时间，精确到s
picBase64	必填	String	抓拍图片base64
picType	必填	String	图片类型 JPG PNG BMP

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "ip":"192.168.1.3",
  "channel":1,
  "deviceType":3,
  "captureTime":1631798285,
  "picBase64":"AABB****1122",
  "picType":"JPG"
}
```

返回

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

3.6 抓拍记录查询

- 名称：edgeEventAdapter/record/get
- 方向：智能安防应用->事件适配应用
- 版本：1.0.1
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
picBase64	必填	String	图片base64
picType	必填	String	图片类型 JPG PNG BMP
recordLimit	必填	Integer	获取记录数量限制
startTime	必填	Long	UTC时间，精确到s
endTime	必填	Long	UTC时间，精确到s endTime与startTime相差<=24h
minSimilarity	必填	Float	最小相似度(0,1)
maxSimilarity	必填	Float	最大相似度(0,1) maxSimilarity > minSimilarity
pageNo	可选	Integer	分页查询，页码，从1开始 默认为1
pageSize	可选	Integer	分页查询，页尺寸，pageSize>1&pageSize<=20 默认为10

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息
data	必填	Json	数据
+totalSize	必填	Integer	查询结果总数，注意，不是当页结果，是所有页的结果总和
+recordList	必填	JsonArray	
++ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
++channel	必填	Integer	通道号，从0开始，deviceType为3时有效
++deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC（NVR下IPC）

++smallPicBase64	必填	String	小图base64
++smallPicType	必填	String	图片类型 JPG PNG BMP
++scenePicId	必填	String	场景图id
++similarity	必填	Float	相似度(0,1)
++captureTime	必填	Long	UTC时间，精确到s
++extinfo	可选	JsonArray	眼镜，性别，年龄等

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "picBase64":"AABB****1122",
  "picType":"JPG",
  "recordLimit":10,
  "startTime":1631794685,
  "endTime":1631798285,
  "minSimilarity":0.55,
  "maxSimilarity":0.65
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "totalSize":100,
    "recordList":[{
      "ip":"192.168.1.3",
      "channel":1,
      "deviceType":3,
      "smallPicBase64":"AABB****1122",
      "smallPicType":"JPG",
      "scenePicId":"pic1234",
      "similarity":0.6,
      "captureTime":1631795685
    },{
      "ip":"192.168.1.3",
      "channel":1,
      "deviceType":3,
      "smallPicBase64":"AABB****1122",
      "smallPicType":"JPG",
      "scenePicId":"pic1235",
      "similarity":0.6,
      "captureTime":1631795685
    }]
  }
}
```

- 名称：edgeEventAdapter/record/scenePicUpload
- 方向：智能安防应用->事件适配应用
- 版本：1.0.1
- 请求方式：POST
- 请求参数

参数	是否必填	类型	说明

token	必填	String	令牌信息
scenePicId	必填	String	场景图ID

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息
data	必填	Json	数据
+picBase64	必填	String	场景图base64
+picType	必填	String	图片类型 JPG PNG BMP

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "scenePicId":"pic1234",
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "picBase64":"AABB****1122",
    "picType":"JPG"
  }
}
```

- 名称：edgeEventAdapter/record/getAll
- 方向：智能安防应用->事件适配应用
- 版本：1.0.0
- 请求方式：GET
- 请求参数

参数	是否必填	类型	说明
token	必填	String	令牌信息
picBase64	必填	String	图片base64
picType	必填	String	图片类型 JPG PNG BMP
startTime	必填	Long	UTC时间，精确到s

endTime	必填	Long	UTC时间，精确到s endTime与startTime相差<=24h
minSimilarity	必填	Float	最小相似度(0,1)
maxSimilarity	必填	Float	最大相似度(0,1) maxSimilarity > minSimilarity

- 响应参数

参数	是否必填	类型	说明
code	必填	Integer	错误码
message	必填	String	信息
data	必填	Json	数据
+recordList	必填	JsonArray	
++ip	必填	String	设备地址，deviceType为3时，ip为NVR的IP
++channel	必填	Integer	通道号，从0开始，deviceType为3时有效
++deviceType	必填	Integer	设备类型 1：NVR 2：直连IPC 3：非直连IPC（NVR下IPC）
++similarity	必填	Float	相似度(0,1)
++captureTime	必填	Long	UTC时间，精确到s
++extinfo	可选	JsonArray	眼镜，性别，年龄等

- 示例代码

请求

```
{
  "token":"cc85****8808",
  "picBase64":"AABB****1122",
  "picType":"JPG",
  "recordLimit":10,
  "startTime":1631794685,
  "endTime":1631798285,
  "minSimilarity":0.55,
  "maxSimilarity":0.65
}
```

返回

```
{
  "code":200,
  "message":"success",
  "data":{
    "recordList":[{
      "ip":"192.168.1.3",
      "channel":1,
      "deviceType":3,
      "similarity":0.6,
      "captureTime":1631795685
    },{
      "ip":"192.168.1.3",
      "channel":1,
      "deviceType":3,
      "similarity":0.6,
      "captureTime":1631795685
    }]
  }
}
```

4. 附录

4.1 错误码

code	message	说明
200	success	成功
801	adapter authentication failed	适配应用认证失败
802	adapter not authentication	适配应用未认证
803	device certificate not exist	设备证书不存在
804	param check error: /'%s/'	参数校验失败，返回失败的参数
805	memory exception	内存异常
806	json data wrong format	json数据格式错误
807	device not exist	设备不存在
808	procotol not support	不支持的协议
809	manufacture error	设备厂商错误
810	abnormal communication	模块通讯异常

4.2 应用要求

硬件要求：

- cpu（2.0 Ghz）： < 0.2core
- 内存： < 1Gb
- 磁盘： < 1Gb

服务要求：

- 本地可持久化数据：

智能安防应用的IP和端口号信息。

事件布防信息。

日志数据。

- 日志：日志中需带有级别、时间信息，且需提供以下项目：接口日志（接入、配置、告警）、设备状态日志、运行异常日志。日志可过滤。

4.3 修改记录

版本号	修改内容	修改时间	责任人
Rev0.1	文档创建	2021-09-01	诚治
Rev0.2	1. 新增、修改人脸库接口，增加通道字段 2. 修改人脸库接口，去掉groupType字段 3. 3.6.2章节增加uploadUrl说明	2022-01-18	诚治
Rev0.3	1. 修改“3.6.1 抓拍记录查询”接口，返回信息增加查询总数字段 2. 修改“3.6.2 抓拍记录场景图获取”接口，下发的uploadUrl改为返回downloadUrl	2022-02-16	诚治
Rev0.4	1. 修改“3.4.3 告警事件上报”接口，类型增加“其他”	2022-02-22	诚治
Rev0.5	1. 修改“3.2.1 获取所有设备配置”接口，子IPC增加用户名密码	2022-03-23	诚治

3.5.4. EBA标准

3.5.4.1. EBA设备接入方案介绍

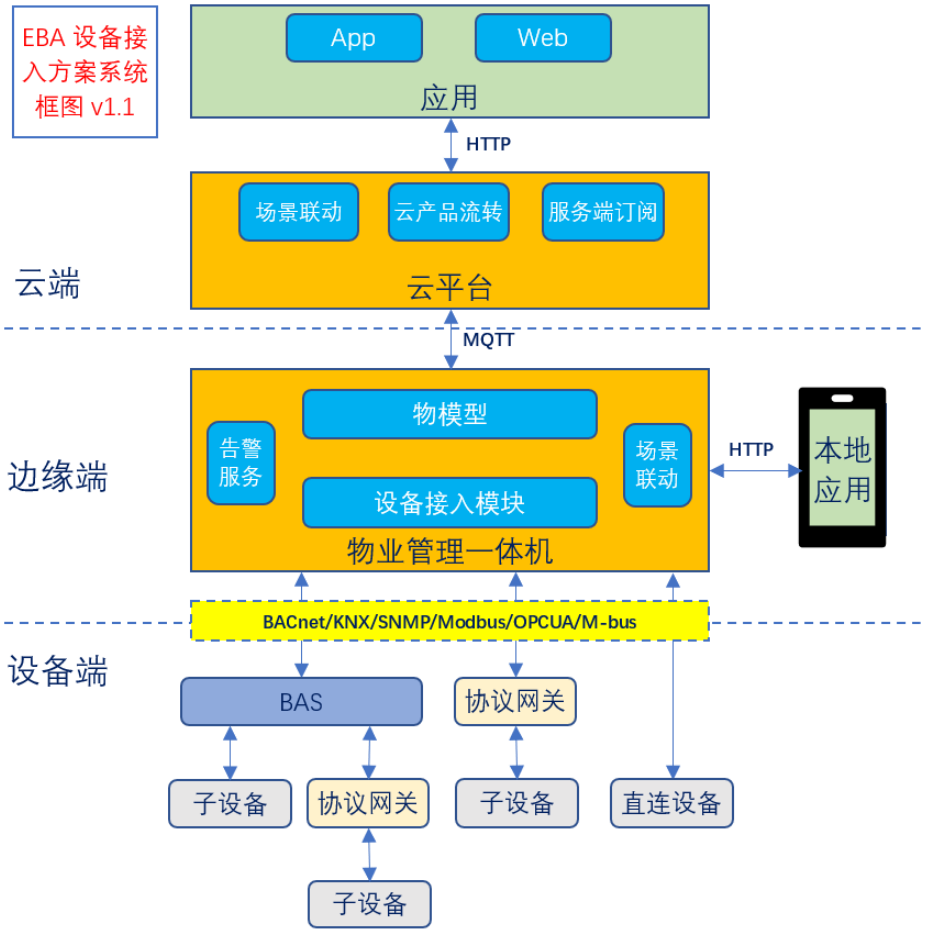
1．概述

本文档针对EBA设备厂商技术人员，目的是使本地的**楼宇设备或系统（支持BACnet、Modbus、KNX、SNMP等协议的设备或系统）**可以接入到阿里云平台。

本方案可以将不同品牌的楼宇设备产生的数据进行标准化，实现数据的统一管理，并提供规则引擎和云平台应用接口，使数据可以流转到不同的应用里。

本方案对原有的硬件设备基本不侵入，只需使设备网网关入口连接到物业管理一体机，即可实现对设备的监控以及操作管理，提升管理效率。

2．系统框架图



图解说明：

①云端：包括物联网云平台和应用。云平台向物业管理一体机提供物模型及边缘端场景联动、告警服务等功能，向云端应用提供云端场景联动，云产品流转及服务端订阅等功能，可向应用转发设备上报的数据，并且可实现自动化业务逻辑。

②物业管理一体机：物业管理一体机内主要包含LE组件、设备接入模块及其他边缘应用。设备接入模块提供设备管理，连接管理及物模型数据转换等功能；边缘端场景联动和告警服务可提供本地自动化业务逻辑。

③设备端：支持多种设备网组态。可使BA系统以及子设备接入云平台，也可使具有网络通讯能力的协议网关及子设备、直连设备接入云平台。目前物业管理一体机已支持标准协议如BACnet/Modbus/KNX/SNMP/M-BUS/OPCUA。

3．各协议版本

物业管理一体机已支持的各协议版本及通信模式如下：

- BACnet：符合ASHRAE 135-2016 Standard标准，支持MS/TP和IP两种模式。
- KNX：符合GB/Z 20965 2007标准，支持TP1和IP两种模式。
- SNMP：符合SNMP标准，支持SNMPV1、SNMPV2、SNMPV3版本。
- Modbus：符合Modbus标准，支持RTU和TCP两种模式。
- OPCUA：符合IEC 62541标准，支持TCP和HTTP两种模式。
- M-BUS：符合C7/T 188-2004标准，支持UART和TCP两种模式。

4．对接流程

4.1 设备要求

4.1.1 直连设备

- 提供通讯协议，且符合物业管理一体机支持的物理总线

4.1.2 BAS/协议网关

- 提供通讯协议，且符合物业管理一体机支持的物理总线
- BAS或协议网关需要为内部每一个子设备提供一个全局设备ID，供物业管理一体机定位和访问设备，使其内部子设备达到与物业管理一体机数据交互的功能

4.2 协议要求

推荐三种设备接入协议：BACnet、Modbus、KNX。

对3大协议的基本技术要求如下：

序号	协议种类	物理总线	总线长度推荐	总线设备数量限定	地址码及设定	物业管理一体机接口能力
1	Modbus RTU	Rs-485 总线选用2芯屏蔽双绞线	理论距离1200米最好控制在（400米以内）	理论上256个，常规使用时一条总线上不超过32个	需要提供device ID和点位表	以Modbus RTU通信方式接入
2	Modbus TCP	网线UTP	80~100米	无	需要提供IP地址，端口号和点位表	以Modbus TCP方式接入
3	KNX	4芯屏蔽双绞线	支线最大长度1000米	单个支线不超过64个设备，一个主线上最多包含15条支线	提供设备ID对照表	以KNX-IP方式接入
4	BACnet MS/TP	Rs-485 总线选用2芯屏蔽双绞线	理论距离1200米最好控制在（400米以内）	理论上256个，常规使用时一条总线上不超过32个	需要提供device ID和点位表	以BACnet MS/TP方式接入
5	BACnet-IP	网线UTP	80~100米	无	需要提供device ID,IP 端口号和点位表	以BACnet-IP方式接入

4.3 设备信息收集

设备信息表模板示例

下图为Modbus设备信息表模板

Modbus是一种串行通信协议，是Modicon公司（现在的施耐德电气Schneider Electric）于1979年为使用可编程逻辑控制器（PLC）通信而发表。Modbus已经成为工业领域通信协议的业界标准（De facto），并且现在是工业电子设备之间常用的连接方式。Modbus比其他通信协议使用的更广泛的主要原因有：1.公开发表并且无版权要求；2.易于部署和维护；3.对供应商来说，修改移动本地的比特或字节没有很多限制
Modbus允许多个 (大约240个) 设备连接在同一个网络上进行通信，举个例子，一个由测量温度和湿度的装置，并且将结果发送给计算机。

项目方提供					设备方提供			阿里提供	设备方提供			
设备IOT编码（设备id）	设备名称(支持英文字母、数字和特殊字符"-"、"_"、@、"."、":"、"."，长度限制4~32)	模型名称（是否采用已有标准模型；可不选）	连接方式（TCP或rtu）	点位上报周期（单位:ms）	modbus地址（设备从站号）	设备IP（传输方式为TCP时填写）	设备端口（传输方式为TCP时填写）	串口名（传输方式为RTU时填写）	串口波特率（传输方式为RTU时填写）	数据位（传输方式为RTU时填写）	校验方式（传输方式为RTU时填写）	停止位（传输方式为RTU时填写）
SHSB-000131	五期综合泵房1号高区生活水泵	遥信遥控智能仪表	TCP	10000	29	10.249.16.11	5101					
SHSB-000136	五期综合泵房1号中区生活水泵	遥信遥控智能仪表	TCP	10000	30	10.249.16.11	5101					
SHSBKZJX-000015	五期综合泵房高区生活水泵电参数	智能电力仪表	TCP	10000	47	10.249.16.11	5101					
SHSBKZJX-000016	五期综合泵房高区生活水泵电参数	智能电力仪表	TCP	10000	48	10.249.16.11	5101					
SHSBKZJX-000017	五期综合泵房高区生活水泵电参数	智能电力仪表	RTU	10000	49			/dev/ttyS0	9600	8	无校验	1

设备点位信息表模板示例

下图为Modbus点位信息表模板：

Modbus是一种串行通信协议，是Modicon公司（现在的施耐德电气Schneider Electric）于1979年为使用可编程逻辑控制器（PLC）通信而发表。Modbus已经成为工业领域通信协议的业界标准（De facto），并且现在是工业电子设备之间常用的连接方式。Modbus比其他通信协议使用的更广泛的主要原因有：1.公开发表并且无版权要求；2.易于部署和维护；3.对供应商来说，修改移动本地的比特或字节没有很多限制
Modbus允许多个 (大约240个) 设备连接在同一个网络上进行通信，举个例子，一个由测量温度和湿度的装置，并且将结果发送给计算机。

项目方提供						设备方提供								
设备IOT编码（设备id）	功能编号（某个设备中不同功能的编号）	功能类型（Modbus只支持属性）	功能名称（例如：用电量。支持中文、英文字母、日文、数字、短划线（-）、下划线（_）、正斜线（/）和点号（.），且必须以中文、英文、日文或数字开头，不超过30个字符。）	功能标识符（支持英文、数字和下划线（_），不超过50个字符）	数据上报方式	操作类型（操作Modbus的功能码）	寄存器地址（十六进制，0x开头）	原始数据类型（Data Type）	取值范围最小值（原始数据经过缩放因子处理之后的取值范围，超出取值范围的数据会被丢弃）	取值范围最大值（原始数据经过缩放因子处理之后的取值范围，超出取值范围的数据会被丢弃）	单位（根据实际情况选择，可选无）	是否交换寄存器内16位数据的高低字节	是否交换寄存器内32位数据顺序	缩放因子（真实数据将采集数据相比的缩放系数，不能为0，默认为1）
SHSB-000131	1	属性	第一路漏接点状态	Line1Status	按时上报	保持寄存器（只读，0x03）	0x09	uint16	0	1	无	否	否	1
SHSB-000131	2	属性	第三路漏接点状态	Line3Status	按时上报	保持寄存器（只读，0x03）	0x0b	uint16	0	1	无	否	否	1
SHSB-000131	3	属性	设备地址	DeviceAddress	按时上报	保持寄存器（只读，0x03）	0x1c	uint16	1	31	无	否	否	1
SHSB-000136	1	属性	第一路漏接点状态	Line1Status	按时上报	保持寄存器（只读，0x03）	0x09	uint16	0	1	无	否	否	1
SHSB-000136	2	属性	第三路漏接点状态	Line3Status	按时上报	保持寄存器（只读，0x03）	0x0b	uint16	0	1	无	否	否	1
SHSB-000136	3	属性	设备地址	DeviceAddress	按时上报	保持寄存器（只读，0x03）	0x1c	uint16	1	32	无	否	否	1
SHSBKZJX-000015	1	属性	系统频率	PowerFrequency	按时上报	保持寄存器（只读，0x03）	0x9112	uint16	0	2147483647	赫兹	否	否	0.01
SHSBKZJX-000015	2	属性	三相功率因数	TotalPowerFactor	按时上报	保持寄存器（只读，0x03）	0x9110	float	0	2147483647	无	否	否	1
SHSBKZJX-000015	3	属性	C相电流	C_Current	按时上报	保持寄存器（只读，0x03）	0x910a	float	0	2147483647	安培	否	否	1
SHSBKZJX-000016	1	属性	系统频率	PowerFrequency	按时上报	保持寄存器（只读，0x03）	0x9112	uint16	0	2147483647	赫兹	否	否	0.01
SHSBKZJX-000016	2	属性	三相功率因数	TotalPowerFactor	按时上报	保持寄存器（只读，0x03）	0x9110	float	0	2147483647	无	否	否	1
SHSBKZJX-000016	3	属性	C相电流	C_Current	按时上报	保持寄存器（只读，0x03）	0x910a	float	0	2147483647	安培	否	否	1

不同协议的设备需提供的设备及点位的信息不同

- BACnet-IP：需提供设备deviceId、max_apdu长度、设备IP及端口号。点位信息需提供对象类型、对象ID、属性ID及数据类型。
- BACnet-MS/TP：需提供设备通信波特率、MS/TP_ID、设备deviceid。点位信息需提供对象类型、对象ID、属性ID及数据类型。

- Modbus-TCP：需提供设备IP、端口号及设备从站地址。点位信息需提供寄存器操作类型、寄存器地址、原始数据类型、是否交换数据高低字节、缩放因子。
- Modbus-RTU：需提供设备通信波特率、数据位、校验位、停止位及设备从站地址。点位信息需提供寄存器操作类型、寄存器地址、原始数据类型、是否交换数据高低字节、缩放因子。
- KNX-IP：需提供设备地址。点位信息需提供写操作群组地址、读操作群组地址及数据类型。
- KNX-TP1：需提供设备IP、端口号及设备地址。点位信息需提供写操作群组地址、读操作群组地址及数据类型。
- SNMP-v2：需提供设备community名。点位信息需提供oid及数据类型。
- SNMP-v3：需提供设备加密方式、用户名、认证算法及认证密码。点位信息需提供oid及数据类型。
- Mbus-UART：需提供设备通信波特率、设备类型、设备地址、是否加密。点位信息需提供点位功能含义和支持服务类型。
- Mbus-TCP：需提供设备IP、端口号、设备类型、设备地址、是否加密。点位信息需提供点位功能含义和支持服务类型。
- OPCUA：需提供设备通信地址、安全策略类型、安全模式类型、用户名及密码、设备节点路径。点位信息需提供节点名称。

4.4 设备接入

根据收集的设备信息，在云平台创建并配置设备及功能点位，然后将设备信息及物模型部署到物业管理一体机，使物业管理一体机内的设备接入模块可以与设备正常通信，实现设备上云及设备管理的功能。

3.6. 常见问题

3.6.1. 网络端常见问题

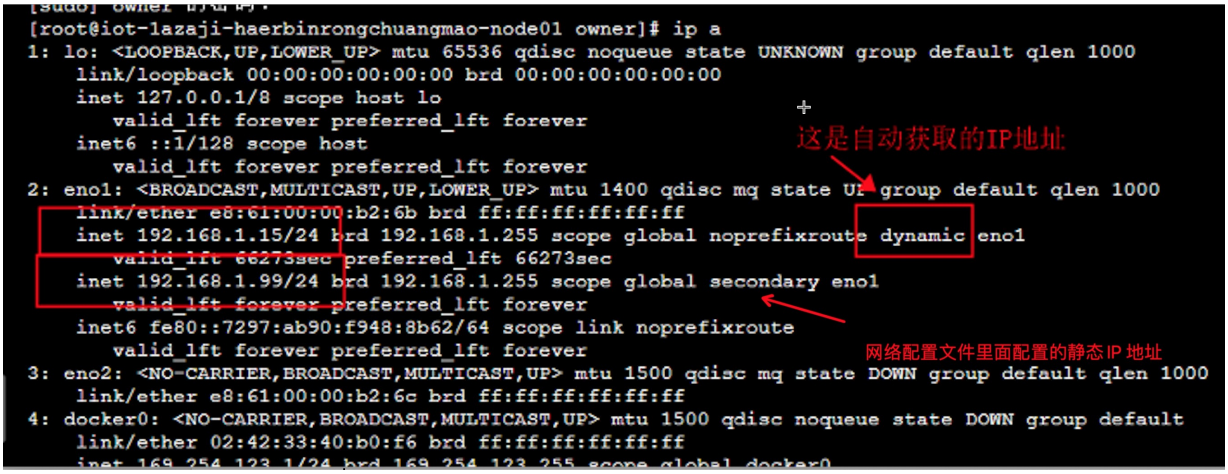
3.6.1.1. 边缘物业一体机静态IP配置错误

1. 问题背景

客户购买边缘物业一体机之后,先完成物业一体机的初始化部署,核心的目标是通电、通网； 物业一体机的IP 地址阿里云推荐使用静态IP 地址，需要修改网络配置文件，但有时出现手动配置错误，导致静态IP 不生效； 在物业一体机重启之后对外的网络出现不稳定

2. 问题现象

- 物业一体机重启之后,通过ifconfig 查看网络会出现2个IP 地址
- DHCP 自动分配的IP 地址
- 网络配置文件里面配置的静态IP 地址 【可能因为配置错误，导致没有生效】



3. 问题排查

3.1 查看网卡对应的配置文件



- DNS1 推荐的配置统一是: 223.5.5.5 ， 把上面的192.168.1.1 修改成: 223.5.5.5

3.2 重启网卡服务或者一体机

- 修改网络配置, 需要重启网卡才能生效； 重启网卡服务需要远程SSH 登录一体机，切换到root 账号或者sudo 来执行命令
- 因为不同的机型可能网卡服务配置不一致, 所以比较推荐的方式是直接重启物业一体机(特别是对未运行业务的机器)
- 远程SSH 登录后， sudo reboot
- 或者现场直接重启服务器

3.6.1.2. 边缘物业一体机网络监测

1. 问题背景

- 在边缘物业一体机在配置好局域网IP 之后, 要确保一体机在局域网跟设备通信正常，跟公网的阿里云产品通信； 而且局域网不稳定性导致一体机离线，或者网络不通

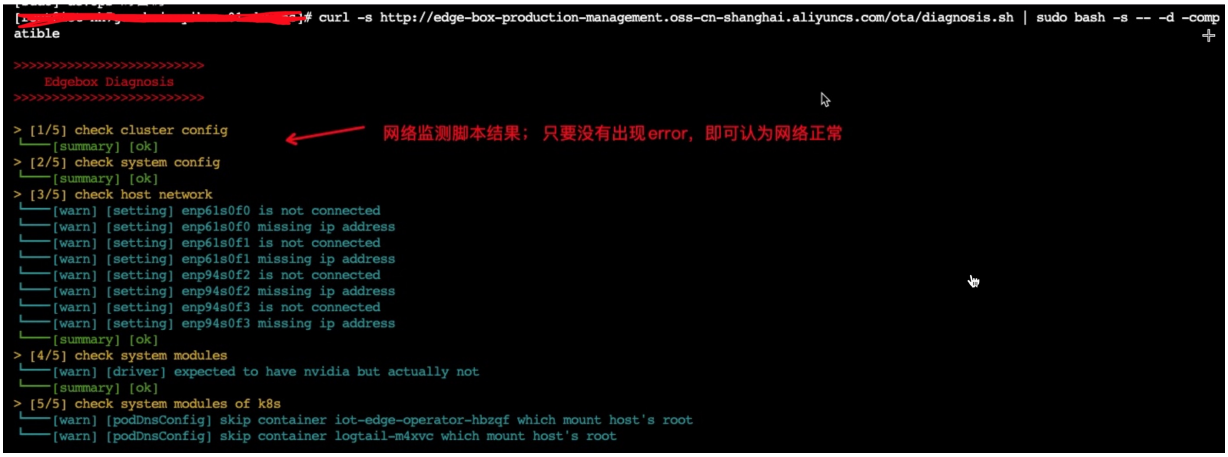
2. 网络监测脚本

- 阿里云IoT 基于网络的问题开发一个网络监测工具, 核心的监测列表:
- 局域网内IP 通行是否正常
- 一体机跟阿里云公网DNS 通信是否正常

- 脚本下载:

```
curl -s http://edge-box-production-management.oss-cn-shanghai.aliyuncs.com/ota/diagnosis.sh | sudo bash -s -- -d -compatible
```

- 监测结果:



3. 网络监测脚本出现error

- 如果出现error 则说明某个网络model 出现异常，需要网络施工团队进行监测和排查
- 如果需要阿里云IoT 协助，可以提供工单咨询或者钉钉群里找相关同学咨询

3.6.1.3. 边缘物业一体机离线排查

1. 问题现象

- 在物联网平台里面显示物业一体机离线状态, 如下图:



- 物业一体机在线的判断条件是下面三个条件进行与逻辑 [只有三个条件都是1 才显示在线, 其中一个条件是0 就是显示离线]
- 本地网络跟公网通信正常： ping 223.5.5.5 [阿里云提供的公网DNS]
- LinkEdge 网关(设备接入的网关)：必须在线
- 物业一体机是云端k8s 集群的一台worker，实时上报状态; k8s 状态正常
- 上面每一个条件离线都会导致物业一体机离线，但个单独的离线对业务的影响不同，下面排查思路里面会具体说明

2. 排查方案

2.1 局域网通信是否正常

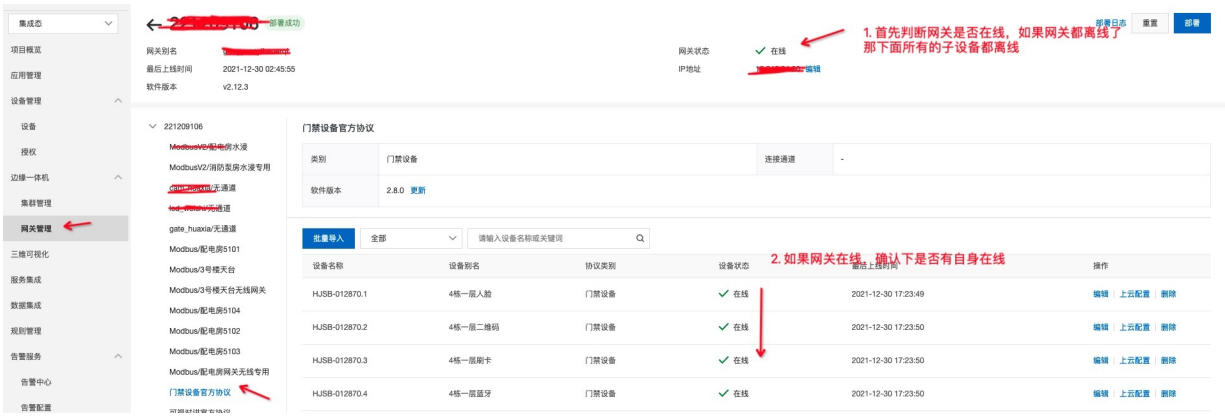
- 如果本地局域网络出现异常, 那对业务的影响是最大的；相当于子设备跟物业一体机的网关连接和通行都异常；
- 几乎所有的业务都不可用；修复办法也只能等待本地网络修复；
- 云端或者远程没有任何好的办法来缓解或者修复
- 具体测试方法：
- 本地2台设备互相 ping IP 地址
- 或者本地网络工程师根据路由器、交换机去判断本地网络问题

2.2 局域网通信正常, 当公网不通

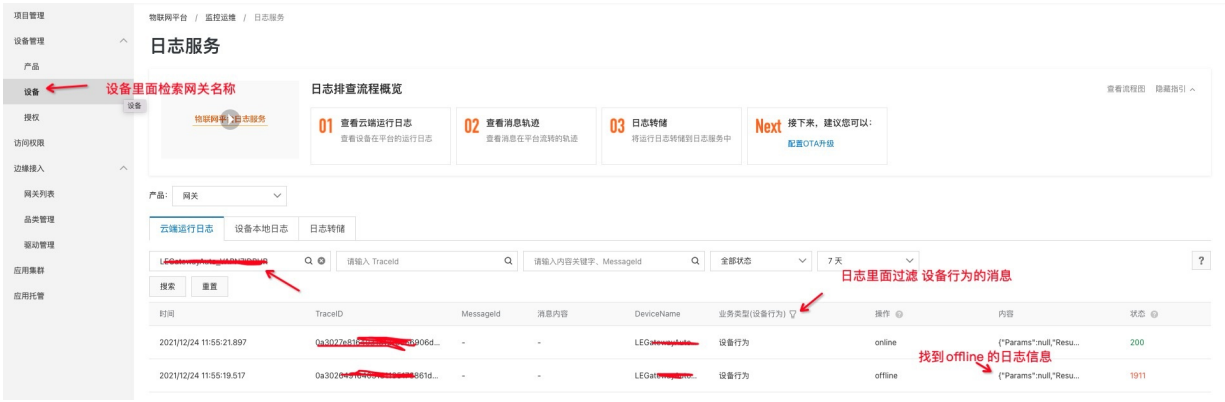
- 局域网通信正常，但公网不通
- 此时在边缘端闭环的业务正常
- 刷脸通行/刷卡通行/二维码通行等，但通行事件上云业务异常
- 固定车位(购买或者长期租车位，不用涉及缴费车辆)正常通行 ；但临时停车缴费业务异常
- 边缘端(小区保安监控)正常运行， 但云端拉流存储、安防事件通知业务异常
- 排查方法：
- 方法一: 远程或者本地SSH 到物业一体机，ping 223.5.5.5 来验证网络是否正常
- 方法二: 用上一个章节里面网络监测脚本来监测
- 修复方案：
- 首先判断: 局域网本身跟公网通行是否正常, 如果局域网都不能通公网，那只能本地网络团队来修复
- 如果本地局域网可以正常通信, 那需要通过网络监测脚本来监测具体的ERROR，根据错误信息来修复；
- 这种情况，可以联系阿里售后来支持快速修复公网通信

2.3 如果网络都正常，LinkEdge 离线

- 首先登录物联网应用服务平台, 查看集群网关和子设备是否在线
- 如果网关设备和子设备都离线，那核心业务都会受影响
- 可以认定为: 人行/车行/安防业务都异常



- 如果网关离线，需要去查看下网关(网关也是一种设备)离线的日志：
- 在物联网平台日志服务查询网关的上上下线日志



2.4 如果网络正常, LinkEdge 网关也在线

- 那只有是第三种情况： k8s 状态离线
- 这种情况基本的核心业务不受影响，只影响物业一体机作为k8s 的woker 不能正常上报master 实时状态
- 这种情况只能离线阿里技术支持来修复

3.6.2. 设备端常见问题

3.6.2.1. 单台设备离线排查

1. 确认网络是否异常

- 设备离线首先排查这台设备的网络是否正常，排查办法也简单：
- 获取到这台设备的IP 地址,远程或者本地SSH 物业一体机, 然后ping 设备IP 地址
- 如果网络不通, 那肯定是离线的； 只能去排查网络问题
- 如果网络正常,就继续进行下面第2 步继续排查
- 查询设备的IP 地址的方法：



2. 设备离线驱动日志

2.1 设备接入网关的驱动协议

- 登录 物联网应用服务平台 找到设备, 然后根据设备的品类(人行设备/车行设备/安防设备/EBA设备)找到对应的驱动协议
- 查询设备对应驱动协议的方法:

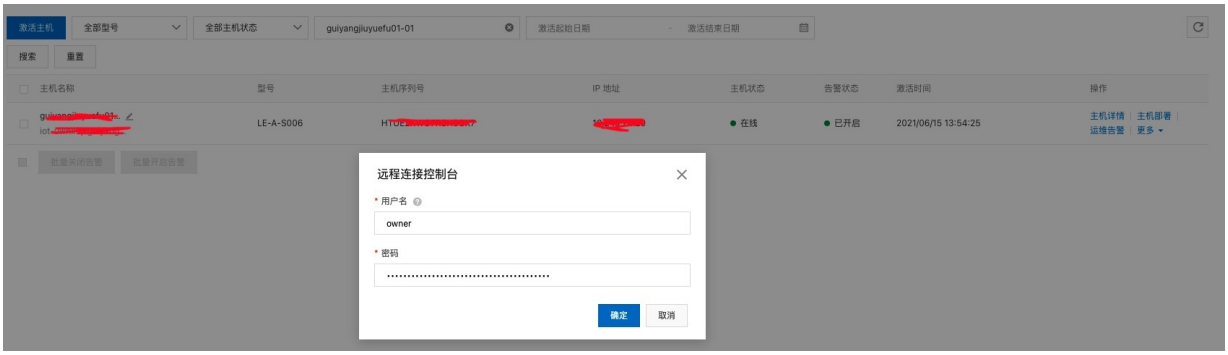


2.2 驱动日志的路径

- 远程登录物业一体机 【账号和密码问下施工的人员】



- 或者到边缘一体机控制台 检索到自己的一体机然后进行远程SSH:



- 进入到驱动日志的目录

```
cd /linkedge/run/logger/fc-base/

cd DeviceAccessControl
```

- 通过2.1 获取到的驱动协议 [DeviceAccessControl]

```
[devops@iot-wkenhn-guanlanhu-01 ~]$ cd /linkedge/run/logger/fc-base/
[devops@iot-wkenhn-guanlanhu-01 fc-base]$ ls
DeviceAccessControl          log_2021-03-10-12-29-21-1.txt  log_2021-03-31-12-21-46-1.txt  log_2021-11-30-19-50-21-1.txt  MSJNet_AGB3_EBA
jieshun_parking              log_2021-03-11-16-48-44-1.txt  log_2021-04-19-18-08-21-1.txt  log_2021-11-30-22-57-33-1.txt  ServerAccessControl
log_2021-03-10-11-52-03-1.txt  log_2021-03-11-18-13-58-1.txt  log_2021-06-08-15-30-24-1.txt  log_2021-12-14-11-45-49-1.txt
log_2021-03-10-11-58-51-1.txt  log_2021-03-16-14-43-16-1.txt  log_2021-10-20-17-11-18-1.txt  log_2021-12-14-12-10-52-1.txt
log_2021-03-10-12-05-34-1.txt  log_2021-03-31-12-00-22-1.txt  log_2021-11-30-19-31-55-1.txt  log.INFO
[devops@iot-wkenhn-guanlanhu-01 fc-base]$
```

- 用设备的DN 来检索驱动日志即可,如果不熟悉可以让写驱动代码的组织来排查具体日志的含义:

```
grep xxxxxxx-000629.1 log.INFO
```

3.6.2.2. 设备服务调用错误

1. 问题背景&现象

1.1 问题背景

- IoT 设备都是智能化设备，也就是设备里面会嵌入代码，并提供HTTP 或者RPC服务；设备联网之后被被调用，设备收到请求之后会按照代码来控制设备做响应的工作。
- 所以设备提供的服务在调用时也可能会出错,设备日志调试难度比较大，需要专业的环境；而且设备批量出货后出现问题的概率不是很大；
- 大概率是现场使用的设备方式可能不对,此时就先确认这款设备是否提供了这个服务

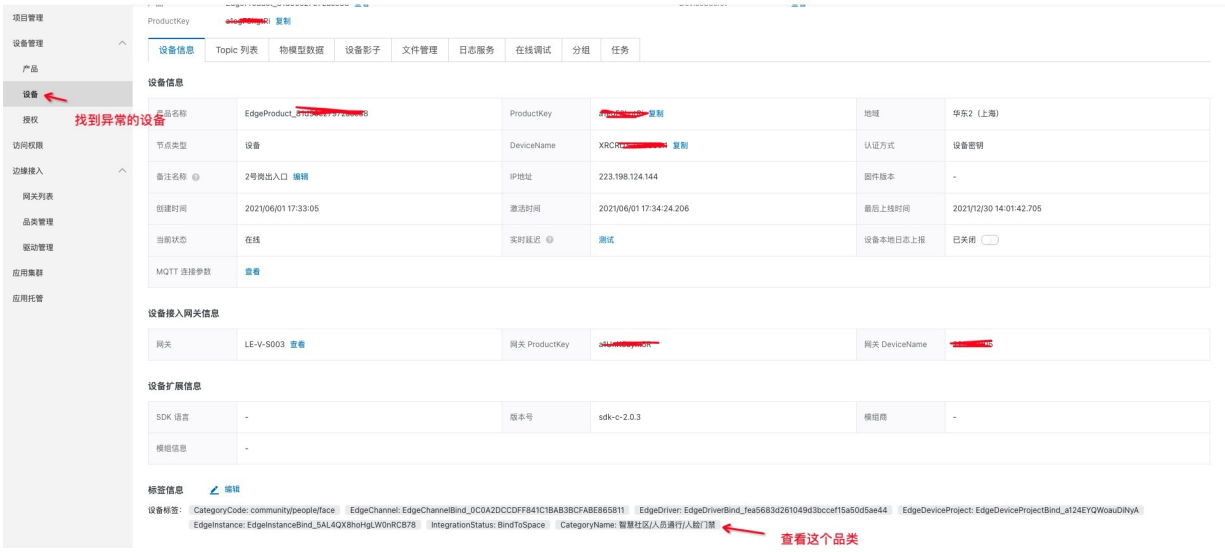
1.2 设备服务调用异常现象

- 一般在通过云端或者APP 调用设备的服务时会报下面的错误
- device data format is error
- 一般这个错误就是设备本身的问题，更可能是设备本身就没有提供这个服务 【设备在创建时使用的品类物模型不正确】

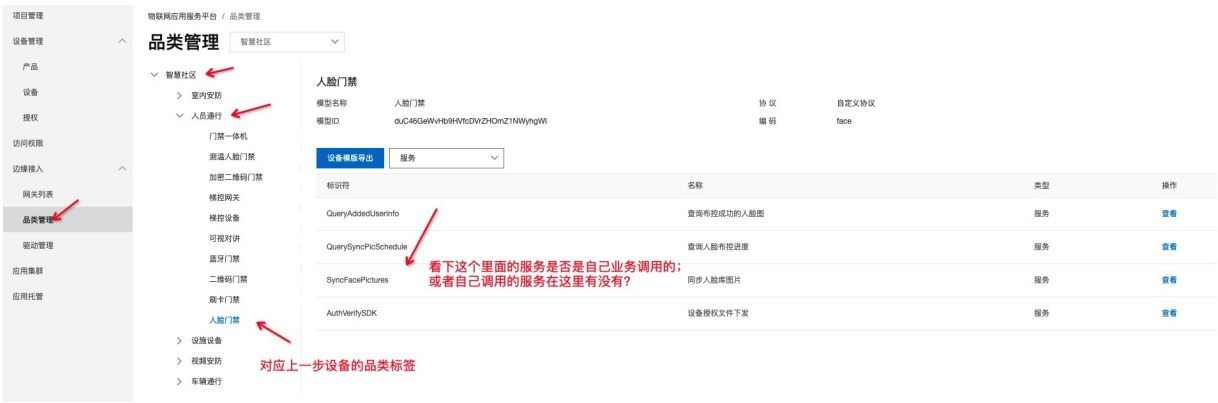
2. 问题排查方法

2.1 设备是否提供了这个服务

- 设备在定义物模型时定义了属性/事件和服务； 对于智慧社区的项目,直接选择品类来来创建的
- 1. 登录 物联网应用服务平台,查看设备对应的品类



- 查看对应品类里面是否提供调用的服务



2.2 如果品类用的不对，修复方案

- 这种情况一般都是品类使用不对或者业务saas调用设备服务使用的API 不对;
- 如果设备品类不对，那需要删除设备之后重新在创建设备，选择正确的品类即可

3.6.3. 边缘物业一体机常见问题

3.6.3.1. 边缘物业一体机网络配置错误

1. 问题现象

1.ping不通网关和外网

2.ping不通部分外网

3.可以ping通外网，但ip不是自己配置的那个，或者是一个网卡有两个ip

4.DNS1配置的不是223.5.5.5

2. 排查方法

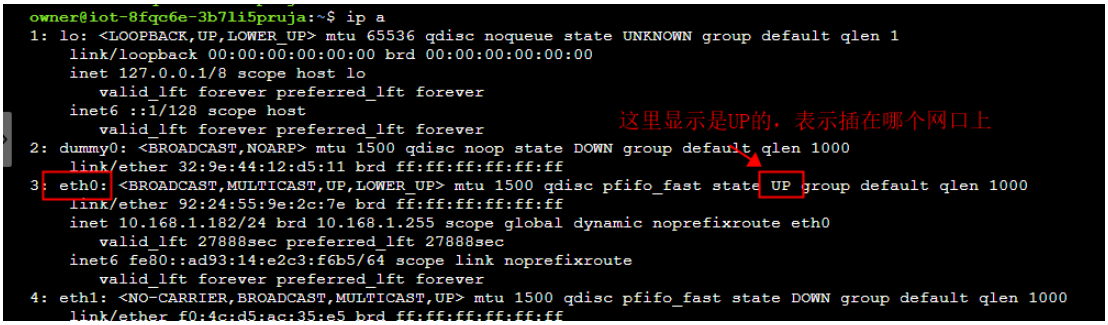
2.1网络配置之前需要注意以下几点：

1.插上网线，一般插在LAN1口，插上电源，开机

2.准备好显示器和键盘鼠标

3.登录进服务器，初始账号密码：owner/changepassword

4.输入ip a 查看网线连接在哪个网口上，如果没有网卡显示UP，那就说明网线有问题或者是服务器插口有问题了，要逐一排查一下



2.2排查思路

1.ping不通网关，基本可以确认是ip有问题了，这是就要和甲方或者网工确认正确的IP地址和网关地址

2.ping不通部分ip，比如ping不通223.5.5.5，或者是ping不通百度等，这说明是有防火墙了，可以问一下甲方和网工，让他们开通白名单

3.可以ping通外网，但不是自己配置的IP，就要看一下配置文件中有没有写错，或者是自动获取IP地址没有关，或者是配置完文件，没有重启网卡，配置完网络文件，一定要输入: ip a 看一下配置的对不对

4.DNS1一定要配置成223.5.5.5，可能激活时没有问题，后续会有别的问题发生

3.配置网络接入（Linux下配置，后面版本可能会移植配置入口）详情见《如何配网.PDF》手册

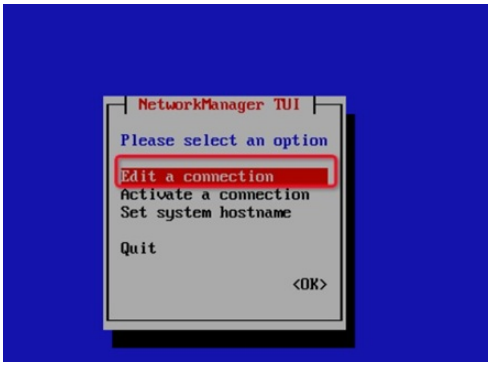
3.1通过页面配置（推荐用这个方法）

步骤一：如下键入nmtui



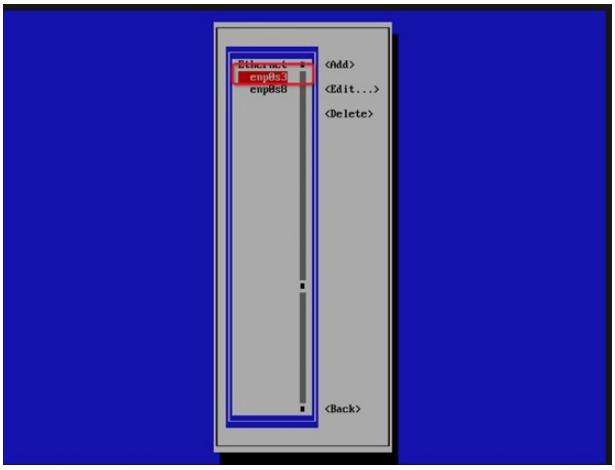
步骤二：选择Edit a connection

按「回车」进入连接配置详情页（有得版本是中文显示，这个无影响）



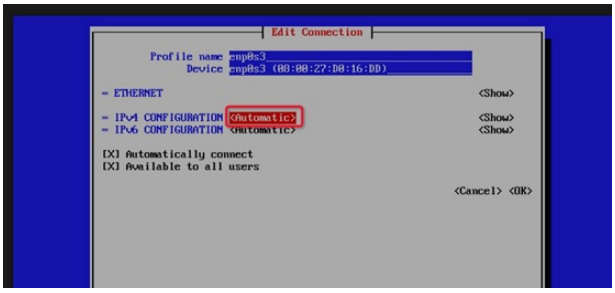
步骤三：选择要修改得网卡信息

1.1中有提到网卡名称，可在配置IP前，用ip a确定网卡信息），这里以enp0s3为例，回车确认。

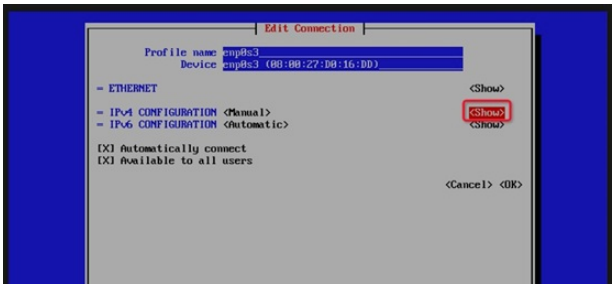


步骤四：配置静态IP

将IPV4 CONFIGURATION配置模式改为Manual（回车键修改）



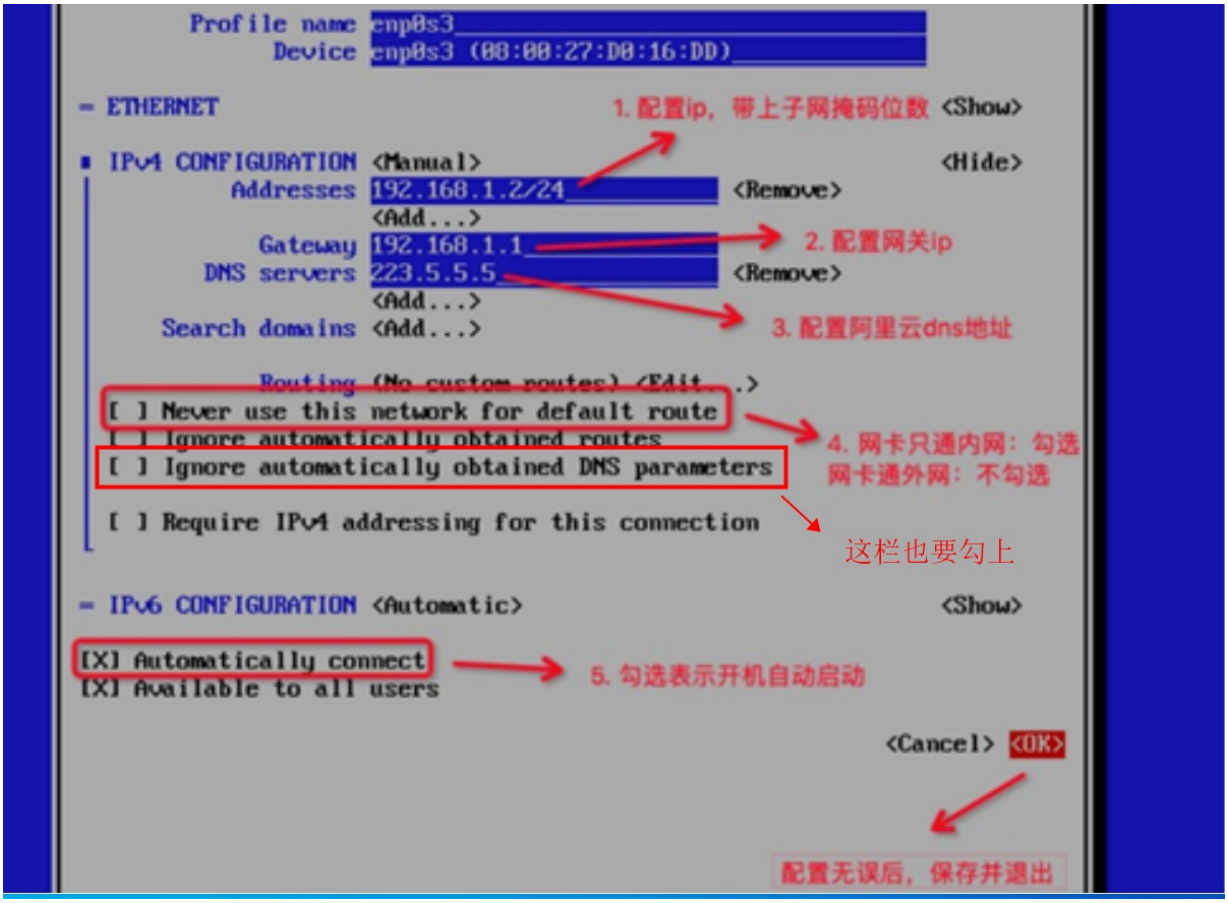
接着按左右键，将光标挪到show，展示配置信息：



附加情况

当该网卡需要访问外网时，不选中（没有X）

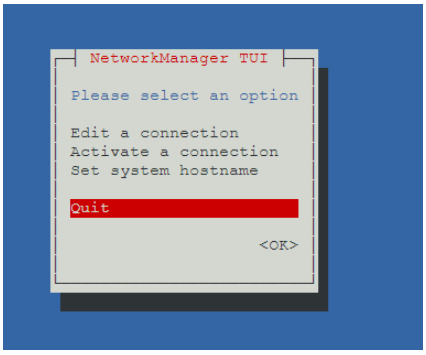
当该网卡只作为内部网络使用时，选中（有X）



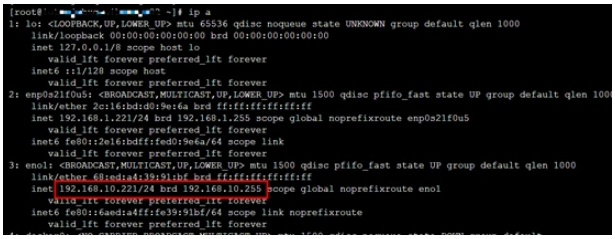
步骤5：重启网卡

返回到第一个页面，选中Quit退出，在执行重启网卡的命令：

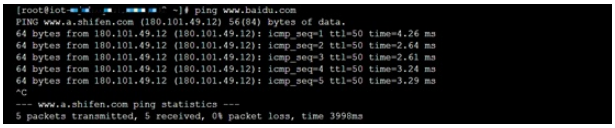
sudo ifdown 网卡名 && sudo ifup 网卡名



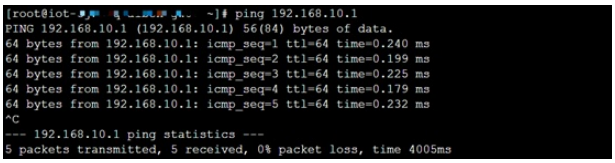
验证：回到linux界面下，执行ip a查看刚才配置得网卡IP信息是否正确修改（这里假设修改得是eno1得网卡IP,IP为192.168.10.221，如果没有发现修改成功，请返回上面得步骤查看是否有错误）



步骤6：验证是否通外网



通内网网关（网络连通性测试，具体网关以项目环境为准）

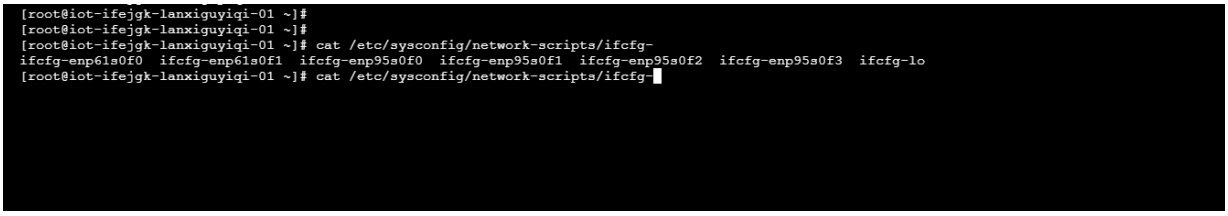


3.2通过系统文件配置网络

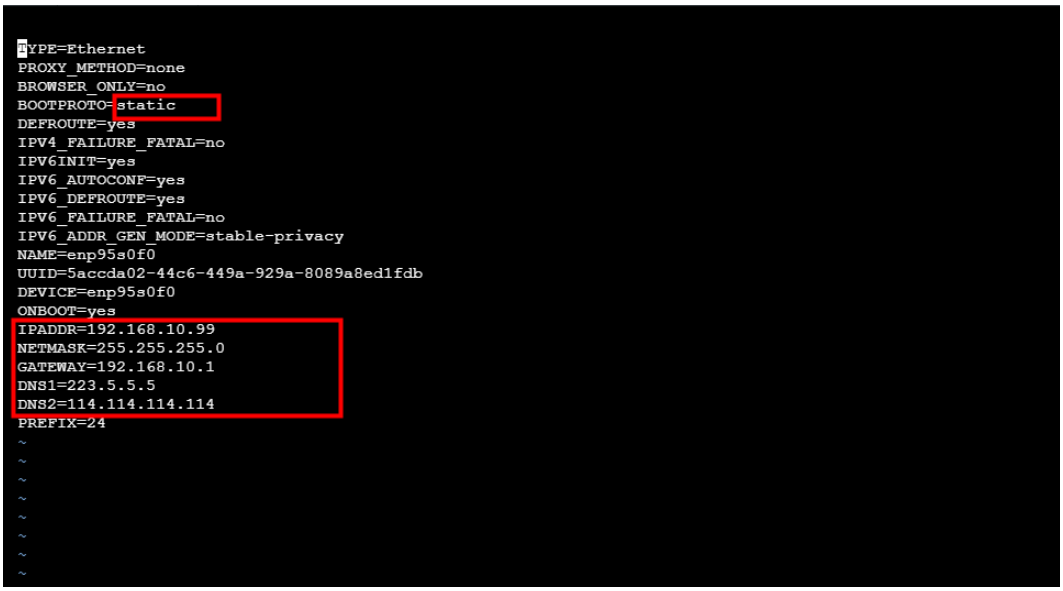
init 3是进入标准多用户模式，基本进入配置文件的地方都是一样的

查看配置：cat /etc/sysconfig/network-scripts/ifcfg-eno1

修改配置：vim /etc/sysconfig/network-scripts/ifcfg-eno1



按照图示的截图，将现场的网络IP配置在需要的网卡上，之后保存：wq!



配置保存完成后，重启网卡使之生效：

ifdown 网卡名 && ifup 网卡名

4.配置错误的例子

4.1如果没有将BOOTPROTO改成static,或者是没有改成Manual，或者是没有重启网卡灯原因，则会出现一个网卡配置两个ip的现象，这时候外网就是自动获取的ip地址，而不是手动配置的那个ip地址了

```
[root@iot-1azaji-haerbinrongchuangmao-node01 owner]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc mq state UP group default qlen 1000
    link/ether e8:61:00:00:b2:6b brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.15/24 brd 192.168.1.255 scope global noprefixroute dynamic eno1
       valid_lft 66273sec preferred_lft 66273sec
    inet 192.168.1.99/24 brd 192.168.1.255 scope global secondary eno1
       valid_lft forever preferred_lft forever
    inet6 fe80::7297:ab90:f948:8b62/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eno2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether e8:61:00:00:b2:6c brd ff:ff:ff:ff:ff:ff
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:33:40:b0:f6 brd ff:ff:ff:ff:ff:ff
    inet 169.254.123.1/24 brd 169.254.123.255 scope global docker0
```

这是自动获取的IP地址

4.2DNS配置错误

```
# Generated by parse-kickstart
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
BOOTPROTO="static"
DEVICE="eno1"
ONBOOT="yes"
UUID="f6c75240-c88d-45df-b3b9-bb171d1fdb17"
IPADDR=192.168.1.99
PREFIX=255.255.255.0
GATEWAY=192.168.1.1
DNS1=192.168.1.1
```

3.6.3.2. 边缘物业一体机激活失败

1. 问题现象

1.激活时，进度一直是0%，显示主机未联网，或者是等待主机上线



2.激活到34%，进度不动了，查看激活日志，显示"获取三元组成功"





3.激活到54%，进度不动了，查看激活日志，显示运行集群节点失败





2. 排查方法

1.一般都是第一种激活是0%，超过10分钟进度还是不动，说明激活失败了，登录到服务器，输入： sudo edgeboxctl -show serailNum 查看开头几行的SN是否和外壳上的SN一致，可以把型号和SN发给阿里同学，帮忙排查一下是不是SN的问题，如果SN没有问题，一般就是网络的原因了，这是就可以先检查主机ping外网通不通，DNS配置有没有问题，有没有防火墙等，远程连接服务器可以跑脚本看一下

curl -s http://edge-box-production-management.oss-cn-shanghai.aliyuncs.com/ota/diagnosis.sh | sudo bash -s -- -d -v -compatible

网络排查完之后，再执行清理命令，将刚才激活遗留的清除掉

第一步：sudo su 按下回车输入开机密码第二步：systemctl stop LinkIoTEdge第三步：rm -rf /linkedge/run 改成 mv /linkedge/run /linkedge/run-bak第四步：systemctl start LinkIoTEdge

```
    c) help        docuement: 'edgeboxctl -h'
[owner@iot-u2nmlj-qqy12645587 ~]$ sudo edgeboxctl -show serailNum
[sudo] owner 的密码：
{
  "model": "LE-A-B002",
  "serialNum": "3J2A5UUQBSWUSG0E",
  "version": {
    "linkedge": "v2.12.3",
    "linkedgeOTA": "v2.12.3"
  },
  "hardware": {
    "model": "LE-A-B002",
    "os": "linux",
```

清理完最后重启一下服务器，再进行激活，一般就没问题了

2.激活到34%，进度不动了，这种情况很少遇到，目前只遇到一台这样的，有某个配置文件有问题了，要找研发排查

3.激活到54%，进度不动了，查看激活日志，显示运行集群节点失败，这种情况偶尔会出现，一般等待几分钟，系统会自动重试，如果超过20分钟还是没反应，找阿里同学帮忙看一下

3.现场无法解决

如果现场无法排查到问题，或者是执行某命令一直报错，这就需要准备一台笔记本，SSH连接上服务器，执行一个脚本，然后阿里同学远程登录排查问题

3.6.4. 开放服务(API)常见问题

3.6.4.1. 开放API 调用和状态码

1. API 接口调用

阿里云IoT 提供的一套开放HTTP 服务，具体API 的调用请参加 API调用对接指引

2. HTTP 状态码

- HTTP 服务调用都有相对应的返回码，具体HTTP 返回码如下：

响应类型	HTTP状态码	场景
正常	200	后端服务顺利返回，没有出现异常，或者即使有异常，也符合预期。注意，业务失败不允许用200返回，请使用400错误码。
后端服务异常	500	**后端服务返回：Internal Error。**错误码没有映射到，返回默认错误码。
服务端无法完成请求	501	服务器不支持请求的功能，无法完成请求。
服务不可用	503	网关调用后端服务失败，后端服务不可用。
入参缺失	400	**必传参数缺失。**有些可选参数在特定业务场景下缺失。
入参非法	400	**参数格式不正确，比如：日期、数字、大小写不正确，长度问题等造成的。**特定业务参数校验失败，比如参数内容不符合接口要求、或者Json和RepeatList格式的入参，不满足接口要求。 * 增加URL、域名、字符串格式合法性校验。
账户验证失败	400	**账户欠费。**余额不足。 * 风控问题。业务失败的错误码统一使用400。
无权操作	403	**RAM校验不通过访问控制。**用户无权执行此操作。
越权操作	403	资源鉴权不通过（资源存在并归属于其他账号，不存在的参考资源不存在）。
SQL错误	403	**【强制】产品内部系统的SQL错误不应透出给用户，应使用 500 错误。**SQL语法错误。 * SQL超时错误。
流控	429	**系统流控：超过了系整体流控的阀值。**用户流控：单位时间内单个用户访问次数达到了上线。**业务规则流控：用户调用超过了业务方规则限制。**IP流控。
资源不存在	404	**获取单个资源信息时，若不存在。**删除时若资源已不存在提示404。如：InvalidInstanceId.NotFound

3.6.4.2. API 调用返回400 错误码

1. 问题背景

API 调用返回的状态码是400，说明HTTP 服务请求是异常的，一般返回400的原因如下：

- 使用的云产品出现欠费
- API 的入参不正确

2. 问题现象

- API 调用返回的错误信息：

```
response:
{
  "code": 400,
  "id": "1bd6b9522f024f669c1f49d3a92ce734",
  "localizedMsg": "请求错误",
  "message": "failed to build result",
  "returnType": "LUCENCY"
}
```

3. 排查方法

3.1 查看账号欠费信息

- 使用主账号登录阿里云页面，然后点击费用的tab
- 查看现在的现金余额，确认账号是否欠费
- 如果是欠费状态，需要先缴费；否则调用API会提示400



3.2 查看API 入参是否准确

- 现在遇到最常见的API 调用返回400 错误码是因为入参不准确，常见错误如下：
- 字段类型不匹配：比如Integer 字段，别复制String 类型
- API 版本：version 不正确
- 必须字段忘记设置或者设置为null

3.6.4.3. API调用返回403 错误码

1. 问题背景

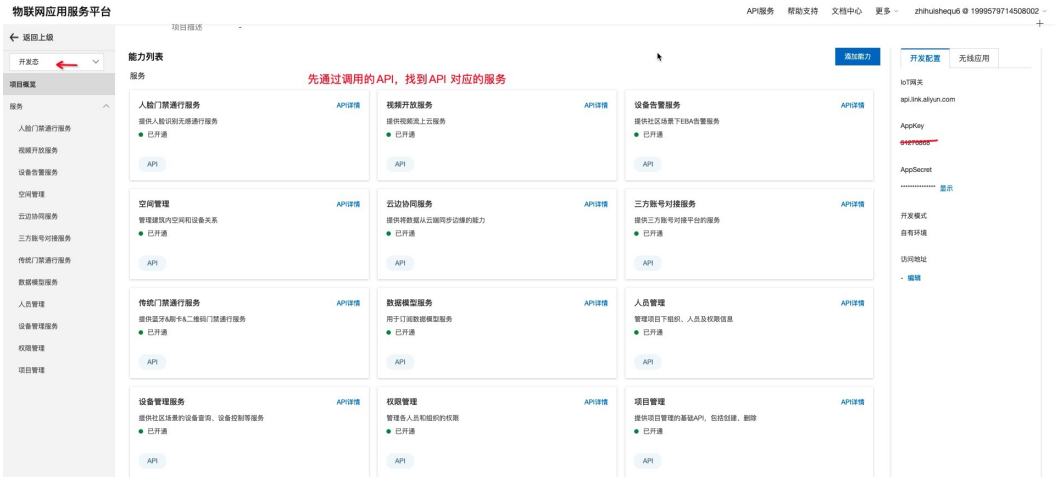
API 调用返回的状态码是403，说明调用API使用的Appsecret 没有被授权；需要先对appkey 进行授权

2. 问题现象

3. 排查方法

3.1 通过API 找到对应的服务

- 先登录物联网应用服务平台(主账号/子账号)均可
- 通过你调用的API 找到对应的服务：

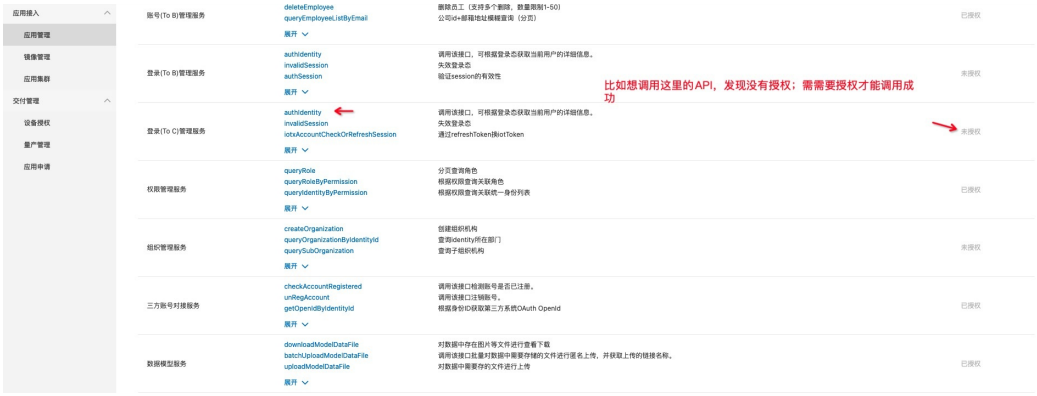


3.2 查看这个Appkey 是否对上面的服务进行授权

- 先登录AIoT 开放平台 然后找到Appkey 对应的应用，可以使用物联网服务平台项目来检索[应用的名称是：项目名称_ts_xxxx]



- 点击模型与权限
- 查看对应的API 是否开通对应服务的授权：



3.3 对包含API 的服务授权

- 如果应用没有发布，直接编辑模型与权限
- 如果应用已经发布，需要新建一个版本；然后编辑模型与权限



- 编辑模型与权限



- 开通之后会自动授权，无须依赖应用发布 【应用不发布，授权也会成功】

3.6.4.4. API调用返回429 错误码

1. 问题背景

- 云端开放的API 因为稳定性要求，会默认对API 接口做限流；
- 当API调用的QPS 达到或者超过限流阈值时，会报下面的错误

2. 问题现象

- API 调用返回429 的reponse

```
response: {
  "code": 429,
  "id": "280688",
  "localizedMsg": "太多请求",
  "message": "too many requests.",
  "returnType": "IoT_PROTOCOL"
}
```

3. 排查方法

31. API 错误信息

- API调用返回错误信息: "too many requests." 的错误信息就1s内说明请求太多，触发了限流机制；
- 触发限流机制的时候，这次请求就是失败；

3.2 修复方案

- 先排查下如此高并发的请求是否合理, 如果不合理可以优化、减小并发量

3.6.4.5. API 调用返回460 错误码

1. 问题背景

```
ontent: 2022/01/04-19: 25: 29INFO[
  hsfCallbackProcessExecutor-1-thread-70
]request-[
  44855
]traceId: 0bc525531641295529640137 9ed01a,
channel: serviceUnify,
appKey: 24875179,
tenantId: 3BB89AD985E84DC483BF64F2027EC76C,
domain: api.link.aliyun.com,
path: api.link.aliyun.com/account/checkOrRefreshSession,
method: /account/checkOrRefreshSession,
version: 1.0.4,
router: {
  "apiId": 1214,
  "apiVersion": "1.0.4",
  "hsfServiceId": 37
},
request: {
  "request": {
    "apiVer": "1.0.4",
    "language": "en-US",
    "appKey": "24875179"
  },
  "id": "11363",
  "params": {
    "request": {
      "identityId": "",
      "refreshToken": ""
    }
  },
  "version": "1.0.0"
},
response: {
  "code": 460,
  "id": "11363",
  "localizedMsg": "Request parameter error",
  "message": "identityid is blank!"
},
clientIp: 100.117.44.191,
hsfExpireTime: 9,
accountExpireTime: 0,
rbacExpireTime: 0,
routerExpireTime: 1,
authExpireTime: 0,
hsfInitExpireTime: 0,
expireTime: 12,
caRequestId: 68B25C56-F541-4CA6-8F8D-48C859F42927
```

2. 问题现象

```
response: {
  "code": 460,
  "id": "1c649f61-cc4a-4a90-b5d7-92f43feb3705",
  "localizedMsg": "请求参数错误",
  "message": "request parameter error."
}
```

3. 排查方法

3.6.4.6. API 调用返回500 错误码

1. 问题背景

- 返回500错误码, 说明API 后台依赖的HTTP 服务异常, 这种情况分为2类：
- API 后台是阿里一方的代码或应用： 如果一旦出现异常, 那就是所有客户、所有应用都异常
- 客户在边缘物业一体机上托管应用, 通过云边通道提供HTTP 服务；

2. 问题现象

- API 调用返回 500 错误码的reponse 信息

```
response: {
  "code": 500,
  "id": "33edc5e8-eba2-4cee-bd7f-eb009944a56a",
  "localizedMsg": "服务端错误",
  "message": "System Error",
  "returnType": "IoT_PROTOCOL"
}
```

3. 排查方法

3.1 先确定是否有托管应用提供服务

- 在智慧社区方案里面，应该车行场景使用的这种方案比较多
- 开发一个车行的应用托管到边缘物业一体机, 车行应用提供HTTP服务，这个HTTP服务通过云端API 来发起调用
- 这种方案： 客户或者API 使用者应该知晓，或者跟车行交付方来确认

3.2 托管应用日志查看

- 如果是边缘托管应用提供的HTTP服务，那先SSH 登录物业一体机查看应用的日志
- 日志里面找对应时间点的请求,查看错误日志；排查 返回500 的原因

3.6.4.7. API 调用返回20003 错误码

1. 问题背景

- API 在调用时, 需要指定域名(domain)、方法(method)、API 版本(version)
- API 网关也是通过上面3个参数来准确定义和完成请求路由到指定的后台服务
- 如果上面的三个三有一个不正确, API 网关在路由配置里面并没有找到对应的后台服务,就会报"没有找到路由服务" 的错误

2. 问题现象

- API 调用返回的错误码如下:

```
response: {
  "code": 20003,
  "id": "8872",
  "localizedMsg": "没有找到路由服务",
  "message": "gateway.not found backend router"
}
```

3. 排查方法

- 检查参数是否正确，需要检查的参数apiVer，domain，method
- 具体可以参考API 调用的官方文档

3.6.4.8. API调用返回20004 错误码

1. 问题背景

- API 调用可能会报“调用后端服务时发生异常” 的错误； 这类错误可以通过表面意思来解读； 但除了API 依赖后台服务真出现异常, 还包括了一种常见的错误，就是入参类型不匹配 (cause: argument type mismatch)

2. 问题现象

```
response: {
  "code": 20004,
  "id": "346ff575d1bc465f92925047009da1a9",
  "localizedMsg": "调用后端服务时发生异常",
  "message": "gateway.backend service error"
}
```

3. 排查方法

- 出现20004 错误码的2种可能性:
- 后台的服务真出异常了, 后台服务返回的reponse 确实是异常
- API 调用的参数不匹配，下面重点说明这类问题的排查
- 遇到20004 的错误码， 根据API 网关返回的入参列表, 结合API 官方文档仔细检查入参的类型:

- 比如下面是入参JSON

```
"params": {
  "iotId": 452878274,
  "checkTemperature": true,
  "minThreshold": 1,
  "maxThreshold": 1,
  "productKey": "a1ZPPPPKKKK",
  "deviceName": "CDSDGY_xx_xxxx"
}
```

- 跟API 官方文档进行参数类型匹配

3.6.5. 人行场景常见问题

3.6.5.1. 人脸权限下发状态(NeedManual)

1. 问题背景

人脸下发是通过API 进行权限数据下发，下发之后通过API 来查询状态； 有时会提示状态是 NeedManual , 这时需要人工介入排查； 错误现象:

```
{
  "deviceIotId": "xxxxxxx",
  "faceId": "aaaaabbbbb",
  "gmtCreate": 1637549159000,
  "gmtModify": 1637549171000,
  "groupId": "iE8Bp6GmlpmMBOq9000000",
  "id": 78901822,
  "isDelete": "n",
  "status": "needManual",
  "tenantId": "TTTTTTTT",
  "transferCount": 0
}
```

1.1 人脸权限下发异步方式

- 人脸权限下发通过API （/face/paas/permission/add）下发, 下发过程中会重试三次, 但下发的状态不会在请求reponse 里面同步返回;
- 需要通过权限查询API (/face/paas/permission/querybyuser) 来查询下发用户的状态信息;

1.2 查询人脸权限下发状态

- 查询人员权限：开发API 服务
- 查询人脸权限下发状态

1.3 人脸权限状态:

- bizType = FACE_SYNC_STATUS
- 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

1.4 needManual 状态

- 人脸权限下发会把人脸的图片/信息下发到边缘端(边缘端进行人脸对比) 边缘端会存储图片信息到数据库，一旦存储到数据库失败，那人脸下发的状态 status 就会设置为 needManual
- needManual 状态： 字面意思就是需要人工介入

2. 问题现象

- 使用人员信息查询人脸权限下发状态

```
{
  "code": 200,
  "data": {
    "deviceListTotal": 1,
    "deviceListPageNo": 1,
    "deviceListPageSize": 20,
    "deviceList": [
      {
        "syncTime": "2021-12-08 15:24:35",
        "iotId": "SxbdAAAAABBBBBCCCCT000",
        "deviceNickName": "门口机_人脸识别",
        "status": "needManual"
      }
    ],
    "userType": "OPEN",
    "userName": "test",
    "userId": "1000018",
    "extInfo": "{"userName":"test","policy":"PERMISSION"}"
  }
}
```


3. 排查方法

3.1 判断网络状态

- 远程SSH 物业一体机, 然后通过网络监测脚本(curl -s http://edge-box-production-management.oss-cn-shanghai.aliyuncs.com/ota/diagnosis.sh | sudo bash -s -- -d -compatible)
- 如果网络监测发现有ERROR，要先修复网络问题
- 修复网络问题后, 需要先把状态(NeedManual) reset
- 重新下发权限

3.2 强行reset 状态

- 如果网络监测正常, 只能reset 状态，然后重新下发
- 如果下发再次失败, 采取工单人工协助解决

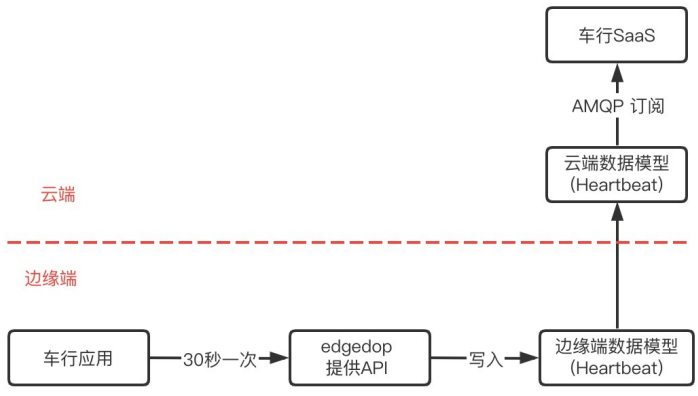
3.6.6. 车行场景常见问题

3.6.6.1. 车场心跳数据缺失

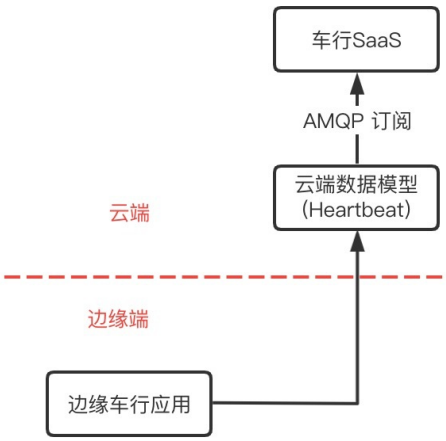
1. 问题背景

- 车场心跳数据的原理：
- 未来保障车场稳定运行, 在车场定时(每30秒一次)上报心跳功能
- saas 端订阅到实时心跳数据,可以简单判断车场业务正常；反之判断为车场业务出现异常
- 车场心跳数据在项目里面有2种方案：
- 边缘车行应用每30秒写边缘数据模型库, 实时同步到云端
- 边缘车行应用每30秒写云端的输数据模型库

1.1 方案一: 边缘写数据实时同步到云端



1.2 方案二： 边缘应用直接写云端



1.3 简单判断方案的方法

上面2种方案在项目中都有使用，因为边缘端数据模型(Heartbeat)是通过官方云边协同组件实时上传到云端的数据模型(Heartbeat)，所以方案一必须依赖云边协同组件；

简单判断是方案一还是方案二的方法：

- 如果安装部署了云边协同组件，那就是方案一
- 如果未安装部署云边协同组件，那就是方案二

2. 问题现象

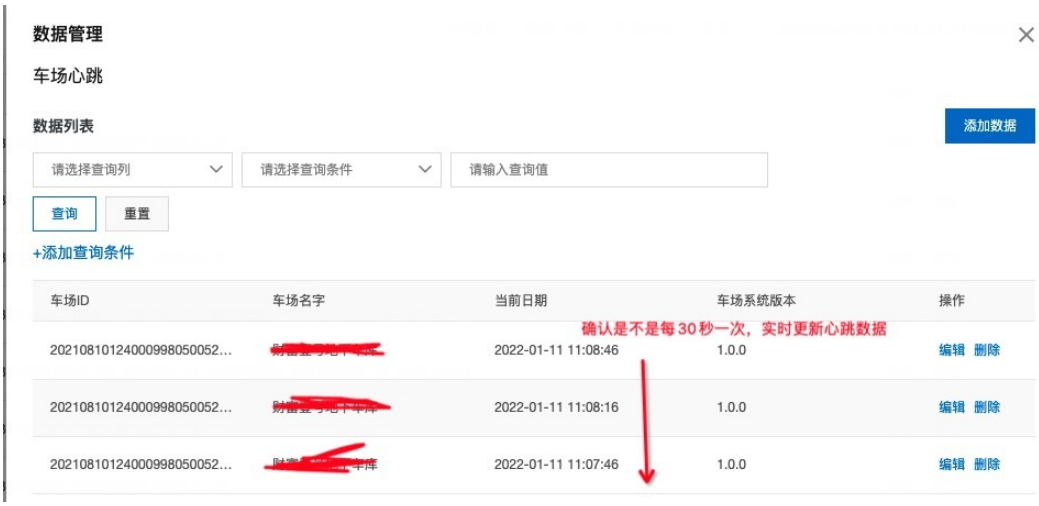
- 车行saas 订阅不到每30秒一次的心跳数据，车行saas 会报警

3. 排查方案

3.1 方案一 车场心跳缺失

3.1.1 查看iot中台数据模型是否有心跳数据

- SSH 物联网应用服务平台，在数据集成里面找到车场心跳(Heartbeat) 的模型表，点击管理查看最新的数据：
- 判断是否有最新的数据在每30秒成功上报一次



- 如果车场心跳每30秒上报心跳数据成功，那局势AMQP 订阅不成功或者车行saas 端问题
- 这时需要先排查车行saas 日志，查看AMQP 订阅是否异常

3.1.2 iot 中台没有实时更新

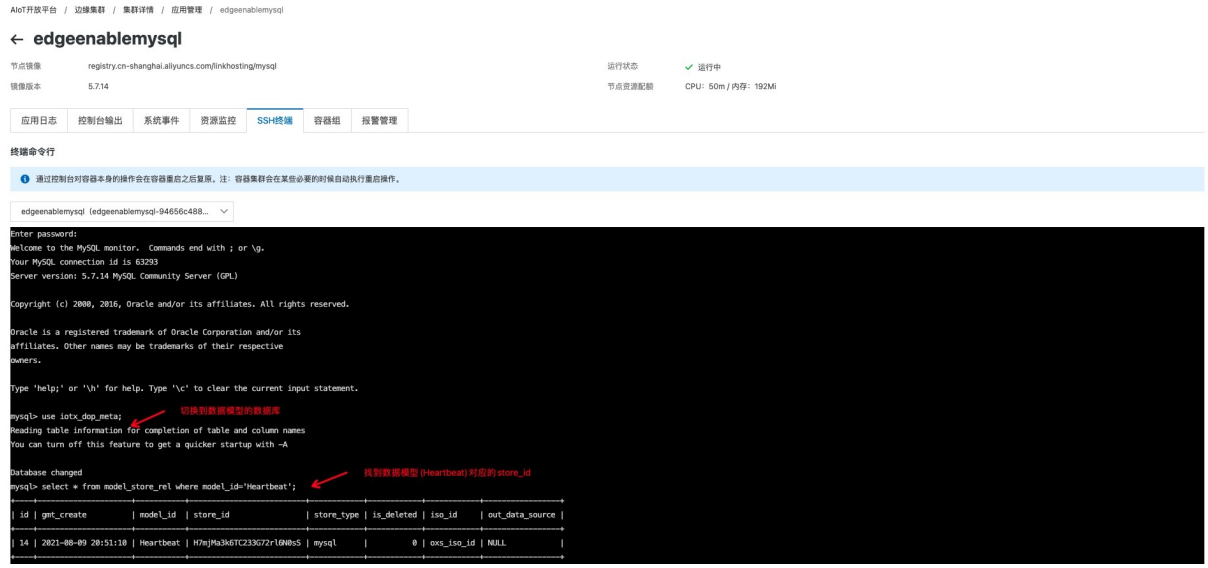
- 如果物联网应用服务平台数据集成里面的心跳数据没有，那最大可能是边缘端没有实时写入数据
- 1). 登录物联网应用平台，应用管理里面找到云边协同组件，找到enablemysql 节点，点击节点运维：



- 2). 登录enablemysql 节点 SSH 客户端，并连接mysql:

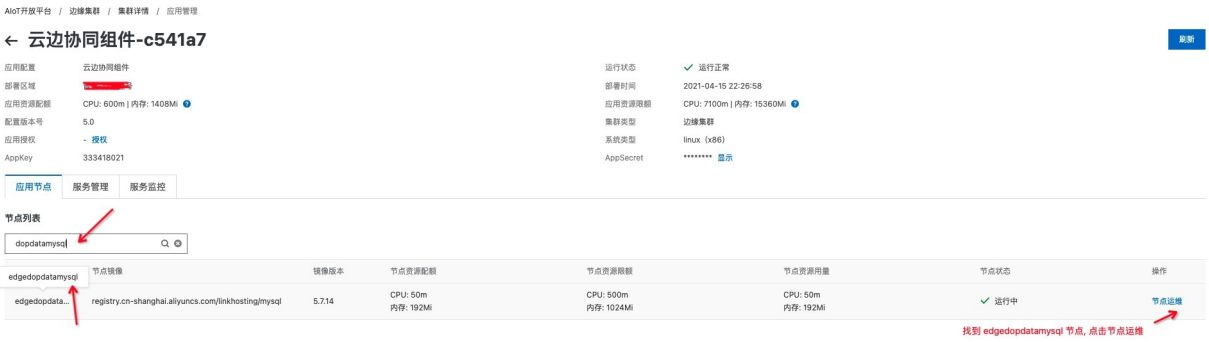
```
use iotx_dop_meta

select * from model_store_rel where model_id='Heartbeat';
```



- 车场心跳数据模型对应的 store_id: H7mjMa3k6TC233G72rl6N0sS

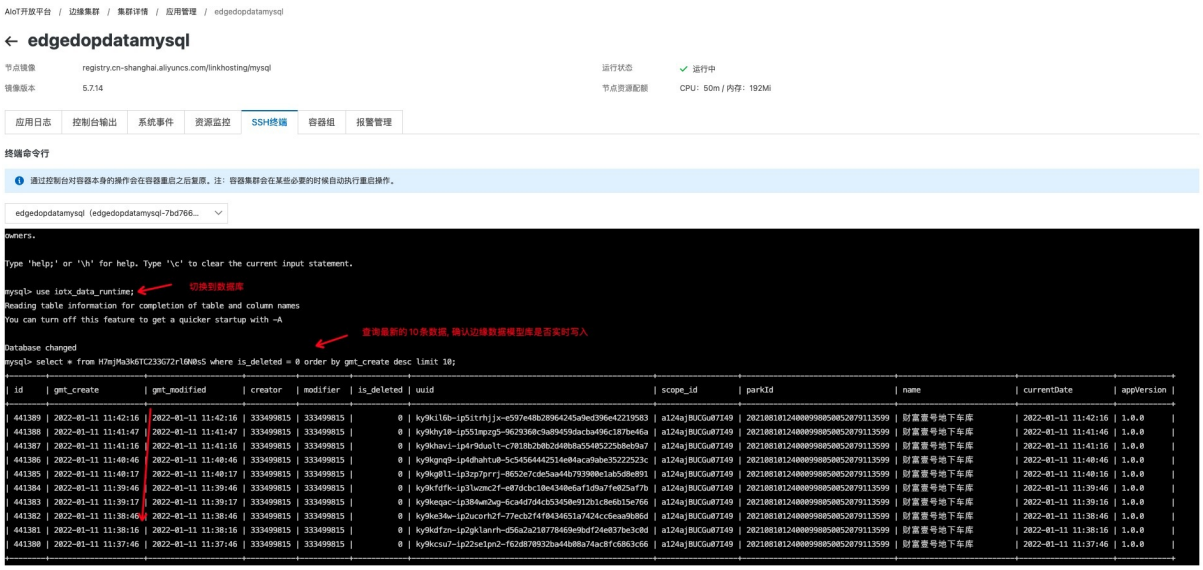
- 3). 找到edgedopdatamysql 节点，点击节点运维:



- 4). 进入SSH 客户端，登录mysql 数据库，查看车场心跳数据:

```
use iotx_data_runtime;

select * from H7mjMa3k6TC233G72rl6N0sS where is_deleted = 0 order by gmt_create desc limit 10;
```



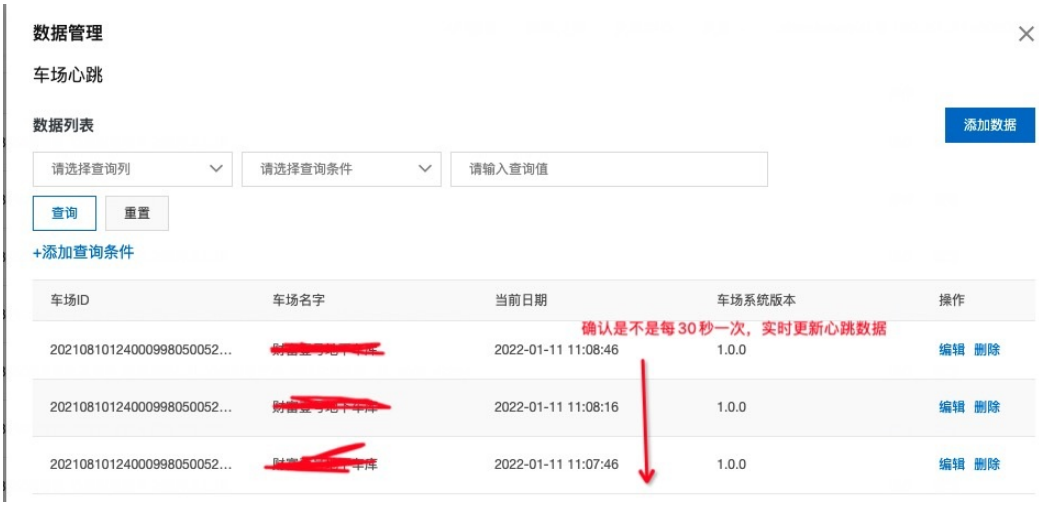
- 5). 如果边缘端数据模型表里面也没有实时写入心跳数据，此时就要查询边缘车行应用的日志,确认边缘车行应用调用API 写入边缘数据模型表的日志，是否有异常

3.2 方案二 车场心跳缺失

- 如果是边缘车行应用直接调用API 写云端的数据模型表，先要判断iot中台是否有心跳数据

3.2.1 查看iot中台数据模型是否有心跳数据

- SSH 物联网应用服务平台，在数据集成里面找到车场心跳(Heartbeat) 的模型表，点击管理查看最新的数据：
- 判断是否有最新的数据在每30秒成功上报一次



- 如果车场心跳每30秒上报心跳数据成功，那局势AMQP 订阅不成功或者车行saas 端问题
- 这时需要先排查车行saas 日志，查看AMQP 订阅是否异常

3.2.2 查看边缘端车行应用的日志

- 因为方案二是边缘车行应用直接写云端数据模型表，如果云端的车场心跳没有实时写入成功；
- 直接查看边缘车行应用调用API 写数据的日志，先找到异常日志

3.7. 发布公告

3.7.1. 2022-01-27

数据模型存储管控通知

平台将会对数据模型功能进行管控，将于2022-03-01开始清理30天前数据，并且对新写入数据的存储周期更改为30天，30天后数据将会被清理，请及时消费数据进行存储。

提供Modbus、BacNet协议点位配置工具

提供Modbus、BacNet协议的点位配置工作台，同时支持将协议点位数据与平台物模型进行映射，快速实现BA系统的数据集成。

发布门禁一体机官方协议，优化门禁的集成效率

平台之前提供的集成工具，是以门禁功能维度提供的，导致添加一个物理的门禁设备，需要分为多个逻辑设备，这对用户非常不友好，本次发布门禁一体机官方协议，用户只需要感知要集成的门禁品牌和型号即可，后台会自动将相关能力关联好，这大大提高了门禁设备的集成效率。

支持人行版数据冷备能力

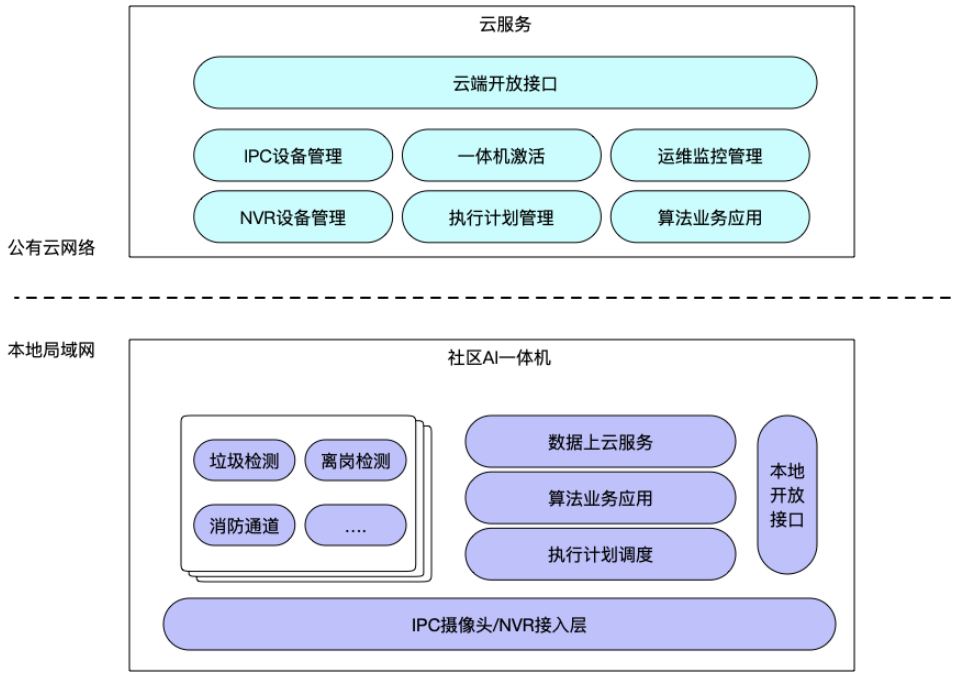
一体机运行阶段，可能会出现各种异常情况，导致一体机无法正常运行，这时就需要有冷备的高可靠方案，将一体机数据迅速迁移到另一台冷备机，实现业务的迅速恢复。

4.边缘智能一体机

4.1. 什么是边缘智能一体机（执行计划版本）

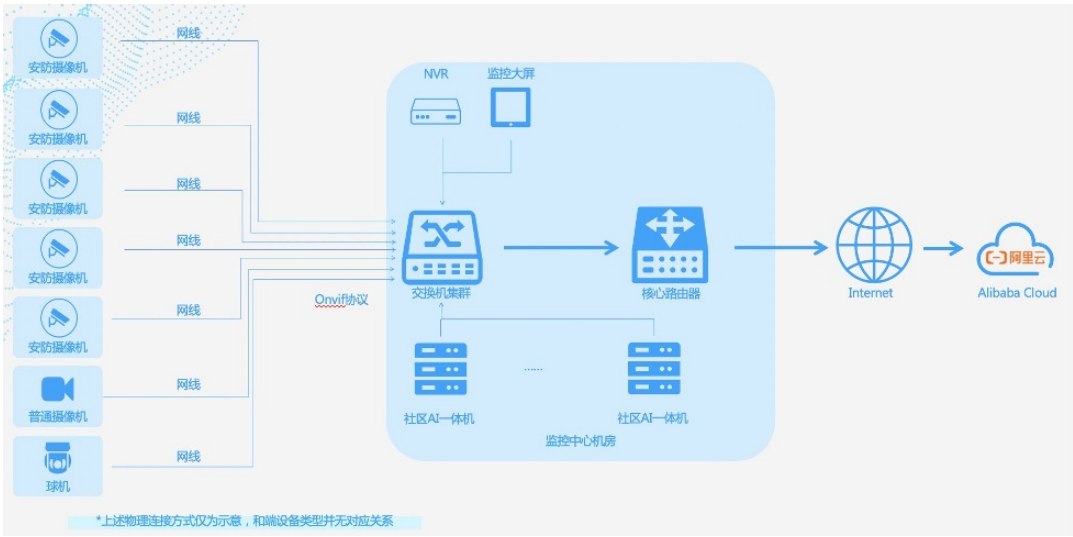
边缘智能一体机（执行计划）基于业内领先的视觉算法及调度策略，面向社区、园区等物业管理场景，通过AIOT数字化和智能化，提升物业管理效率，提升发现风险时效。通过AI本地算法能力提升老旧摄像头升级，为线下社区或园区安防场景、品控场景提供智能化AI分析，同时通过多维度调度策略提升单体硬件的视频接入量，从而降低社区/园区的数字化成本，以AI事件为服务驱动，助力客户实现数字化业务提效。

- 系统架构



边缘智能一体机（执行任务版）采用云边协同架构，包含AIOT边缘计算、视频管理、AI算法等多种优质服务。基础设施包括部署在社区或园区的高清摄像头网络和本地AI BOX服务器以及阿里云基础计算资源。执行计划对摄像头进行轮循算法识别，可按时间周期及识别频率进行自定义计划；为了降低算法误报或场景诉求，执行任务还支持对事件进行多次多间隔的二次确认能力；为了保证一体机在边缘稳定运行，边缘一体机采用容器化管理，并对运行的计划任务进行实时监测，当资源成为瓶颈时，动态调整事件优先级。云端和边缘两侧的标准开放接口，以便满足不同场景的应用进行集成，开放能力包含视频实时视频流、历史视频流、AI事件订阅多种服务。

- 部署架构



- 产品功能

功能分类	功能名称	功能描述

算法服务	边缘算法	支持消防通道占用、重点区域占用、摄像头遮挡、离岗检测、人群聚集、区域入侵、垃圾检测7类非实时算法的推理；
计划管理	周期可视化	支持执行计划按日周期化运行，同时提供运行周期可视化，直观表达各任务执行周期和频率；
	风险时间段	系统通过配置的执行任务数量、周期、频率计算一体机可能出现的计算瓶颈，用户可基于提示进行调整，以保证一体机的有效运行；
	错峰启动	一体机会根据任务数量和启动时间，在边缘正式执行时，进行错峰算法分配，保证硬件资源得到最大的利用；
	二次确认	支持对AI事件进行按频率按次数的二次确认能力，降低AI算法带来的误报和错报问题；
	升降级能力	支持一体机识别到现场AI事件超出硬件可处理的拼接时，一体机会根据用户配置的升降次策略进行处理，降低硬件在高压力情况下出现宕机的风险；
	任务感知	支持在云端对边缘一体机中的任务进行远程开关，同时可通过执行记录对边缘运行的任务情况进行实时掌握；
开放能力	云端开放	支持云端系统和AIOT平台进行云上集成，一体机在云端开放视频拉流、AI事件数据接口；
	边缘开放	支持本地系统和一体机进行本地集成，一体机在本地开放API服务，包含设备拓扑关系、视频拉流、抓图、接收事件结果等能力；
远程运维	运维大盘	提供边缘计算产品运维监控大盘，大盘数据包含主机资源top5排名，按时间周期统计告警数量，告警终端设备排名，可视化设备分布，设备在线数统计等指标
	告警规则	支持对VPU使用率/GPU使用率/CPU使用率/CPU温度/VPU温度/GPU温度/磁盘使用率/内存使用率多个维度的告警警戒线设置，告警数据可通过钉钉机器人实时推送至钉钉群
	运维告警	基于告警规则触发的所有异常告警事件进行分类查询及处置，并分享远程运维终端给到排查人员进行异常排查
	远程终端	可通过远程运维控制进入边缘计算产品控制台，进行黑屏化系统级运维

- 产品限制

接入协议	支持ONVIF协议的IPC
接入数量	最多支持256路摄像头
任务数量	最多支持32个轮循任务
单任务运行摄像头数量	最多支持256个摄像头
检测频率	>=1分钟

算法并发数	4TPS（峰值支持8TPS，峰值不易长时间运行，影响硬件生命周期）

4.2. 激活一体机

通用安全注意事项

安装所有型号的边缘一体机前，请先阅读并遵循通用的安全注意事项。

- 负责安装、操作、维护边缘一体机的人员，必须了解各种安全注意事项，掌握正确的安装操作方法之后，方可安装、操作和维护设备。
- 为保障人身和设备安全，在安装、操作和维护边缘一体机时，请根据随机附带的安装说明书安装。
- 接触任何导体表面或接线端子之前应使用电压表测量接触点的电压，确认无电击危险。
- 安装完边缘一体机后，应清除设备区域的空包装材料，如纸箱、泡沫、塑料、扎线带等。
- 边缘一体机应在符合安装环境要求的环境下使用，否则可能造成设备故障，由此引发的设备功能异常或部件损坏、人身安全事故、财产损失等不在设备质量保证范围之内。

安装激活环境要求

安装和使用边缘一体机时，需满足一定的安装环境要求。本文介绍安装与激活边缘一体机时的基本环境要求。

> **说明**
> 每个型号边缘一体机的详细安装要求不同。具体内容，请参见随机附带的安装说明书。

参数项	参数说明
机架	可选导轨安装和托板安装两种方式，可根据现场环境自由选择。 导轨安装和托板安装时的详细要求，请参见随机附带的安装说明书。
环境温湿度	需要明确现场是户外还是室内后，保证温湿度满足设计参数，参考设备使用手册。
电源	使用直流电压，需要的电源功率相对较小，多数情况都可以满足。直流电压范围和电源功率，请参见随机附带的硬件说明书。
网络	支持有线网络，部分机型支持蜂窝网络。 边缘一体机具体支持的网络，请扫描设备机身的设备信息二维码获取。
云上服务	激活边缘一体机时，需要开通云上服务。 **说明** 云上服务仅需开通一次，则长期有效。 1. 注册阿里云账号，详情操作，请参见阿里云账号注册流程。 2. 开通物联网边缘计算。 （可选）使用物联网边缘计算的视频能力时，需要开通视频边缘智能服务。

边缘一体机安装与激活

物联网边缘计算有两种类型多个型号的边缘一体机，每个型号的一体机的安装方式有所不同，但激活方式都一样。本章节主要说明边缘一体机的安装方式、互联网连接配置和激活方式。

安装方式

B类型边缘一体机根据型号不同，有如下两种安装方式：

- 机架式托盘安装，部分型号边缘一体机允许在机柜中使用托板安装。
- 卡轨式安装，部分型号边缘一体机有卡轨卡扣，支持将边缘一体机安装在机柜卡轨中。

互联网连接配置

边缘一体机总体上有如下三种连网方式。

- DHCP上网，当本地网络支持动态主机配置协议DHCP（Dynamic Host Configuration Protocol）时，可以使用网线直连边缘一体机的DHCP网口，进行互联网连接配置。具体网口说明请参见随机附带的接口说明文档。
- 静态IP上网，静态IP上网方式，需要登录物联网边缘计算管理平台进行网络配置。
  - 登录物联网边缘计算管理平台。
  - 进行网络配置。

- 蜂窝网络上网

> 说明
>
> 请根据随机附带的接口说明文档，将SIM网卡正确安装到边缘一体机的网卡卡槽。安装（或移除）SIM卡后，需要进行断电和商店操作，重启边缘一体机后SIM卡生效。

激活边缘一体机

完成网络配置后，可激活边缘一体机。您可以直接扫描边缘一体机上的二维码，或者登录边缘计算控制台，根据提示信息，激活边缘一体机。

1. 扫描二维码激活
   i. 找到张贴在边缘一体机上的硬件标签，扫描二维码，激活边缘一体机。以LE-V-B015型号边缘一体机为例，硬件标签张贴在一体机底部。



   ii. 使用阿里云账号，登录激活页面，**此处登录的阿里云账号，将作为边缘一体机的归属账号，因此请确保使用您的正式生产账号**。
   iii. 安装电源线，确认电源灯常亮。



iv. 插入网线。



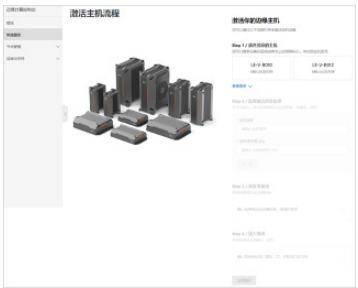
v. 确认边缘一体机配置信息后，单击立即激活。您可以修改主机名称，支持数字、小写英文字母、短划线（-），长度限制为4~30个字符，不能以短划线（-）开头或结尾。



2. 在边缘计算控制台激活

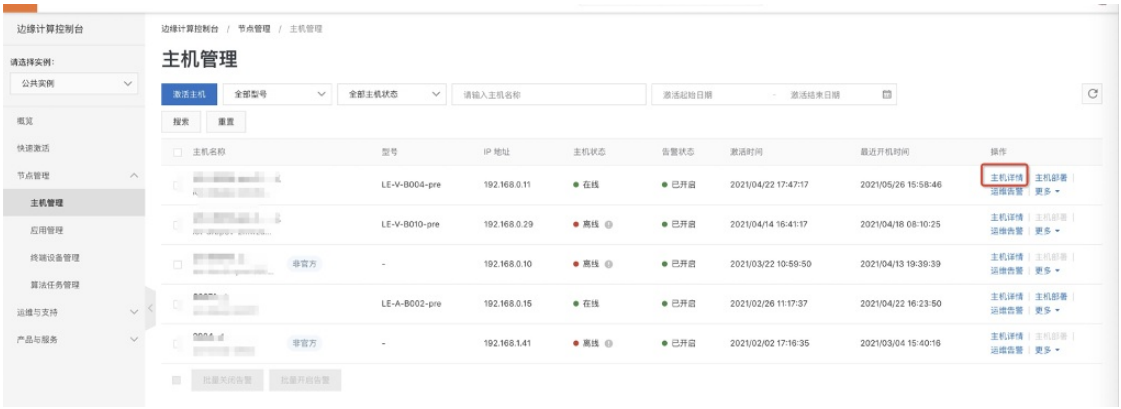
i. 使用阿里云账号，登录边缘计算控制台。

ii. 在左侧导航栏单击快速激活，**首次在您的阿里云账号下激活边缘一体机时，登录控制台后，直接显示激活主机流程页面。**

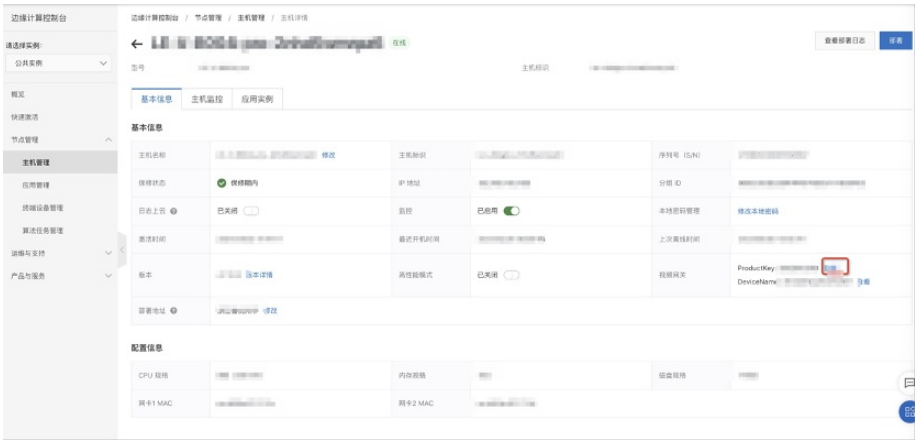


iii. 根据提示信息，选择您要激活的边缘一体机型号、配置激活信息、插入电源线和网线后，单击立即激活。

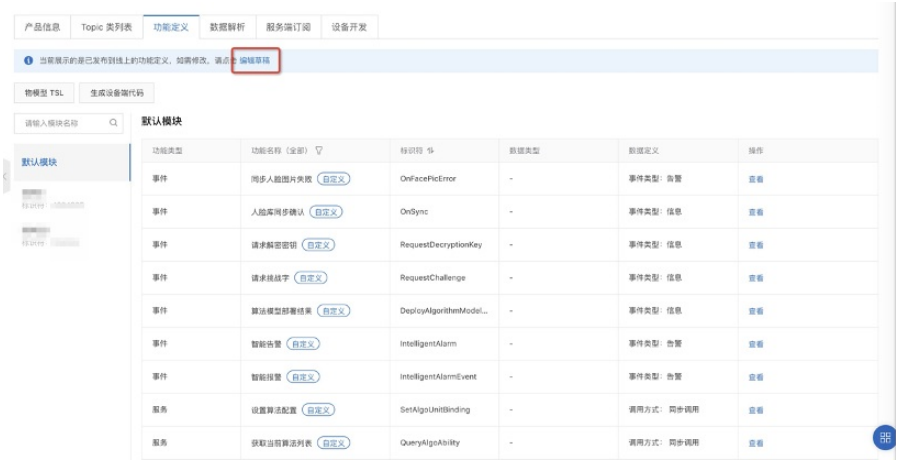
3. 升级一体机物模型，一体机激活后默认使用通用的物模型，需要手工更新下物模型才能使用执行任务相关功能
   i. 一体机激活后请进入“主机管理”中找到激活的设备，并点击“主机详情”。



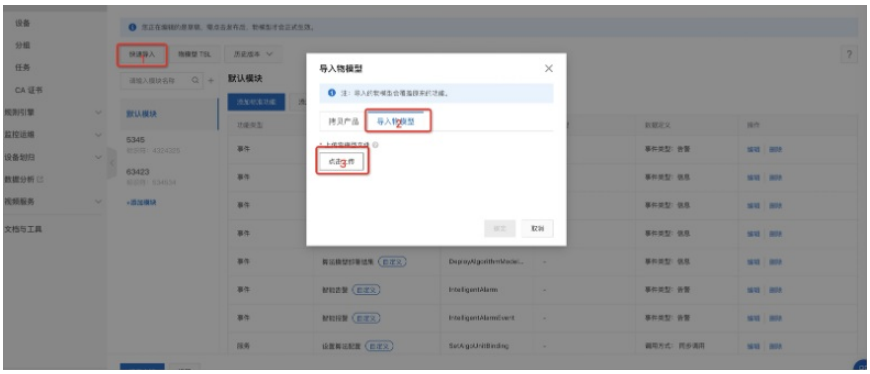
   ii. 进入“主机详情”->“基本信息”并跳转一体机产品管理页。



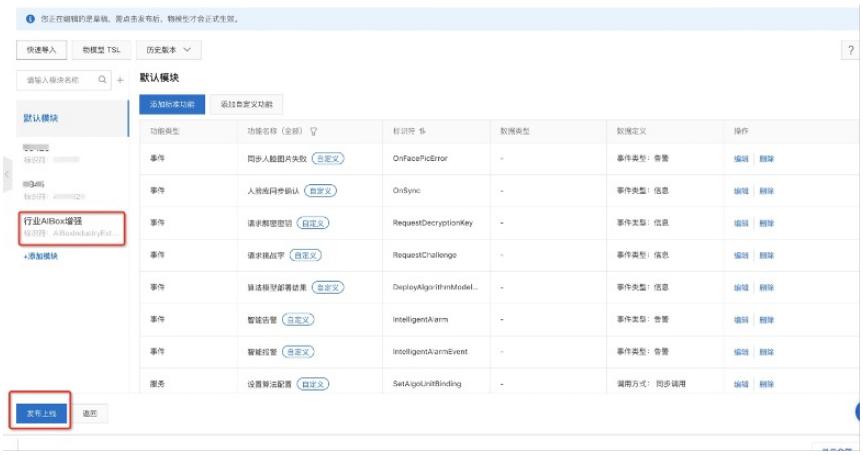
   iii. 进入产品详情>功能定义>编辑草稿进行自定义功能添加。



iv. 上传产品配置，快速导入>导入物模型>点击上传页面中，上传模型zip包，标准zip包：见附件《AI一体机物模型.zip》



v. 将物模型发布上线



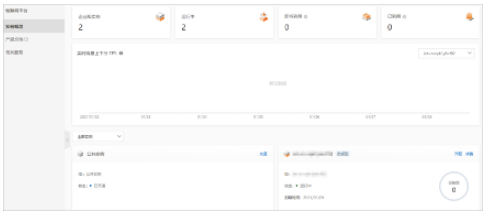
4.3. 添加摄像头

视频设备支持以标准物模型为载体的消息流转，为了使用标准物模型，需要为视频设备创建产品和设备。

1. 创建摄像头产品

   i. 登录物联网平台控制台，登录物联网平台的阿里云账号，需和云端调用接口的账号一致。否则会出现无法订阅云端报警消息的情况。

   ii. 在实例概览页面，找到对应的实例，单击实例进入实例详情页面。物联网平台实例相关内容，请参见实例管理。



iii. 在左侧导航栏选择设备管理 > 产品，单击创建产品，创建视频设备所属的产品。详细操作，请参见创建产品。

a. 创建网络摄像头IPC设备所属产品时：

b. 所属品类需要选择智能生活 > 家居安防 > 摄像头，您也可以在选择品类面板的搜索框中输入摄像头，搜索该品类并选中。

c. 节点类型需要选择直连设备。



iv. 产品创建成功后，可以根据实际情况通过添加IPC设备，具体配置请参照对应章节。

2. 添加IPC设备流程，边缘一体机支持接入常见协议的网络摄像机（IP Camera，简称IPC），本章主要介绍IPC设备的接入方法。

i. 登录边缘计算控制台。

ii. 在左侧导航栏选择节点管理 > 终端设备管理 。

iii. 在终端设备管理页面，选择您的主机名称 > IPC 。

iv. 在设备列表区域，单击添加设备。



v. 根据系统提示设置参数，然后单击确定。

参数	描述
设备添加方式	支持新建设备，也可以选择已有设备。
产品名称	选择已在前提条件中创建的产品。
DeviceName	设备名称： ■ 设备添加方式为设备新建时，此处需要设置设备名称，同一个边缘一体机下唯一。支持英文字母、数字、短划线（-）、下划线（_）、at（@）、英文句号（.）和英文冒号（:），长度限制为4~32字符。 ■ 设备添加方式为已有设备添加时，此处选择产品下已有的设备。
设备别名	设备别名，同一个边缘一体机下唯一。支持中文、英文字母、数字、短划线（-）、下划线（_）、at（@）、英文句号（.）和英文冒号（:），长度限制为2~32字符。
协议类型	设备的协议类型。执行任务版本一体机仅仅支持ONVIF协议的IPC ■ ONVIF：开放式网络视频接口论坛（Open Network Video Interface Forum）制定的协议，是网络视频设备之间的信息交换定义通用协议。
IP地址	设置设备的IP地址和端口。 **说明** IP地址:端口的组合，在同一边缘一体机下需唯一。
端口	
用户名	设置设备用户名，用户访问设备。支持英文字母、数字、下划线（_）、短划线（-）、英文冒号（:）、at（@）、英文句号（.），长度限制为4~32字符。
密码	访问设备时使用的用户名密码。支持英文字母、数字、下划线（_）、短划线（-）、英文冒号（:）、at（@）、英文句号（.），长度限制为2~15字符。
确认密码	再次输入密码。
备注	可输入文字，用来描述设备信息。字数限制为100个字符，可以为空。

vi. 设备添加完成后，通过单击设备对应操作栏中的功能按钮，管理IPC设备。



功能按钮	说明
设备详情	查看设备的详细信息和视频信息。
设备配置	修改当前设备的配置。详细说明，请参见本文上方参数说明。
离线诊断	设备状态为离线时出现的功能按钮。 单击离线诊断后，系统为您进行故障诊断，您可以根据诊断结果，处理设备离线问题。
运维告警	单击运维告警，系统跳转到该设备的运维告警页面，您可以查看运维告警信息。更多信息，请参见运维告警。
系统事件	单击系统事件，系统跳转到该设备的系统事件页面，您可以查看系统事件信息。更多信息，请参见系统事件。
更多设备信息	单击更多设备信息，系统跳转到物联网平台控制台上的设备详情页面，您可以查看该设备在云端的详细信息。更多说明，请参见查看具体设备信息。
申请服务支持	描述设备问题、填写联系方式后，可申请售后服务与支持。更多信息，请参见售后服务。
移除设备	可从边缘一体机中移除该设备。 **注意** 从边缘一体机中移除设备，可能会影响您正在运行的边缘业务，请谨慎操作。

4.4. 配置执行任务

边缘一体机支持对接入的摄像头进行执行计划的编排，以便满足不同摄像头在特定周期及间隔中。

1. 登录边缘计算控制台。
2. 在左侧导航栏选择产品与服务 > 增值服务。
3. 在增值服务页面，单击执行计划服务下的管理服务。
4. 若您还未开通执行计划服务，请按如下步骤开通服务：

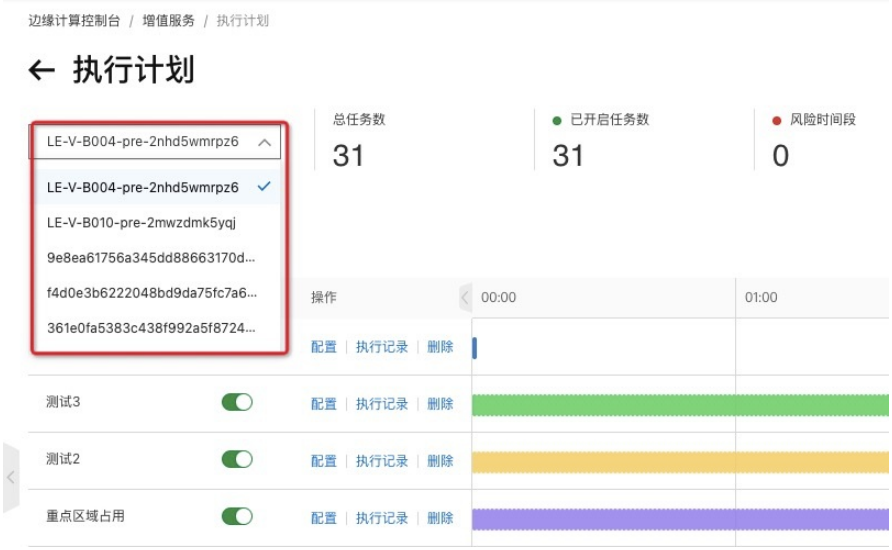


i. 单击执行计划服务下的前往开通。

ii. 在服务详情页面查看服务介绍，然后单击立即开通。

iii. 在开通服务对话框，阅读并选中我已阅读并同意，单击确认开通，开通执行计划服务。

5. 在执行计划页面左上角，可以根据名称切换一体机，从而完成在不同一体机下的配置工作。



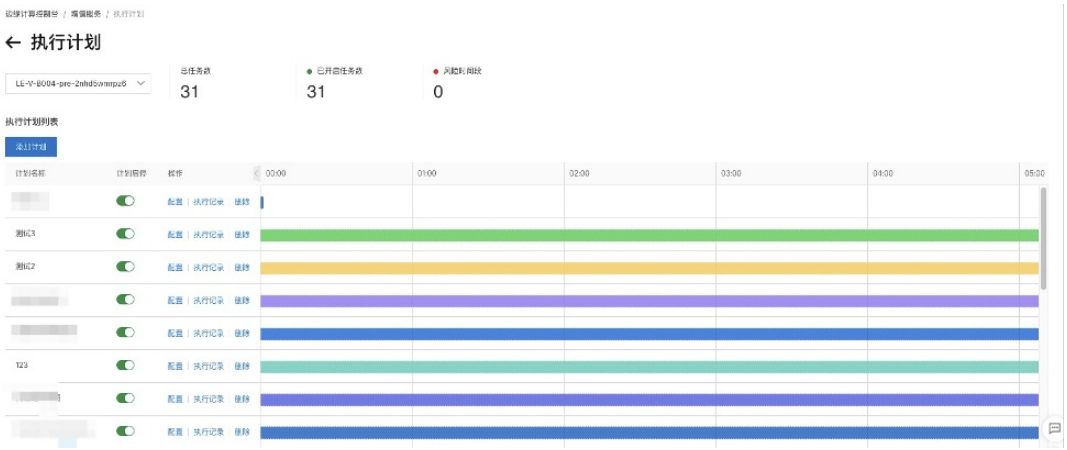
6. 在左侧执行计划列表中点击添加计划，并在右侧弹出的添加计划面板中，对计划进行配置。



参数	描述
计划名称	为当前运行的计划的唯一名称，这个名称设置后不允许做修改。
时间范围	计划允许被执行的时间范围，可以被指定的时间包含小时和分钟，所有计划都是每日执行，所以日期不开放配置。
执行算法	为让当前计划选择执行的算法，由于不同的算法对摄像头的角度和画框区域有不同要求，所以算法选择后不允许被修改。
计划周期	在时间范围内，每隔多少分钟执行一次计划，计划周期支持小时、分两种单位选择

淘汰策略	执行计划技术架构中采用队列保护机制，当任务队列开始堵塞时为了保证一体机的稳定运行，系统提供了在这种情况下的淘汰机制，可以根据算法的重要性进行选择“直接上报”或者“丢弃事件” i. 直接上报：指发现AI算法识别的事件后当一体机处于繁忙状态时，系统不会进行二次确认，直接上报至云端，一般适用于有安全隐患类的算法，例如：消防通道占用 ii. 丢弃事件：指发现AI算法识别的事件后当一体机处于繁忙状态时，系统不会进行二次确认，直接丢弃掉，一般适用于没有安全隐患类的算法，例如：垃圾检测
摄像头执行列表	每次执行AI算法任务时，会对当前摄像头列表进行批量处理，根据实际需求选择矩形框或多边形框来进行区域标注，标注 区域将会进行AI算法计算
发现风险后的动作	执行计划在执行过程中发现某个摄像头中存在AI事件时，下一步动作的配置 i. 直接上报：指首次发现AI事件后直接上报至云端 ii. 二次确认：指首次发现AI事件后每隔多少分钟继续监测，连续发现多少次后在上报事件，通过系统的二次确认，来减少误报和错报，以便减少对用户的干扰报警

7. 配置完成执行计划后，在执行计划列表中可以查看到计划的数据。



参数	描述
计划名称	计划创建时输入的名称。
计划启停	计划创建后，计划的状态默认为停止状态，需要手工启动后，系统才会对计划进行周期性执行。
操作	配置：对任务内容进行修改，包含时间范围、计划周期、淘汰策略、摄像头执行列表、发现风险后的动作 执行记录：记录一体机执行的摘要记录，包含计划开始时间、计划运行耗时、事件数量、淘汰数量、运行摄像头数量 删除：将计划从一体机中删除，删除的计划不可恢复，请谨慎操作
执行视图	系统根据计划的配置进行24小时执行时段的标注，可视化视图方式观察到任务执行的时间点
风险时间段	系统根据一体机可以承受的最高负载对计划进行预检测，当检测到某一个时间段存在负载风险时，会提示这个时间段存在负载高峰，建议进行错峰配置