百度智能云

ABCROBOT 文档

目录

功能发布记录

发布时间	功能概述
2022-04	ABCRobot SDK v2.1.5正式发布。多模态语音交互方案，实现配置云控。增加SDK使用限制策略。设备ID算法升级。修复已知bug。
2022-03	ABCRobot SDK v2.1.2正式发布。新增多模态语音交互方案，并且支持戴口罩交互。新增音视频通话功能。支持离线TTS语音播报。全新设备指纹算法，增加日志上报功能，增加消息推送功能。
2020-03	ABCRobot SDK v2.0正式发布。SDK框架升级，规范功能接口定义，升级人脸检测、人脸识别功能，优化识别效果和性能。升级语音识别功能，优化识别效果。
2019-07	新增设备标签管理和数据统计功能。设备标签用于批量管理一组具有相同特征的设备，例如根据设备的地理位置、是否支持头部运动等，给设备打上不同的标签。数据统计功能提供多维度的数据指标和图表信息，可以帮助开发者掌握机器人的核心数据指标和走势，为开发者进一步开发和优化效果提供有效的数据支持
2019-04	ABC Robot提供了调用人脸库管理的相关接口。要完成人脸的管理，首先要构建人脸库，然后向人脸库中添加人脸图片
2018-12	百度机器人开发平台ABC Robot正式发布。为开发者提供硬件参考设计、SDK框架、开发工具包和一站式云端操作平台，彻底降低研发门槛；为行业客户精选优质合作伙伴，提供搭载百度Nuwa系统的机器人整机方案。

产品描述

概述

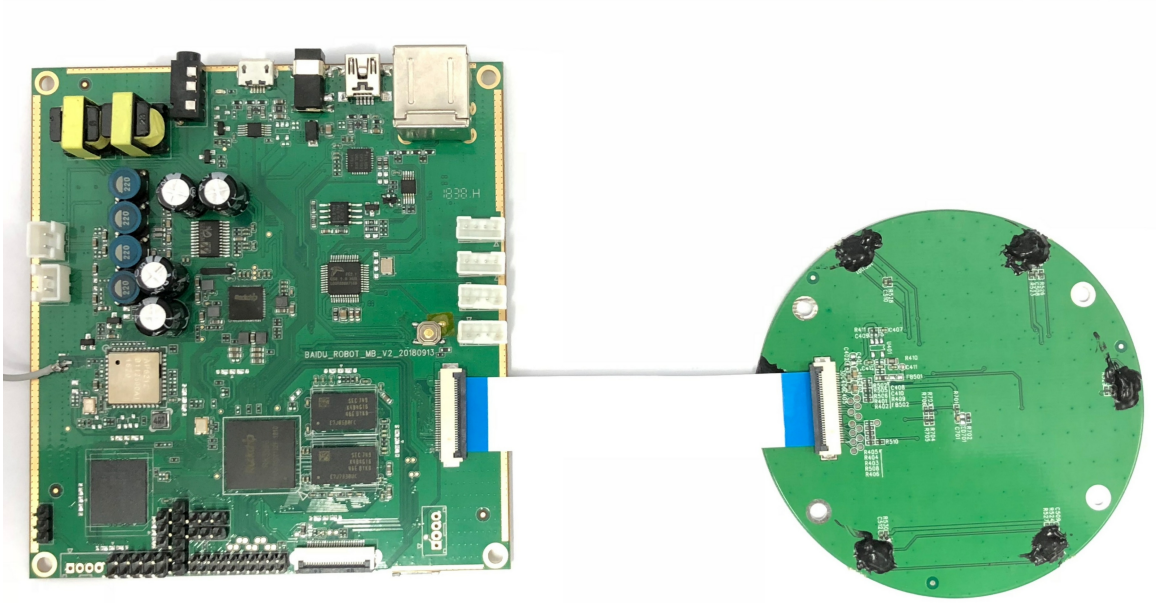
核心概念

开发机

百度提供麦克风阵列、视觉导航等硬件模组的参考设计和核心算法，以及主板硬件规范；由合作厂商基于此提供搭载 ABC Robot 的硬件实验装置，以便开发者快速上手实践。厂商需开放开发机的硬件参考设计，百度不参与各类开发机的售卖行为。

ABC Robot开发套件

包括百度自研的环形6-MIC阵列、RK3326，支持远场识音、回声消除、降噪、声源定位等功能，让机器人具备在空旷空间或嘈杂环境下全双工人机交互的能力。



RobotUI

百度自研的基于机器人场景的人机交互最佳实践，包括Android APP Demo+Android SDK，为您提供机器人前端应用的参考设计，大幅度减少开发难度和缩短产品化上线时间。

Boteye

百度自研的双目视觉导航方案，支持建图、定位、导航和避障功能，让机器人具备智能化的运动能力。

优势

端到端一站式解决方案破解AI技术栈难题

机器人涉及语音、视觉、NLP和运动导航等多项 AI 技术，传统集成商方案存在学习曲线陡峭、算法模型适配繁琐、工程化难度高、各项AI能力难以联动融合等问题。而ABC Robot依托百度在看、听、说、动四大核心AI技术方向的综合领先优势，为机器人厂商和行业开发者提供端到端的平台化解决方案，一站式解决高度综合的AI技术栈难题。

开放式全功能开发机彻底降低研发门槛

ABC Robot 联合硬件合作伙伴提供功能全面的服务机器人开发硬件实验平台，即ABC Robot 开发机。开发机使用百度的 MIC 阵列远场语音和双目视觉导航方案，并根据百度硬件规范要求设计。百度对外开放的 ABC Robot 可在开发机上运行，并供开发者二次开发实践。开发者通过使用搭载 ABC Robot 的开发机，配合百度 ABC Robot 平台提供的云端一站式服务，可快速验证自己的创意，无需在机器人方向有很深的硬件和软件底层积累，即可实现具备看、听、说、动四大能力的人机交互体验。开发机包含的完整开发套件可以彻底降低机器人研发门槛，打破行业准入壁垒。百度希望助力机器人厂商和开发者更专注于深耕场景化的服务机器人应用开发，实现合作共赢。

领先的视觉导航技术大幅降低硬件成本

ABC Robot 采用百度自研的 Boteye 双目视觉导航方案，包含 SLAM 定位、智能导航、避障、地图重建等核心功能，让机器人具备精准的室内定位和导航能力。与传统激光导航方案相比，百度 Boteye 视觉方案不仅能获取更多信息量，同时也可以大幅降低硬件成本，有助于机器人运动能力的持续升级和低成本规模化应用。

定义统一标准助力合作伙伴应用生态建设

服务机器人产业正在飞速发展，与之形成鲜明对比的是当前机器人行业生态却尚未成熟。开发者不仅需要针对每家机器人硬件本体单独开发软件，而且AI能力及IP内容提供商没有统一的接口和服务等级规范，导致机器人生产成本高昂，能力参差不齐，难以普及。

通过提供硬件参考设计、完整的开发套件和端云一站式解决方案，百度 ABC Robot 一方面可以为机器人整机制造商进行核心技术赋能，使其更聚焦于机器人品牌、渠道和产能本身；另一方面，可以为行业应用开发商提供一站式的云服务及开发工具平台，助力其更专注于深耕场景化的服务机器人应用开发，为机场、金融、新零售、医疗、政务、教育等领域提供专业的服务机器人解决方案。全面降低机器人行业的准入门槛，加速建设机器人产业生态。

应用场景

ABC Robot平台可以搭载人形机器人、智能桌面助理机器人、智能互动屏幕、智能终端设备等多种产品形态，应用在机场、金融、新零售、医疗、政务等行业场景：

- 机场

  承担机场服务大使的角色，对接机场航显等系统，为旅客提供航班问询、登机提醒、位置指引、机场设施问询、值机流程等服务，全面提升机场服务智能化程度。

- 金融

  承担大堂经理、迎宾接待等角色，快速识别VIP客户和普通客户，针对性提供理财产品推荐、业务分流、业务咨询、大盘指数、金融知识问询等服务，提升客户服务满意度。

- 新零售

  为顾客提供商品导购、商品推荐、商品介绍、位置指引等服务，沉淀顾客消费大数据，挖掘潜在消费能力，促进营销智慧化，打造零售新业态。

- 医疗

  为患者提供导诊、预诊、位置指引、医疗咨询等服务，进行患者分流，为医生提供预诊参考建议，辅助医生诊断决策。

- 政务

  为群众提供智能取号、政务办理、政务咨询等服务，为工作人员提供桌面智能助理服务，根据用户问题快速给出答案或推荐话术，辅助工作效率提升。

功能列表

语音

包括远场语音识别、情感语音合成、语音唤醒、声源定位、回声消除、多角度声音抑制、噪声抑制等功能。

人脸

包括1：N人脸识别、1：1人脸识别、属性检测、人脸跟踪、人脸库管理等功能。

运动

百度Boteye双目视觉导航方案，支持建图、定位、导航和避障功能。

智能知识库

基于FAQ构建知识库，支持图片、视频等富媒体知识，可通过单条新增和批量导入的方式维护知识。基于线上真实数据对语义模型进行自动优化，提升机器人召回率、准确率。

多轮对话

基于上下文语义理解，通过多轮对话的方式，收集用户表述中的关键信息，满足特定场景的任务型对话需求，如“航班查询”、“订座”等。

预置技能

我们针对不同场景为您提供了丰富的预置能力，您可以实现快速复用，免去您从零开始搭建机器人对话能力的烦恼。

应用配置

自定义设置机器人的人机交互参数，快速实现机器人的个性化定制。

产品定价

产品定价

目前机器人开放平台ABC Robot处于邀测阶段，如果您有购买意向，请发送邮件到 robot-bd@baidu.com 咨询。

快速入门

使用流程概览

本文档旨在帮助用户快速上手使用百度ABC Robot机器人平台服务。

> **说明：**
>
> 1. 首次使用，推荐您前往产品描述，了解ABC Robot平台产品介绍和核心概念。
> 2. 如果您想要了解更多功能，请参考操作指南。

步骤一、开通服务

1. 访问百度ABC Robot官网，点击页面右上角的管理控制台按钮。
2. 注册并登录您的百度智能云账号，请参考注册和登录。
3. 首次使用，需要开通服务。
   - 点击“**申请开通**”按钮，按页面提示填写信息，提交申请。
   - 提交申请后，百度智能云将尽快对您的申请信息进行审核，届时会有专人与您联系。请保持预留的联系方式畅通。
4. 服务开通

   申请信息审核通过后，将为您开通产品使用白名单。

步骤二、选择开发套件（可选）

我们为您提供了软硬一体的远场语音方案，包含麦克风阵列和开发板，支持远场拾音、声源定位、噪声消除、语音唤醒等能力，您需要在您的产品中集成百度ABC Robot开发套件，才能具备上述功能。如需购买开发套件请点击合作咨询。

> **说明：** 如果采用手机、平板等内置麦克风，则不具备远场拾音、声源定位等开发套件独有的功能。

步骤三、创建项目

1. 登录ABC Robot平台管理控制台。
2. 在“**项目管理**”页面，点击“**添加项目**”按钮，根据页面的提示信息创建一个项目。
3. 您可以在该项目下开发和管理云端和Client端的服务。

> 注：我们为您提供了2个项目配额供您试用，您也可以根据页面提示，申请更多配额。

步骤四、SDK下载和集成

1. 前往“**项目管理>概览>开发者资源**”，下载SDK，并获取该项目的Client ID和Client Secret。
2. 参考Android SDK开发指南将SDK集成到您的工程中。
3. 参考激活设备文档激活您的设备。
4. 在管理控制台“**设备管理**”页面，您可以看到已激活的设备，并管理设备信息。
   - 设备ID：设备的唯一标识，您可以通过该字段建立云端和设备端的对应关系
   - 设备名称：用于平台内机器人的辨识，并非机器人名字
   - 经纬度：机器人的坐标，用于天气等涉及地理位置的问答
5. 至此，您的产品已经成功接入百度机器人开放平台，可以提供基础的服务。此外，如果需要机器人完成更复杂的业务场景，您还要完成步骤五的工作。

步骤五、配置云端服务

1. 知识管理。通过问答库管理模块添加FAQ问答，实现不带上下文的一问一答能力。具体操作请参考问答库管理文档。
2. 技能管理。我们为您提供了丰富的系统预置能力，您可以直接在预置技能模块，勾选需要复用的技能。具体操作请参考预置技能文档。
3. 人脸与人脸库。通过人脸库管理模块添加人脸信息，实现VIP身份识别等功能。具体操作请参考人脸库管理。
4. 项目配置。通过兜底金句模块配置机器人异常情况的应对话术。具体操作请参考兜底金句文档。
5. 模拟测试：在平台模拟测试，验证机器人对话效果。具体操作请参考测试机器人文档。

步骤六、设备端开发

1. 基于平台提供的SDK，结合您的业务场景完成机器人端的应用开发。具体操作请参考Android SDK开发指南。

控制台操作指南

操作准备

控制台概述

通过控制台可以实现项目创建、设备激活、对话编排、数据标注、统计分析、知识库和人脸库管理等操作，再结合ABC Robot平台提供的SDK和API可快速完成机器人的二次开发。

主要功能如下：

1. **将您的产品接入ABC Robot平台。**
   - 开通服务：首次使用需要开通服务。
   - 项目管理：创建项目以管理一类机器人。
   - 设备管理：激活设备，管理设备和设备标签。
2. **基于平台提供的能力，快速搭建一个具备看、听、说、动多模态交互能力的机器人。**
   - 知识管理：管理FAQ形式的问答知识，实现无上下文的一问一答。

- 技能管理：管理系统预置的技能，丰富机器人常用能力。
- 人脸与人脸库：管理人脸库和人脸信息，实现VIP身份识别等功能。
- 项目配置：配置机器人的兜底金句，处理异常情况。
- 测试机器人：在网页上模拟测试，验证机器人对话效果。

3. **日常运营管理，监测机器人运营状态，持续优化交互效果。**
   - 效果优化：通过日志标注等手段持续优化机器人交互效果。
   - 数据统计：统计机器人的交互概况、知识分析、历史对话记录。
   - 权限管理：管理用户的权限。

开通服务

1. 访问百度ABC Robot官网，点击页面右上角的管理控制台按钮。
2. 注册并登录您的百度智能云账号，请参考注册和登录。
3. 首次使用，需要开通服务。
   - 点击“**申请开通**”按钮，按页面提示填写信息，提交申请。
   - 提交申请后，百度智能云将尽快对您的申请信息进行审核，届时会有专人与您联系。请保持预留的联系方式畅通。
4. 服务开通

   申请信息审核通过后，将为您开通产品使用白名单。

项目管理

使用百度智能云ABC Robot平台服务，需要先在管理控制台创建一个项目。项目是机器人的集合，用来管理一组功能相近的机器人。

> 例如，某机场客户在航站楼投放100台机器人，每个机器人的功能大体相近，略有差别，则可以创建一个项目和100台设备，用“**设备标签**”来管理个性化功能。

创建项目

1. 登录ABC Robot平台管理控制台。
2. 在“**项目管理**”页面，点击“**添加项目**”按钮，按照弹框提示填写信息。

> 每个用户默认用户有两个创建项目的配额，您也可以根据页面提示申请更多配额。

3. 点击“**确定**”按钮，完成项目创建。
4. 项目创建完成后，系统会自动分配一组Client ID和Client Secret，用于SDK集成。可前往“**概览>开发者资源**”获取。

编辑项目

1. 在“**项目管理**”页面，找到要编辑的项目，鼠标移动到项目卡片上方，点击“**编辑**”按钮。

2. 按照弹框提示填写信息，操作同“**创建项目**”。

删除项目

1. 在“**项目管理**”页面，找到要删除的项目，鼠标移动到项目卡片上方，点击“**删除**”按钮。

2. 在弹框中点击“**确认**”，即可删除项目。

> 注意：仅项目创建者才能删除项目。

设备管理

设备

创建项目后，需要在项目下完成设备的激活操作，方能访问ABC Robot平台服务，在设备端进行人机交互。通过设备管理模块，您可以完成申请设备配额、激活设备、编辑设备信息、绑定设备与标签等操作。

申请设备配额

一个配额可以激活一台设备，平台提供了2个试用配额供开发测试使用，如需更多配额，您可以提交申请。

操作步骤

1. 登录ABC Robot平台管理控制台。

2. 选择一个项目，左侧导航栏点击“**设备管理>设备**”，页面上方可查看当前项目下的配额。

3. 点击“**申请配额**”，输入申请数量和申请理由，点击“**确定**”。

（1）如果是新购买设备需要申请设备配额，请填写购买个数和购买日期等信息，以便更快的通过审核。

（2）如果是因为刷机导致设备ID变化，需要申请配额，请填写设备刷机前后的设备ID对应关系，以便更快的通过审核。如果不能提供设备刷机前的设备ID，请提供设备硬件编号。

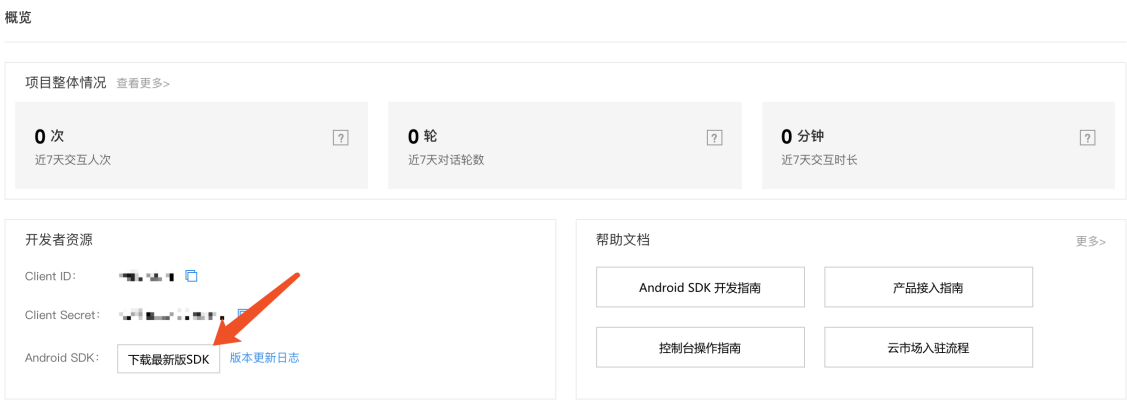
4. 审核通过后，会为您变更设备配额。

激活设备

设备接入ABC Robot平台前，需依照本文提供的激活方法，完成设备激活。激活完成后，设备才能访问ABC Robot平台服务。

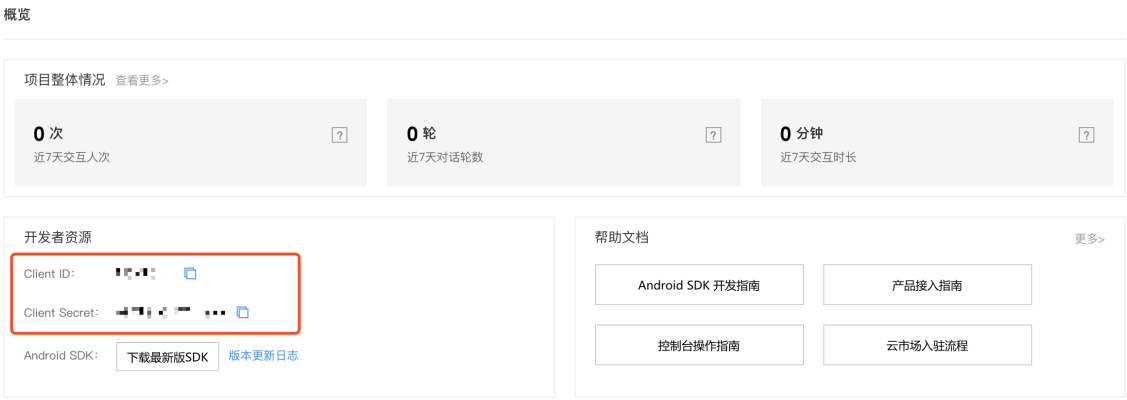
操作步骤

1. 确保项目下可激活的设备配额>0，申请配额请参见申请设备配额 文档。
2. 左侧导航栏点击“**概览**”，进入“**概览**”页面。
3. 在“**概览**”页面，点击“**下载最新版SDK**”，获取最新版Android SDK。



4. 在“**概览**”页面，获取项目的Client ID和Client Secret。

> 说明：所有设备均通过Client ID、Client Secret进行认证鉴权，请避免Client ID和Client Secret泄漏。



5. 在Android SDK中，填入Client ID和Client Secret，完成SDK初始化，具体请参阅初始化SDK文档。
6. 初始化过程会自动完成设备的注册激活，初始化完成后，在管理控制台“**设备管理>设备**”页面，可以查看最新激活的设备。为了便于区分不同设备，您可以修改设备名称等信息，具体请参考编辑设备文档。同时，SDK也提供了获取设备ID的接口，具体请参考获取设备ID文档。

7. 至此，已经完成了设备激活的所有步骤。

绑定设备与标签

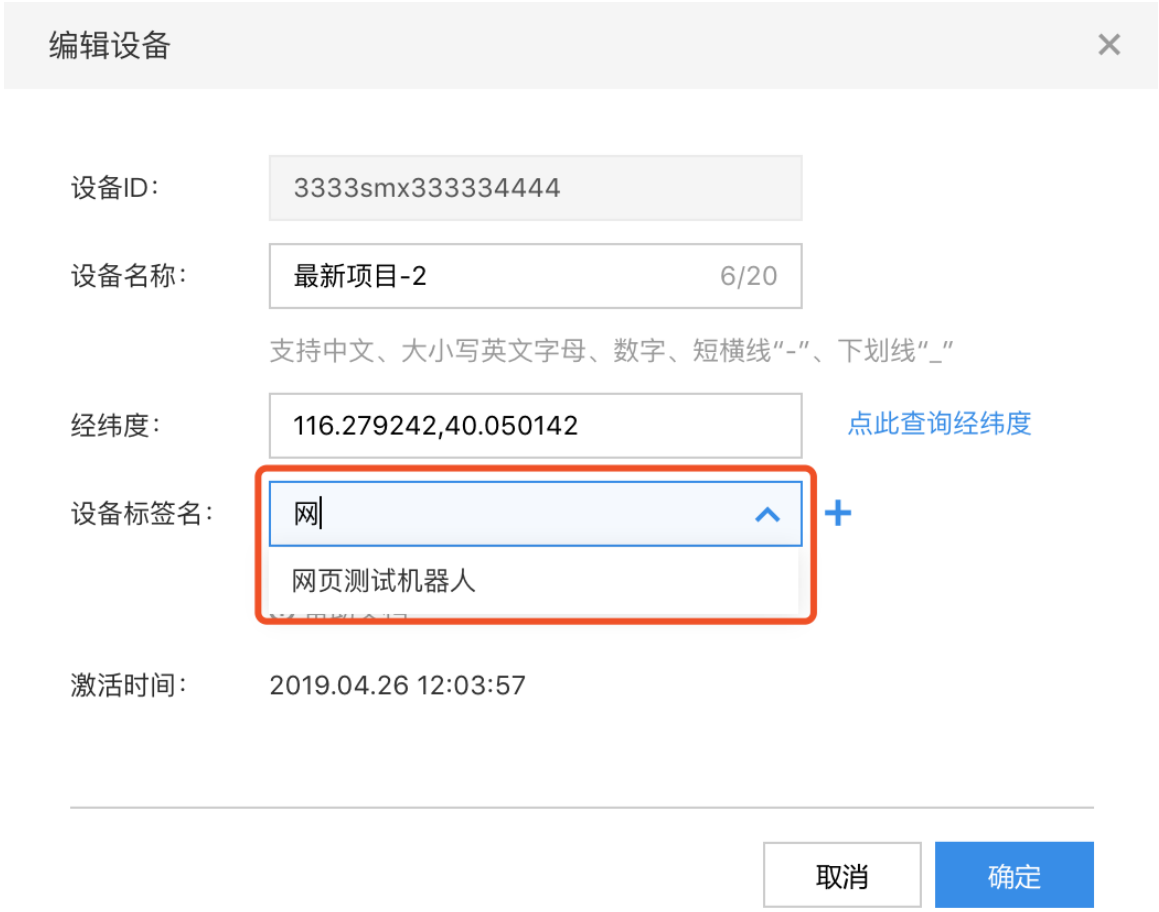
设备标签用于批量管理一组具有相同特征的设备，例如根据设备的地理位置、是否支持头部运动等，给设备打上不同的标签。同时在问答库中，对于同一个问题，如需不同设备回复不同答案，也可在答案中指定设备标签，来实现个性化回复。

支持单个或批量两种方式为设备添加标签：

单个设备绑定标签

1. 在“**设备列表**”页面，找到要绑定标签的设备，点击列表操作项中的“**编辑**”按钮。

2. 在弹框中，找到“**设备标签名**”字段，在输入框中可以选择已有标签或创建新标签。
   - **选择已有标签**：输入关键词搜索已有标签，在下拉框中选择目标项，点击“+”号添加下一个标签。



- **创建新标签**：直接在输入框中输入新标签，点击"+"号添加下一个标签

3. 点击确定，完成单个设备绑定标签。

批量绑定标签

1. 在“**设备列表**”页，勾选多个列表项，点击列表上方的“**批量编辑设备标签**”按钮。

2. 在弹框中，可以批量添加或删除选中设备的标签。操作同“**单个设备绑定标签**”。

3. 点击“**确定**”，完成编辑。

编辑设备

1. 在“**设备列表**”页面，找到要编辑的设备，点击列表操作项中的“**编辑**”按钮，即可编辑该设备信息。

2. 按弹框提示修改相关信息。

- 设备ID：设备的唯一标识，不可修改。SDK中也提供了查询设备ID的接口，具体参见获取设备ID。
- 设备名称：用于平台内机器人的辨识，并非机器人名字。
- 经纬度：机器人的坐标，用于天气等涉及地理位置的问答。
- 设备标签：通过标签批量管理机器人，可通过标签来建立“设备-标签-内容”的关联关系。同时在个性问答中，可指定答案对哪些设备标签生效。
- 激活时间：设备的激活时间。

查找设备

1. 按关键词模糊查找：设备名称、设备ID、设备标签。

设备标签

设备标签用于批量管理一组具有相同特征的设备，例如根据设备的地理位置、是否支持头部运动等，给设备打上不同的标签。同时在问答库中，对于同一个问题，如需不同设备回复不同答案，也可在答案中指定设备标签，来实现个性化回复。

新增标签

1. 登录ABC Robot平台管理控制台，进入一个项目，左侧导航栏点击“**设备管理>设备标签**”，进入“**设备标签管理**”页面。
2. 在“**设备标签管理**”页面，点击“**添加设备标签**”按钮，出现新建设备标签弹框。

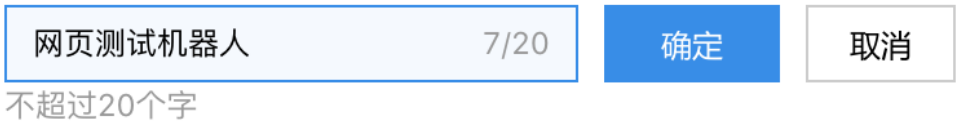
3. 在输入框中输入设备标签，按回车或“+”号批量添加。

4. 点击“**确定**”按钮，完成新建。

编辑标签

1. 在“**设备标签管理**”页面，鼠标移动到要编辑的列表项上方，点击设备标签后的“**编辑**”图标。

2. 在浮窗中修改设备标签名，点击“**确定**”，完成编辑。



删除标签

1. 方法一：单个删除。在“**设备标签管理**”页面，找到要删除的标签，点击列表操作项中的“**删除**”按钮，即可删除该标签。

2. 方法二：批量删除。在“**设备标签管理**”页，勾选多个列表项，点击列表上方的“**删除**”按钮，即可批量删除选中的设备标签。

查找标签

1. 按设备标签关键词进行模糊查找。

知识管理

问答库管理

问答库简介

问答库是基于NLP、大数据处理和深度学习等AI技术，利用智能化手段，在特定领域内构建的知识集合。

问答库的能力：支持以FAQ形式存储一问一答的知识，回复支持图片、视频等富媒体形式，能指定不同的设备回复不同的答案。

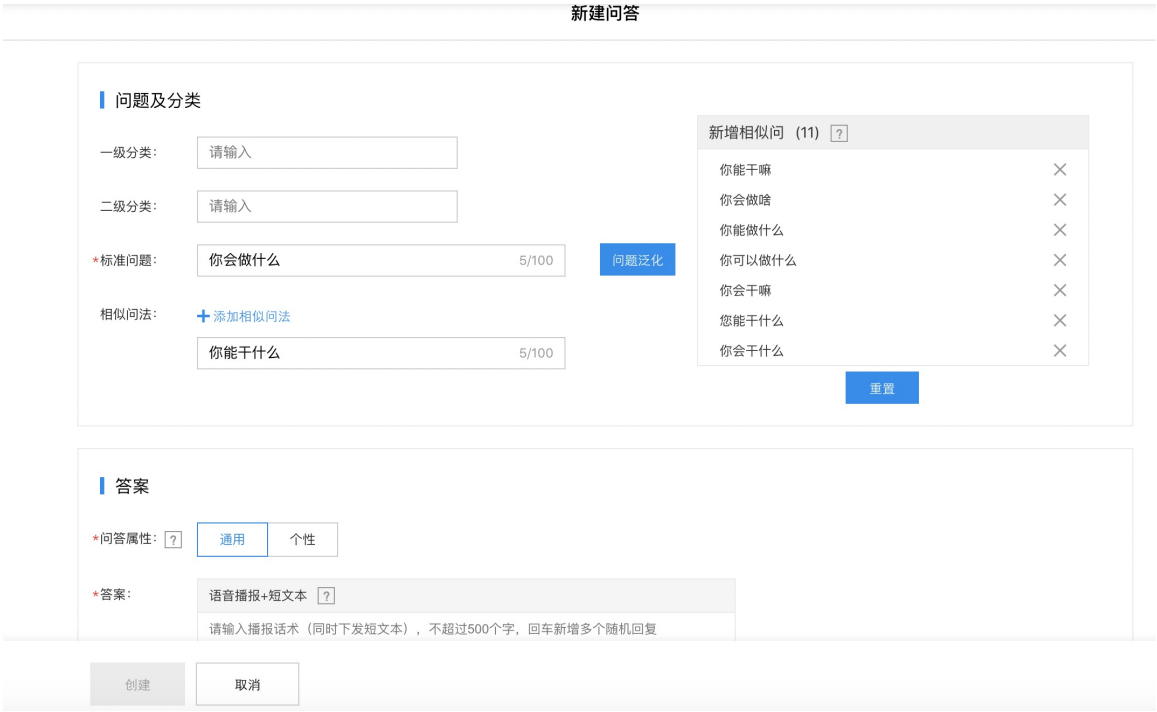
名词解释

- 一级分类、二级分类：可对业务知识进行两级分类，如一级分类“水果”，二级分类“苹果”。
- 通用：对于同一个问题，所有设备回复同一个答案。
- 个性：对于同一个问题，不同设备回复不同的答案。例如，分别位于二楼和三楼的机器人，对于“这里是几层？”这个问题，需回复不同的答案。
- 答案组：对于个性答案，可添加多个答案组，每个答案组需填写答案和对应的设备标签。
- 设备标签：用于批量管理一类设备，可以在个性答案中关联设备标签，实现不同设备回复不同答案。

单条新增

1. 登录ABC Robot平台管理控制台，选择一个项目，左侧导航栏点击“**知识管理>问答库管理**”，进入“**问答库管理**”页面。
2. 点击列表上方的“**新增**”按钮。

3. 按弹框提示填写问答内容。



- 一级分类、二级分类：选填，可对业务知识进行两级分类，如一级分类“水果”，二级分类“苹果”。
- 问题：按照尽可能贴合用户习惯的问法填写问题，避免书面用语。

> 添加更多相似问法可以增加机器人的泛化效果。

- 问题泛化：当点击【问题泛化】按钮时，会根据当前用户已经输入的标准问和相似问法进行泛化，并在“新增相似问”页框中展示泛化结果，点击"X"可以删除指定的相似问。

> “新增相似问”中的内容，将在保存问答对时自动保存到数据库并参与模型训练，本质上用户手动添加的相似问没有区别。

- 问答属性：
  - 通用：对于同一个问题，该项目下的所有机器人回复均一致。
  - 个性：对于同一个问题，该项目下的不同机器人可以设置不同的回复。如A机器人在一楼，B机器人在二楼，那对于同一个问题“卫生间在哪里”，答案应个性化回复。
- 答案：机器人的回复内容，支持文本、图片和视频类型。
- 答案组：当问答属性为“**个性**”时，可添加不同的答案组，每个答案组可以指定对应的机器人回复该答案组的答案。
  - 答案。
  - 设备标签：为答案组指定需要回复该答案的设备标签。

> 您可以在“**设备管理**”页面维护设备和标签的绑定关系。

编辑问答

在列表中找到该问答，在该问答的操作选项中选择"**编辑**"。编辑的操作方法同“**单条新增**”。

删除问答

1. 方法一：单条删除。在“**问答库管理**”页面，找到要删除的问答，点击列表操作项中的“**删除**”按钮，即可单条删除问答。

2. 方法二：批量删除。在“**问答库管理**”页面，勾选多个列表项，点击列表上方的“**删除**”按钮，即可批量删除选中的问答。

批量导入

点击列表上方的“**批量操作**”，并在下拉框列表中，选择“**批量导入**”，在弹出框中点击“**下载模板文件**”，下载模板文件后，将要导入的问答库数据填写到模板文件中，然后上传文件，并点击“**导入**”按钮即可。

批量导入时，平台会开启异步任务，并在前端展示导入进度。同时会自动对批量导入的问答对进行泛化，推荐更多的相似问，以提高泛化效果。

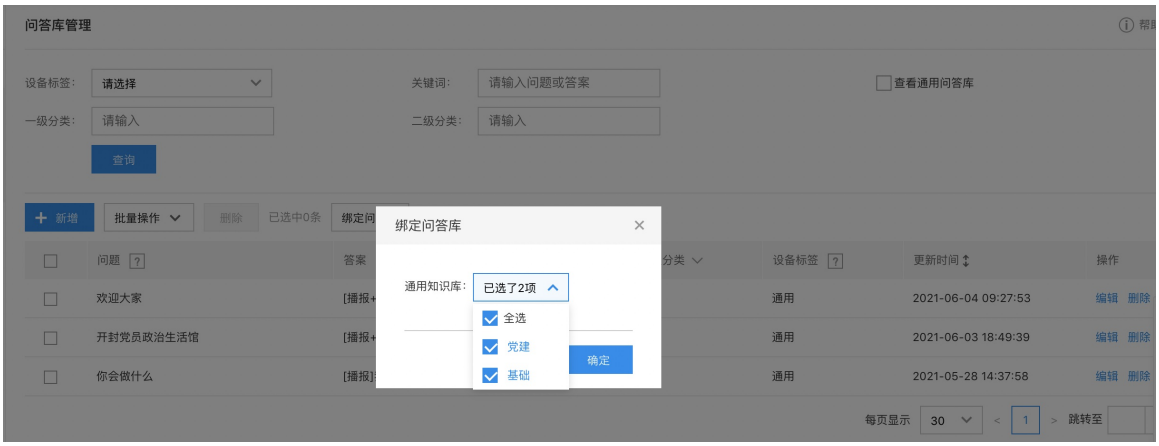
注意：这些推荐的相似问会自动保存到数据库，并参与问答库模型训练。

> 异步任务进行期间，不可以对问答库进行新增、编辑、删除等操作。

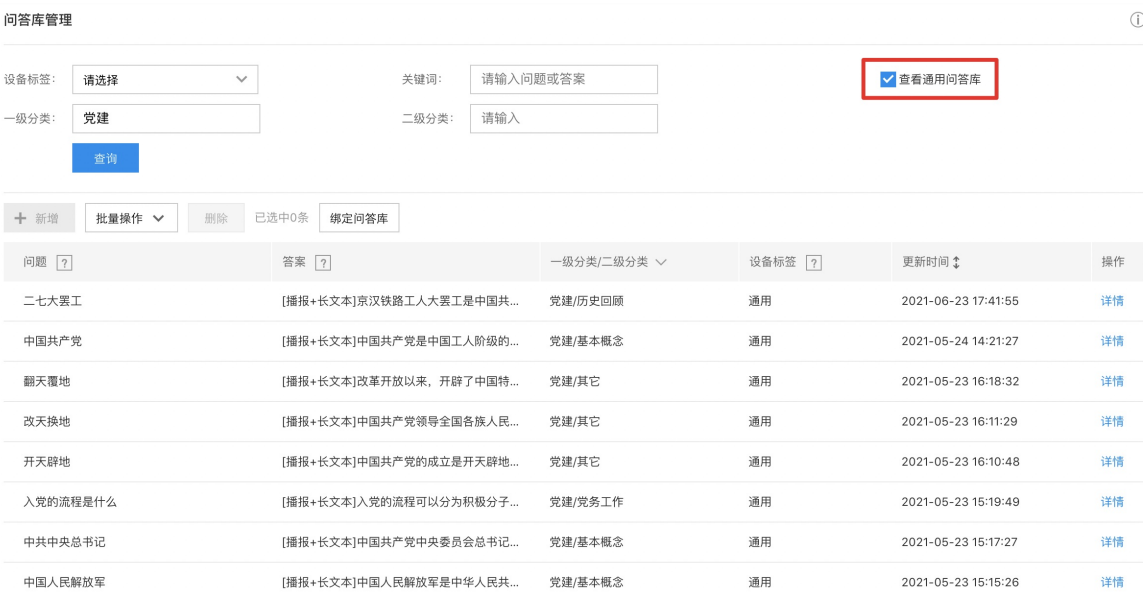
通用问答库

“**通用问答库**”是机器人平台内置的问答库数据。

1. 绑定通用问答库。点击列表上方“**绑定问答库**”，在弹出框中选择要绑定的问答库分类，点击确定即可。



2. 查看通用问答库数据。选中“**查询通用问答库**”选择按钮，然后点击“**查询**”按钮，即可展示通用问答库数据。通用问答库数据为机器人平台内置数据，只能查看，不允许修改，点击“**详情**”按钮，即可查看问答对详细数据。



查找问答

- 按关键词模糊查找：问题、答案、一级分类、二级分类。
- 按设备标签查找：查找对特定设备标签生效的问答。

技能管理

图形化技能

技能简介

图形化技能简介

什么是图形化技能

在智能对话领域中，一般可以将对话分成以下三种类型：

- **任务型：**有特定的任务目标，且需要特定的参数来完成这个任务。如“帮我查一下从北京到上海的航班”，“查询航班”是任务目标（即意图），“北京”和“上海”是具体的参数（即槽位）；
- **问答型：**有特定的任务目标，但不需要特定的参数。如“怎么退火车票”；
- **闲聊型：**开放域，没有特定的目标。如“你真聪明”。

任务型对话通常需要与实际的业务场景结合，解决具体的场景问题，因此也常常涉及到许多定制化的需求和开发。为了满足开发者大量定制化的开发需求，ABC Robot平台推出了图形化技能，帮助开发者通过简单的、图形化拖拽的方式，完成复杂场景下定制化需求的开发。

相较于填槽型技能，图形化技能提供了丰富的功能节点和定制化能力，通过节点的拖拽组合，能实现更加复杂的业务场景。

图形化技能的优势

- **结构清晰：**开发者在实现时，将自己的业务逻辑进行拆分，基本的逻辑由节点来表示，各个逻辑之间的联系通过连线实现，由此可以画出一个清晰的图，理解起来更直观。
- **强大的定制化能力**：通过不同类型节点的组合，可以实现复杂的业务场景，定制化能力更强。
- **云端托管：**基础的服务完全由平台托管，开发者无需投入过多资源。
- **使用门槛低：**开发者只需要简单了解NLP和AI相关知识，即可通过图形编排构建智能交互。

名词解释

术语	含义
意图	用户的某种目的或需求。如“订座” 。
问法/用户话术	用户的口语化表达，可以识别出特定的意图。如“我要订明天下午2点5个人的座位”
词槽/槽位	为了完成一个意图所需要的关键参数。如“明天下午2点”=“时间”，“5个人“=”人数“。
自定义词典	用户自己定义的词典。槽位下需要关联对应的词典，用于识别用户问法中的槽位。
系统词典	系统提供的常用词典。
交互流程	用户与机器人交互的一个对话场景，一般由对话流程图的方式来表示。
根节点	进入交互流程的起始节点，这个节点只是入口，没有条件判断。
判断节点	依据交互流程图的连接关系，判断节点定义的条件组，来决定走哪条分支。
填槽节点	定义参与填槽的所有参数，填槽交互由这个节点处理。
交互节点	返回给用户的内容，支持多种返回类型，同时支持结束本流程分支或等待用户下一轮输入。
测试	在网页上提供了模拟用户与机器人交互的测试窗口，支持展示交互中每一步的返回值和中间状态。
发布	提供交互流程统一发布的功能。

功能介绍

意图

意图

用户的意图可以理解为用户的某种目的或需求，在对话中常常以意图为单位来理解用户的表述。完成一个完整的意图，可能需要一轮或几轮对话。

常见问法

一个意图可能会有多种表达方式，**“常见问法”**用于识别用户什么样的表述可以进入到该意图，常见问法中可以包含意图所需的一些关键参数。

词槽

用户问法中包含的关键参数。如“我要给131****1212充话费”，手机号即为词槽。

词槽标注

鼠标选中对应关键词后，可以在用户常见问法中标注对应的词槽。

页面示例

词槽

词槽

表示意图需要的关键参数，开发者要明确定义，以免影响后续交互流程处理。词槽的取值需要关联相应的词典。

基本信息：

词典

平台提供常用的系统词典，开发者可以选择关联的系统词典。如果开发者要自定义词典，也可以点击并添加自定义词典值。

系统词典：

自定义词典：

交互流程

什么是交互流程

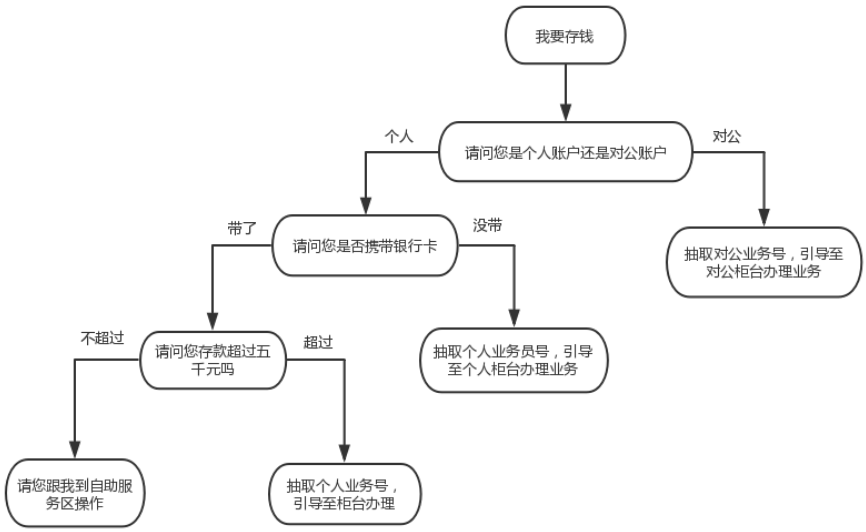
交互流程是用户对话的一个场景，我们将原本抽象、难理解的交互过程通过流程图表示，开发者可以通过绘制流程图的方式，来进行复杂业务流程的处理。

举例，常见的银行存款引导业务场景：

> 客户："我要存款"
>
> 引导员："请问您是个人账户还是对公账户"
>
> 客户："个人账户"
>
> 引导员："请问您是否携带银行卡"
>
> 客户："带卡了"
>
> 引导员："请问您存款金额超过五千元吗"
>
> 客户："不超过五千"
>
> 引导员："好的，请您跟我到自助服务区操作"

在这个场景里，用户明确表达了存款意图，但是缺少账户类型信息，所以需要询问用户账户类型，当收集到账户类型信息后，会引导用户去办理业务。

详细流程如下图：



在对话中，我们可以抽象出构成整个交互流程的基本元素，并捋清元素间的关联关系，从而将复杂的交互转化为清晰的图解表示，这就是我们设计交互流程图的初衷。在整个图中，用意图和槽位做原料，用节点和连线来组装，生成定制化和个性化的交互流程。

节点介绍

在交互流程设计中，会涉及以下几个节点的使用，具体介绍如下：

根节点

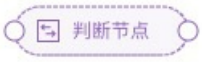
进入交互流程的入口，这个节点本身不会配置判断条件，与根节点连接的判断节点作为进入判断。

节点显示为： 

具体内容为：

判断节点

触发交互流程某个环节的节点，当满足该节点包含的条件，就可以触发该节点对应的交互环节。

节点显示为： 

具体内容为：

处理节点

处理节点允许开发者编写代码来进行复杂逻辑的处理，进而提供更智能的对话体验。如词槽值校验、查询API获取业务系统结果等。目前，开发者可以在页面上通过代码编辑的形式，编写自己的逻辑代码，并返回相应处理结果。

节点显示为： 

具体内容为：

下面这个代码编辑的例子中，我们以航班查询为例，涉及槽位值（原始值、归一化值）引用、结果返回、以及已有槽位值（原始值、归一化值）赋值。

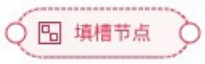
Node.js

```
exports.handle = (event, context, callback) => {
  try {
    // 返回正常结果
    var slotInfo = JSON.parse(event.Payload.slotInfos);
    var departure = slotInfo['FLIGHT_FROM_TO.filghdeparture'];
    var arrival = slotInfo['FLIGHT_FROM_TO.filgharrival'];
    if (departure == null) {
      departure = "北京";
    }
    var time = slotInfo['FLIGHT_FROM_TO.flight_time'];
    if (time == null) {
      time = "今天";
    }
    var response = {
      "functionResponse": JSON.stringify(
        {
          "departure": departure,
          "time": time,
          "arrival": arrival
        }
      ),
      "overrideResponse": JSON.stringify(
        {
          "FLIGHT_FROM_TO.filghtdeparture.origin":"北京"
        }
      )
    };
    callback(null, response);
  } catch(e) {
    // 将异常错误信息返回
    callback(e, null);
  }
};
```

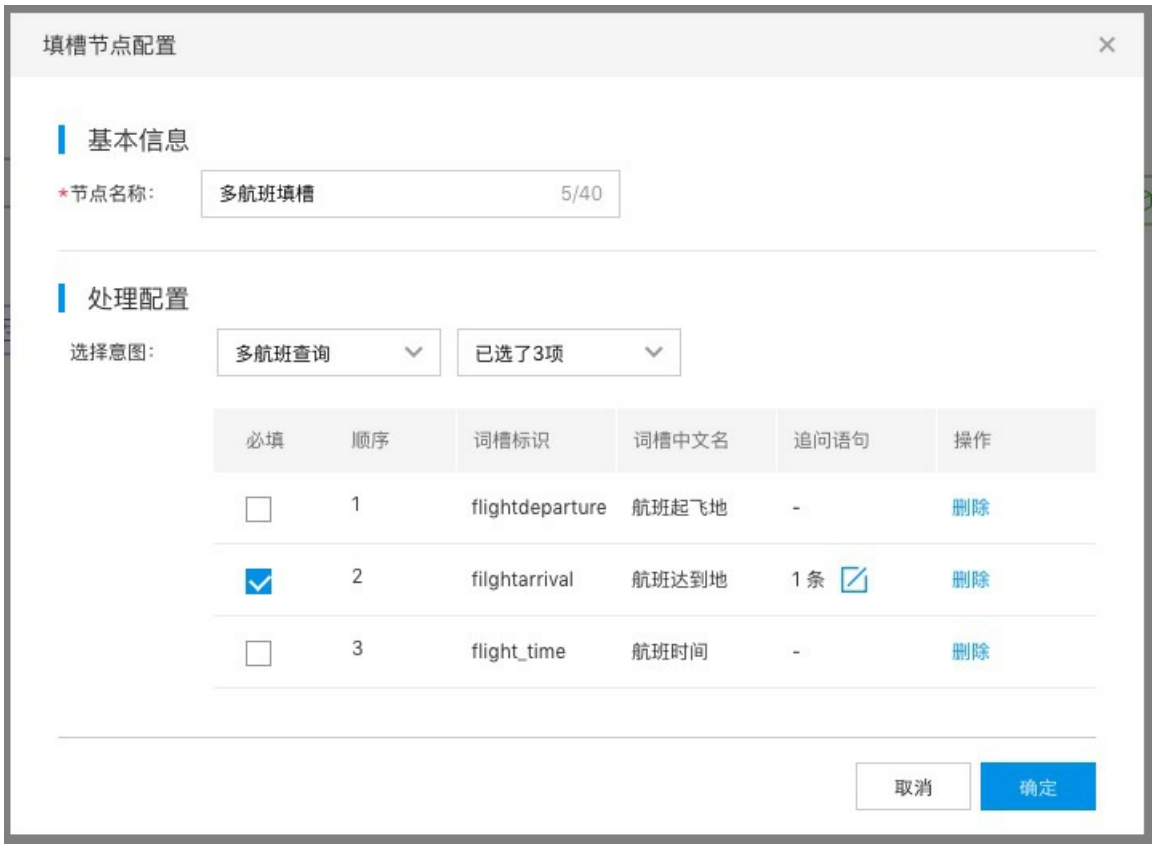
填槽节点

填槽节点是为某个意图统一收集槽位的节点，当满足触发条件的意图缺失槽位，会根据填槽节点的反问配置进行反问，直至收集齐。

> **根据槽位值进行流程分支：**对于需要在交互流程中通过词槽的值进行分支的场景，可以在填槽节点中只选择一个槽位，并在填槽节点后连接多个判断节点，判断具体槽位值后，进行交互流程的分支。

节点显示为： 

具体内容为：



交互节点

交互节点主要用来将交互的结果回复给客户端，同时支持等待客户端下一轮输入。交互的内容目前支持文本类型和自定义指令类型，自定义指令类型中支持按照平台的协议格式来返回多种类型的结果，如图片、列表等。

节点显示为： 

具体内容为：

怎样使用变量

什么是变量

在整个交互流程中可以使用的字段，我们称之为变量，比如意图的槽位、处理节点返回结果等。变量可以在交互节点、判断节点等被引用，来实现动态的内容回复，或根据变量值来进行具体的业务流程处理。

变量引用

变量	引用方式	说明
槽位归一化值	${intentName.slotName}	例如，${FLIGHT_FROM_TO.flightdeparture}
槽位原始值	${intentName.slotName.origin}	例如，${FLIGHT_FROM_TO.flightdeparture.origin}
处理节点返回值	${functionResponse.处理节点名称.字段名}	例如，${functionResponse.处理多航班查询.time}
变量路径	处理节点判断条件，直接引用变量的path	判断节点使用处理节点返回值做判断条件时，变量路径可以直接引用，例如：返回值为{"functionResponse":{"var1":{"var2":"value1"}},"var3":"value2"}，引用var2时使用var1.var2，引用var3时使用var3。

已有变量赋值

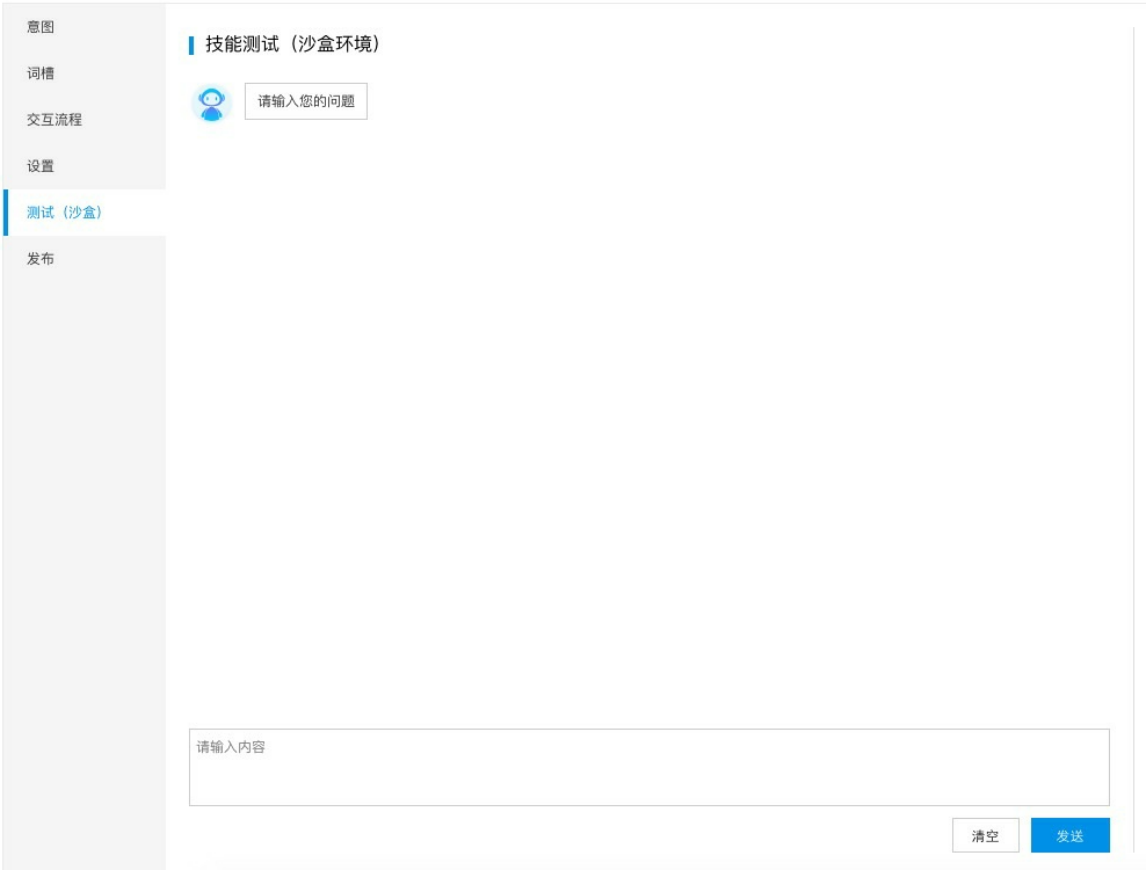
在返回值中添加**overrideResponse**字段，数据类型是JSON，每个字段的KEY是已有变量名，VALUE是将要赋值的具体值，例如：

```
  "overrideResponse":{
    "FLIGHT_FROM_TO.filghtdeparture.origin":"北京"
}
```

测试

测试

平台提供模拟测试功能，开发者可在网页上模拟用户与机器人的交互过程，测试机器人的交互效果。



发布

发布

发布是将在沙盒环境下编写的交互流程统一发布到线上环境，线上环境比沙盒环境更稳定，且不会受到沙盒环境数据变动的影响。发布需要分两步走：

第一步：需要先点击【申请】，申请线上环境的发布权限；

第二步：申请通过后，点击【发布到线上环境】，选择要发布的交互流程并填写版本说明，将交互流程发布到线上环境。

实践案例

案例一：简单的订餐流程

场景

本案例实现简单的订餐场景，具体实现见下面的操作视频。

操作视频

案例二：存款业务

场景

用户想要存款，询问账户类型种类，并引导用户完成存款业务。具体见下面的操作视频。

操作视频

案例三：航班查询

场景

用户查询某个时间从A地飞往B地的航班列表；用户查询某个航班的具体信息。具体实现见下面的操作视频。

操作视频

填槽型技能

技能设置

平台代码编辑

ABC Robot平台提供代码编辑模块，开发者直接在平台上编辑代码，即可处理技能回复逻辑。本质是提供一个FaaS服务（Function as a Service），让开发者专注于技能业务代码的开发，无需关注服务器资源（Serverless），来减少自定义技能的开发成本。本文主要介绍自定义技能代码编辑如何使用，帮助开发者完成技能服务的开发，实现多轮对话能力。

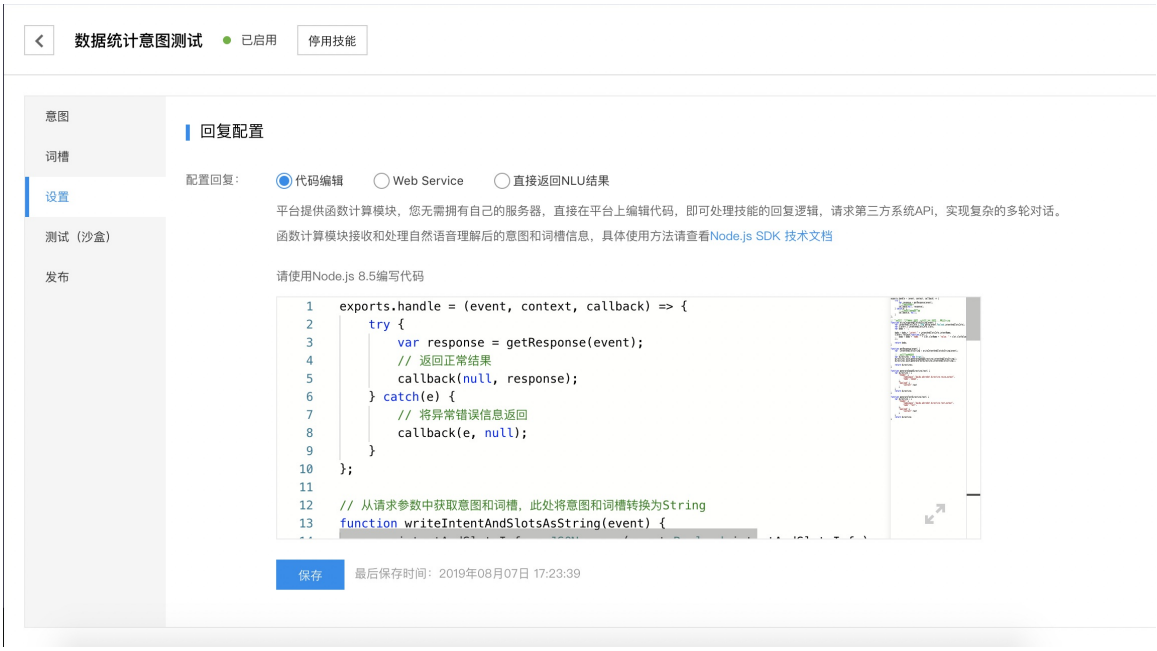
在填槽完毕后，您在平台上编辑的代码会收到机器人平台的请求，请求中带有语义理解后的意图和槽位信息，您可以根据这些信息进行具体的业务处理，随后返回符合机器人平台格式要求的响应。

配置代码编辑

1. 登录ABC Robot平台管理控制台，选择一个项目，左侧导航栏点击“**技能管理>自定义技能**”，进入“**自定义技能**”页面。

2. 点击“**新建技能**”按钮，按照提示创建一个技能。

3. 新建技能后，进入自定义技能设置页。默认会选择当前已经设置的回复类型，如果之前技能还没有设置回复类型，那么默认选中“**代码编辑**”，并在代码编辑框中展示系统配置的示例代码。



4. 回复方式选择“**代码编辑**”，修改代码，并保存，即可生效

> 注：代码编辑当前仅支持Node.js 12
>
> **代码编辑开发** 页面上配置的代码会收到机器人平台的请求，请求中带有NLU（自然语言理解）后的意图和词槽信息。在代码中解析意图和词槽内容，根据自身的业务逻辑来生成技能的回复结果，并按照ABC Robot平台要求的格式返回结果。请求和响应参数请参考：自定义技能协议。代码示例如下：

本示例将请求中意图和槽位拼接为字符串，并将该字符串设置为返回结果中指令的内容，并返回结果给调用方。

```
 exports.handle = (event, context, callback) => {
try {
 var response = getResponse(event);
 // 返回正常结果
 callback(null, response);
} catch(e) {
 // 将异常错误信息返回
  callback(e, null);
}
};

// 从请求参数中获取意图和词槽，此处将意图和词槽转换为String
function writeIntentAndSlotsAsString(event) {
 var intentAndSlotsInfo = JSON.parse(event.Payload.intentAndSlotsInfo);
```

```
 var slotArr = intentAndSlotsInfo.slots;
 var body = "";

 body = body + "intent:" + intentAndSlotsInfo.intentName;
 slotArr.forEach(function(slot) {
  body = body + "name: " + slot.slotName + ",value: " + slot.slotValue + ";";
 });

 return body;
}

function getResponse(event) {
 var  intentAndSlotsString = writeIntentAndSlotsAsString(event);
 // 生成要返回的指令
 var directives = new Array();
 directives.push(generateSpeakDirective(intentAndSlotsString));
    directives.push(generateTextDirective(intentAndSlotsString));

 var response = {
  "logId":event.Payload.logId, // 将传入的logId原样返回，以方便定位问题
  "directives":directives
 };
 return response;
}

function generateSpeakDirective(text) {
 var directive = {
  "header":{
   "namespace":"baidu.abcrobot.directive.voice_output",
   "name":"Speak",
  },
  "payload":{
   "content":text
  }
 }
 return directive;
}

function generateTextDirective(text) {
 var directive = {
  "header":{
   "namespace":"baidu.abcrobot.directive.text_output",
   "name":"Text",
  },
  "payload":{
   "content":text
  }
 }
 return directive;
}
```

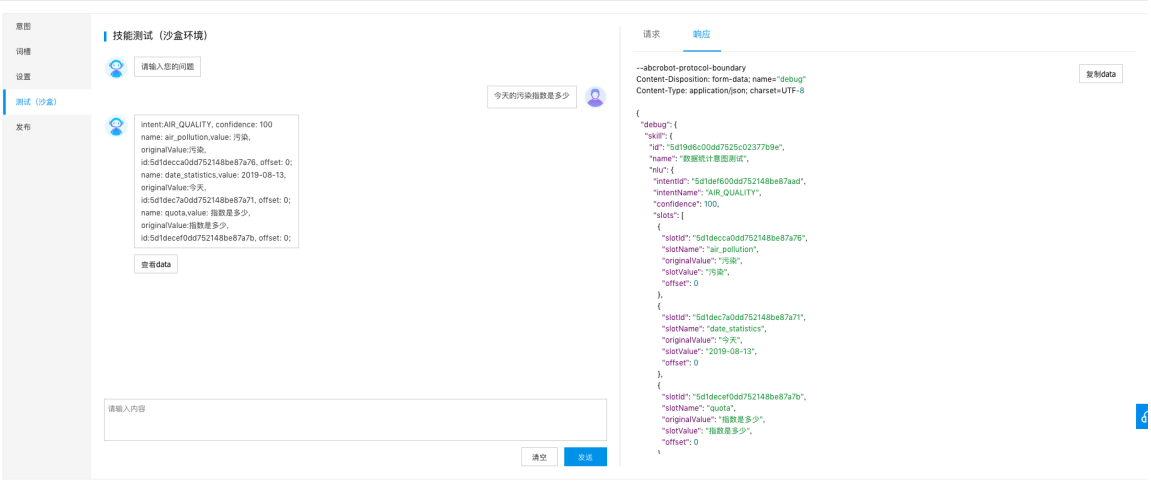
说明

1. 代码编辑的处理入口是exports.handle = (event, context, callback)为固定写法，其中，意图和槽位的内容可以在event中获取。callback中的第一个参数为异常信息，可将第一个参数设置为异常信息，如果没有异常设置null；第二个参数为正常的结果，可将第二个参数设置为正常结果，如果有异常时，将第二个参数设置为null；如果两个参数都设置优先选择第二个参数的值。
2. 请求参数可以通过event.Payload.intentAndSlotsInfo获取，需要先将参数JSON格式化。intentAndSlotsInfo的内容参考自定义技能协议。

3. 响应结果为指令数组，指令的具体内容参考自定义技能协议。

代码编辑调试 代码编辑完成，点击保存后，可在测试窗口中进行技能测试，并查看运行结果和日志。

1. 点击“**测试（沙盒）**”按钮，进入测试窗口，进行问答测试。



2. 当填槽完毕后，点击测试框中的“**查看data**”链接，在右侧的“**响应**”tab中，可以看到返回的指令信息。其中Content-Disposition: form-data; name="debug"后面的内容是调用配置代码时产生的调试日志。调试日志中各个字段含义如下：

字段名称	含义
debug.skill.id	技能ID
debug.skill.name	技能名称
debug.skill.nlu	命中的意图和填充的词槽值
debug.skill.cfc.response	正常情况下该字段的内容是配置代码的响应结果。如果代码运行报错，该字段为错误提示信息
debug.skill.cfc.logResult	代码执行时的环境信息，代码中console.log打印的日志，以及解析结果异常信息

WebService

本文主要介绍自定义技能Web Service如何使用，帮助开发者完成技能服务的开发，实现多轮对话能力。

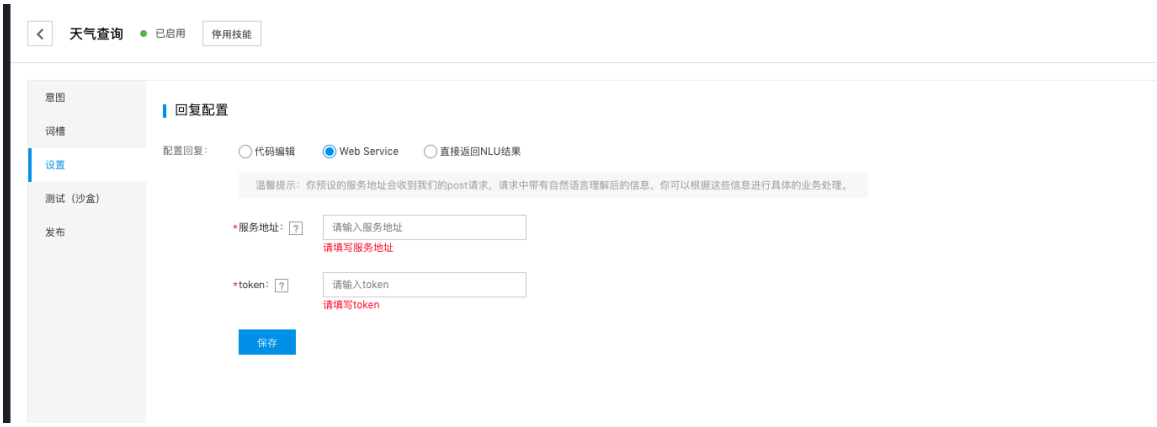
在填槽完毕后，您预设的服务地址会收到机器人平台的请求，请求中带有语义理解后的意图和槽位信息，您可以根据这些信息进行具体的业务处理，随后返回符合机器人平台格式要求的响应。

配置服务地址

1. 登录ABC Robot平台管理控制台，选择一个项目，左侧导航栏点击“**技能管理>自定义技能**”，进入“**自定义技能**”页面。

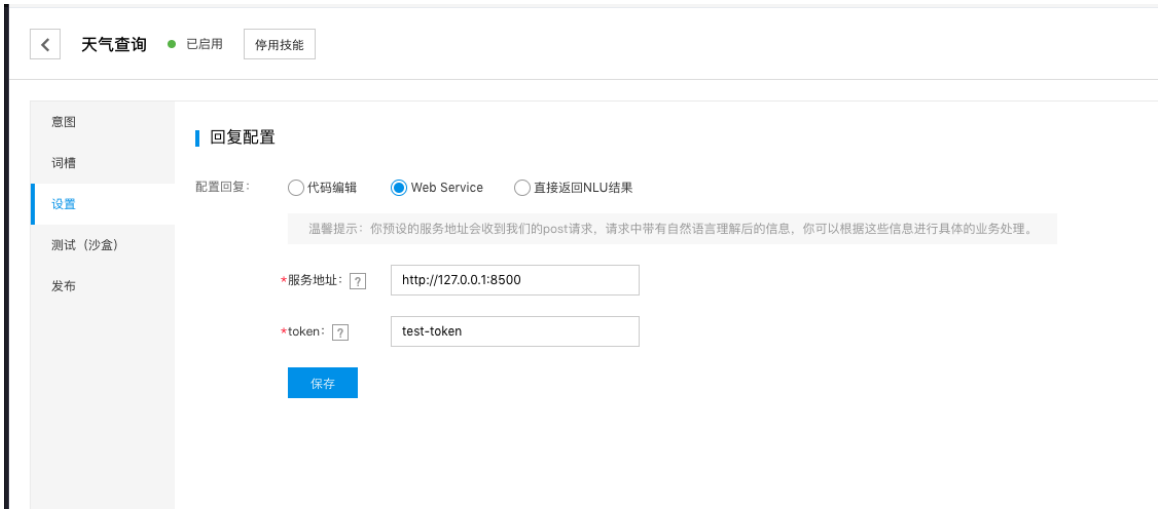
2. 点击“**新建技能**”按钮，按照提示创建一个技能。

3. 新建技能后，进入自定义技能设置页。



4. 回复方式选择"**Web Service**"，填写服务地址及token。

> 注：服务地址以及token都是必填项，建议在您的服务端对token进行验证



开发服务接口 页面上配置的服务地址会收到技能的请求，请求中带有NLU（自然语言理解）后的意图和词槽信息。您需要搭建一个Web Service，根据自身的业务逻辑来处理技能的回复结果，并按照ABC Robot平台要求的格式进行返回。 **请求与响应说明**

请求说明

- 请求方式：post
- 请求地址：服务地址与token的拼接 如服务地址为`http://127.0.0.1:8500`，token为`auth`，则请求url为`http://127.0.0.1:8500?token=auth`
- 请求参数参考：自定义技能协议

响应说明

- 响应参数说明参考: 自定义技能协议

自定义技能协议

当为自定义技能配置了Web Service或代码编辑模块，就可以处理技能回复逻辑。在多轮对话填槽完毕后，这些模块会收到机器人平台的请求，请求中带有语义理解后的意图和槽位信息，Web Service或代码编辑模块根据这些请求信息进行具体的业务处理，并返回符合机器人平台格式要求的响应。下面介绍请求和响应的具体内容。

请求参数 机器人平台以POST方式将意图和槽位信息发送给Web Service或代码编辑模块。请求参数的格式如下：

请求说明 说明

参数名	参数类型	参数含义
logId	STRING	本次请求的唯一标示，用于查日志
intentAndSlotsInfo		语义处理结果
intentAndSlotsInfo.intentName	STRING	意图名
intentAndSlotsInfo.intentId	STRING	意图id
intentAndSlotsInfo.confidence	DOUBLE	置信度
intentAndSlotsInfo.slots		具体的槽位值列表
intentAndSlotsInfo.slots.slotName	STRING	槽位名
intentAndSlotsInfo.slots.slotValue	STRING	槽位值原始值归一化后的值
intentAndSlotsInfo.slots.originalValue	STRING	槽位原始值
intentAndSlotsInfo.slots.slotId	STRING	槽位id
intentAndSlotsInfo.slots.offset	INT	该槽位值填槽的轮数
deviceTags	STRING	当前对话设备关联的标签名称，多个标签名称使用","分割，可能为空
deviceId	STRING	当前对话设备ID

请求示例

```
{
    "logId":"{{STRING}}",
    "intentAndSlotsInfo":
     {
          "intentName":"{{STRING}}",
          "intentId":"{{STRING}}",
          "confidence":"{{DOUBLE}}",
          "slots":[
             {
                "slotName":"{{STRING}}",
                "slotValue":"{{STRING}}",
                "originalValue":"{{STRING}}",
                "slotId":"{{STRING}}",
                "offset":"{{INT}}"

             }
          ]
       }
     "deviceTags":"{{STRING}}",
     "deviceId":"{{STRING}}"
 }
```

响应参数 Web Service或代码编辑模块按具体的业务逻辑处理意图和槽位后，返回符合机器人平台格式要求的响应。响应参数的格式如下：

响应说明 | 参数名 |参数类型 |参数含义 | | --- | -- |--- | | logId |STRING| 建议与接收到request中的logId一致 | | directives | |返回给机器人的指令列表 | | directives.header.namespace |STRING| 指令namespace | | directives.header.name | STRING|指令name | | directives.payload | |指令payload |

响应示例

```
{
  "logId":"{{STRING}}",
  "directives":[
     {
        "header":{
           "namespace":"baidu.abcrobot.directive.directive_namespace_sample",
           "name":"directive_name_sample"
        },
        "payload":{

        }
     }
  ]
}
```

可选指令列表 您可以通过返回不同的指令对机器人端进行控制，以下是机器人平台支持的指令列表： **1. 语音播报指令** 用途: 语音播报

```
{
 "header": {
  "name": "Speak",
  "namespace": "baidu.abcrobot.directive.voice_output"
 },
 "payload": {
  "content": "今天天气晴朗"
 }
}
```

2. 纯文本展示指令 用途: 纯文本展示，一般用于播报时播报语的显示

```
{
 "header": {
  "name": "Text",
  "namespace": "baidu.abcrobot.directive.text_output"
 },
 "payload": {
  "content": "今天天气晴朗"
 }
}
```

3. 卡片展示指令 用途: 屏幕主体部分的信息展示

> 注:卡片展示指令有五种子卡片类型，分别是图片卡片，文本卡片，列表卡片，简单图片列表卡片，视频列表卡片

- 图片卡片

```
{
  "header": {
    "name": "RenderCard",
    "namespace": "baidu.abcrobot.directive.screen"
  },
  "payload": {
    "type": "ImageCard",
    "title": "图片卡片标题",
    "image": "图片地址",
    "content": "描述信息",
    "url": "可选，如果图片可以点击，点击图片的跳转地址"
  }
}
```

- 文本卡片

```
{
  "header": {
    "name": "RenderCard",
    "namespace": "baidu.abcrobot.directive.screen"
  },
  "payload": {
    "type": "TextCard",
    "content": "文本信息",
    "url": "可选，如果文本可以点击，点击文本的跳转地址"
  }
}
```

- 列表卡片

```
{
  "header": {
    "name": "RenderCard",
    "namespace": "baidu.abcrobot.directive.screen"
  },
  "payload": {
    "type": "ListCard",
    "list": [
      {
        "title": "图片卡片标题1",
        "image": "图片地址1",
        "content": "描述信息1",
        "url": "可选，如果图片可以点击，点击图片的跳转地址"
      },
      {
        "title": "图片卡片标题2",
        "image": "图片地址2",
        "content": "描述信息2",
        "url": "可选，如果图片可以点击，点击图片的跳转地址"
      }
    ]
  }
}
```

- 简单图片列表卡片

```
{
  "header": {
    "name": "RenderCard",
    "namespace": "baidu.abcrobot.directive.screen"
  },
  "payload": {
    "type": "SimpleImageListCard",
    "list": [
      {
        "image": "图片地址1",
        "url": "可选，如果图片可以点击，点击图片的跳转地址"
      },
      {
        "image": "图片地址2",
        "url": "可选，如果图片可以点击，点击图片的跳转地址"
      }
    ]
  }
}
```

- 视频列表卡片

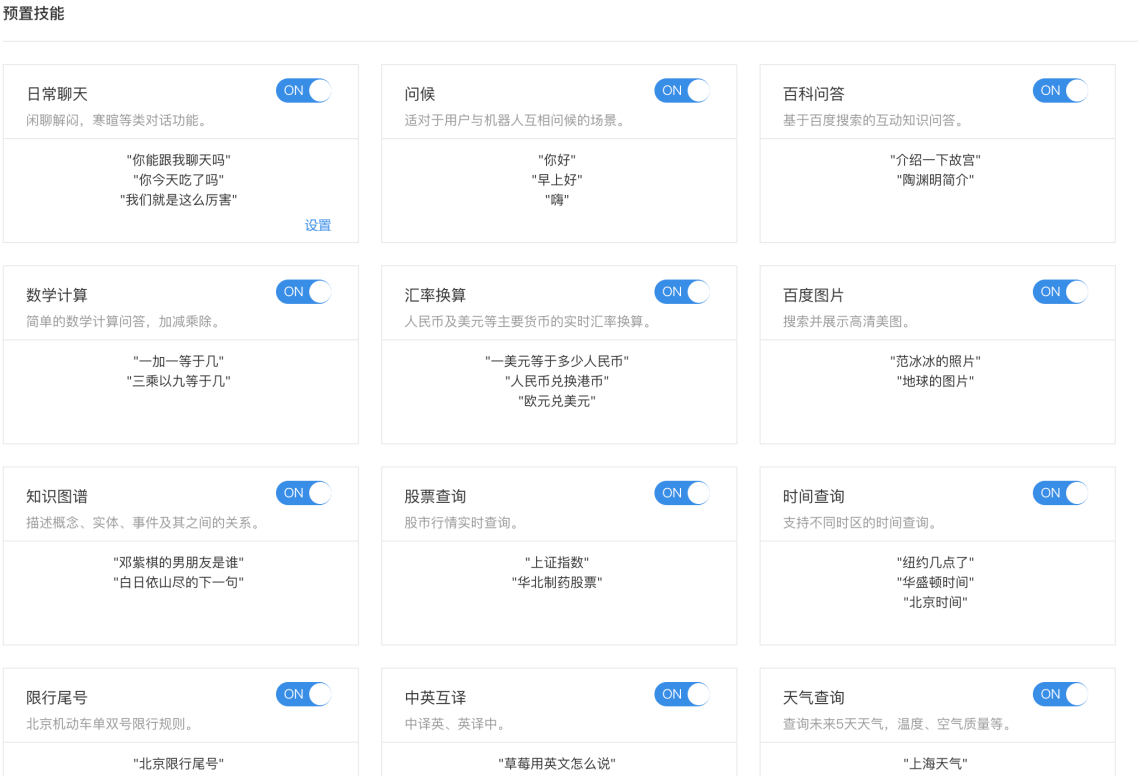
```
{
  "header": {
    "name": "RenderCard",
    "namespace": "baidu.abcrobot.directive.screen"
  },
  "payload": {
    "type": "SimpleImageListCard",
    "list": [
      {
        "video": "视频地址1",
        "thumbnail": "可选，视频缩略图1"
      },
      {
        "video": "视频地址2",
        "thumbnail": "可选，视频缩略图2"
      }
    ]
  }
}
```

4. 页面渲染指令

```
{
 "header": {
   "name": "RenderHtml",
   "namespace": "baidu.abcrobot.directive.screen"
 },
 "payload": {
   "url": "需要渲染的url地址"
 }
}
```

预置技能

预置技能是百度机器人平台为开发者预置的系统技能，开发者只需要简单配置，就能获得相应的垂类能力。目前包括天气查询、时间查询、日常聊天、问候、百科问答、百度图片、知识图谱、股票查询等能力，未来还将逐步开放更多技能。



第三方技能

简介

如果开发者在接入ABC Robot机器人平台时，自己已经购买或开发了智能交互应用（如智能客服、知识库等），机器人平台开放了第三方技能接入功能，帮助开发者快速将自己的应用对接到机器人平台。

接入方式

开发者将自己的服务抽象成统一的HTTP接口，并将接口地址登记在机器人开放平台上，机器人端的请求会经过机器人大脑分发给开发者的第三方技能，接收并处理第三方技能的返回结果，页面入口如下：

配置信息

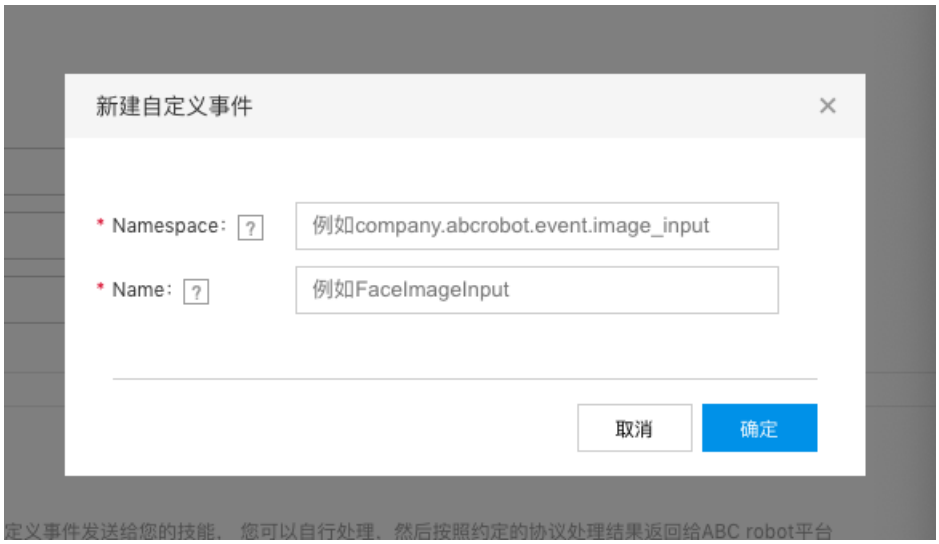
技能信息 开发者需要登记自己的技能名称、技能描述、技能服务地址、示例问法等信息，机器人大脑会根据相关信息，将客户端的请求转发给开发者的第三方技能，页面配置如下：

授权配置 开发者提供的接口鉴权采用OAuth2.0 Client Credentials Grant模式，开发者提供clientId、clientSecret、OAuth服务地址，并登记到机器人平台。调用接口前，机器人大脑会获取接口访问的access token，并使用这个access token访问开发者的接口，页面配置如下：

自定义事件

开发者接入第三方技能后，机器人大脑会将语音识别文本的TextInput事件发送给第三方技能，同时允许开发者按照端云交互协议的事件格式自定义事件。开发者将自定义事件header的namespace和name登记在机器人平台，当机器人客户端使用自定义事件发起请求时，机器人大脑会匹配自定义事件的header信息，匹配成功后将事件发送给开发者的服务，机器人大脑处理相关返回值发送给机器人客户端。自定义事件的payload内容，机器人大脑不做限制，完全是开发者的私有协议，机器人大脑只做透传，页面配置如下：

开发者输入自定义事件的namespace和name，如下图：



请求格式

```
 POST /thirdparty/service/url
Authorization bearer {{access token}}


{
   "interactionId":"a977b80d-7e39-4eef-b556-443de830234e", // 机器人大脑维护的交互id
   "sessionId":"6ed00b2f-c8f4-4509-99af-bf8a922f8d9d", // 机器人大脑维护的session id
   "deviceId":"E248-91D6-348B-6A03", // 设备id
   "clientId":"FEALz7He", // 机器人平台项目的client id，请前往"概览>开发者资源"获取
   "eventRequest":{ // eventRequest中的event为端云交互协议的事件，clientContextV2为端云交互协议的端状态（KV结构）
      "event":{
         "header":{
            "name":"TextInput",
            "namespace":"baidu.abcrobot.event.text_input",
            "messageId":"msg_6oG",
            "interactionId":"inter_DpaUSmBx"
         },
         "payload":{
            "query":"今天天气"
         }
      },
      "clientContextV2":{
         "a": "b"
      }
   }
}
```

返回格式

```
{
  "errorCode":0, // 错误码
  "errorMsg":"", // 错误描述
  "success":true, // 是否成功被第三方技能接住，机器人大脑会判断此字段
  "seizeNextInput": false, // 通知机器人大脑是否抢占下一轮客户端输入（例如，在一些流程办理的场景下，需要当前技能连续不中断地处理用户的请求，直至完成当前业务）
  "directives":[ // directives为端云交互协议的指令列表
    {
      "header":{
        "namespace":"baidu.abcrobot.directive.voice_output",
        "name":"Speak",
        "interactionId":"a977b80d-7e39-4eef-b556-443de830234e",
        "messageId":"70fbab19-5710-4993-a533-3b2aa8372148"
      },
      "payload":{
        "content":"今天天气晴，阳光不错。"
      }
    }
  ]
}
```

人脸与人脸库

人脸概述

人脸识别简介

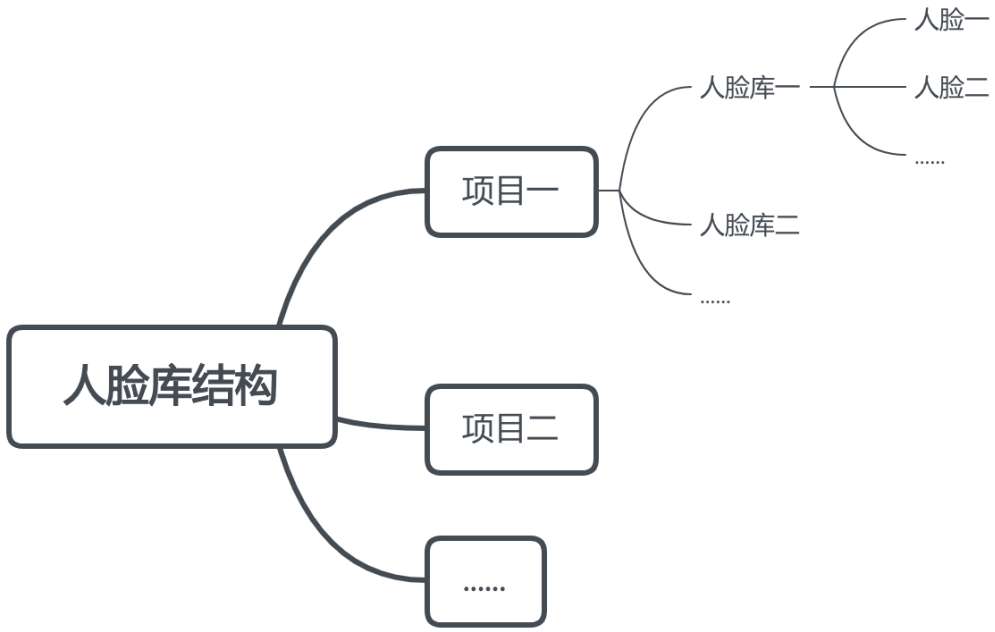
ABC Robot平台基于百度领先的人脸识别和分析技术，提供人脸检测与分析、五官定位、人脸查找、人脸比对、活体检测等多种服务，可应用于智慧零售、智慧楼宇等多种应用场景，充分满足各行业客户的人脸属性识别及用户身份认证等需求。

人脸识别目前支持在线和离线两种接入方式，人脸库支持通过管理控制台和Open API两种方式进行管理，本文主要介绍如何通过控制台管理人脸库，API部分请参阅人脸库管理API参考。

名词解释：

- 人脸库：人脸的集合。根据您自身的业务需求，可进行分库管理，提升人脸查找的准确率。例如，可以根据用户所处的地理位置、手机尾号等进行分库，在进行人脸查找时，按用户的地理位置或手机尾号，定位到指定的人脸库ID，在该库中进行查找。
- 人脸：一个用户的人脸信息，包括人脸照片、人脸ID、性别等。

人脸库结构：



产品功能

- **人脸检测**

  检测图片中的人脸并标记出位置信息，展示人脸属性信息，如年龄、性别等。

- **人脸比对**

  比对两张图片中人脸的相似度，并返回相似度分值。

- **人脸库管理**

  要完成1：N识别，首先需要构建一个人脸库，用于存放所有人脸特征。

- **人脸查找**

  也称为1：N识别，在指定的某一个或多个人脸库中，找到最相似的人脸。

- **活体检测**

  对人脸识别采集到的照片进行人脸活体检测，检测照片中的人是否是真人。

应用场景

您可以将人脸识别能力应用于机器人，满足在不同场景下迎宾问候、会员识别等需求。您可以将用户的人脸信息添加到人脸库中，并通过ID与您的客户系统打通。

- **迎宾接待**

  基于人脸检测和查找结果，实现个性化迎宾接待服务。

- **VIP管理**

  会员到店无需出示会员凭证，只要刷脸即可完成会员身份验证，实现「无卡化」身份确认和人流统计。

- **个性化服务**

  根据用户的不同身份，结合平台的自定义技能等能力，为不同用户提供个性化服务。

- **身份认证**

  通过公安身份图像与真人图像比对，判断用户是否为本人，从而完成在线用户身份核真检验。

- **人脸签到**

活动开始前录入人脸图片，活动当天即可通过刷脸进行签到，提高签到效率。

- **人脸跟踪**

  基于人脸检测信息实现智能人脸跟踪，提升交互智能化水平。

- **人脸支付**

  将人脸与用户的支付渠道绑定，支付阶段即可刷脸付款，无需出示银行卡、手机等，提高支付效率。

- **用户画像**

  基于人脸ID打通您自身的客户系统，建立用户的画像标签体系，实现精准营销、商品推荐等能力。

人脸库管理

新增人脸库

1. 在“**人脸库列表**”页面，点击列表上方的“**添加人脸库**”按钮。

2. 根据页面提示填写人脸库名称和人脸库 ID，完成后单击“**确定**”，即可创建一个人脸库。

   - 人脸库名称：可修改，不可重复。
   - 人脸库ID：唯一标识符，不可修改，不可重复。

编辑人脸库

1. 在“**人脸库列表**”页面，鼠标移动到要编辑的列表项上方，点击人脸库名称后的“**编辑**”图标，即可对人脸库名称进行编辑。

删除人脸库

1. 方法一：单个删除。在“**人脸库列表**”页面，找到要删除的人脸库，点击列表操作项中的“**删除**”按钮，即可删除该人脸库。

2. 方法二：批量删除。在“**人脸库列表**”页面，勾选多个列表项，点击列表上方的“**删除**”按钮，即可批量删除选中的人脸库。

> 注：若一个人脸同时存在于多个人脸库中，删除该人脸库，不会删除其他人脸库中该人脸的信息。

查找人脸库

1. 按关键词模糊查找：人脸库名称、人脸库ID。

人脸管理

新增人脸

1. 在“**人脸库列表**”页面，点击列表项中的“**人脸库名称**”或“**详情**”按钮，进入该人脸库。

2. 点击列表上方的“**添加人脸**”按钮。

3. 按照弹框中的提示信息填写对应数据。
   - 人脸照片：上传2M以下清晰人脸照片，照片中请勿包含多张人脸。仅支持png、jpg、jpeg、bmp格式。
   - 人脸ID：用于标识唯一的人脸照片，您也可以通过该字段关联您自己的用户／会员系统。
   - 姓名：用户姓名，若称谓未填写，则用姓名打招呼。

- 称谓：用于机器人欢迎语，如“王总（称谓）你好，欢迎光临北京。”
- 性别：男或女。
- 备注：添加备注信息。

4. 点击“**确定**”，新建成功。

编辑人脸

1. 在“**人脸列表**”页面，找到要编辑的人脸，点击列表操作项中的“**编辑**”按钮，即可编辑该人脸信息。编辑操作同**新增人脸**。

删除人脸

1. 方法一：单个删除。在“**人脸列表**”页面，找到要删除的人脸，点击列表操作项中的“**删除**”按钮，即可删除该人脸。

2. 方法二：批量删除。在“**人脸列表**”页面，勾选多个列表项，点击列表上方的“**删除**”按钮，即可批量删除选中的人脸。

查找人脸

1. 按关键词模糊查找：姓名、人脸ID、称谓、备注。

效果优化

日志标注

为了让机器人更好地理解用户的问题，平台提供了日志标注功能。从用户和机器人真实的对话日志中，挖掘出用户的多样问法，通过人工标注再训练，持续提升机器人的语义泛化效果。

> 日志标注目前仅支持对问答库的问题进行标注再训练，暂不支持自定义技能的问题。

日志标注可分为几种使用场景：

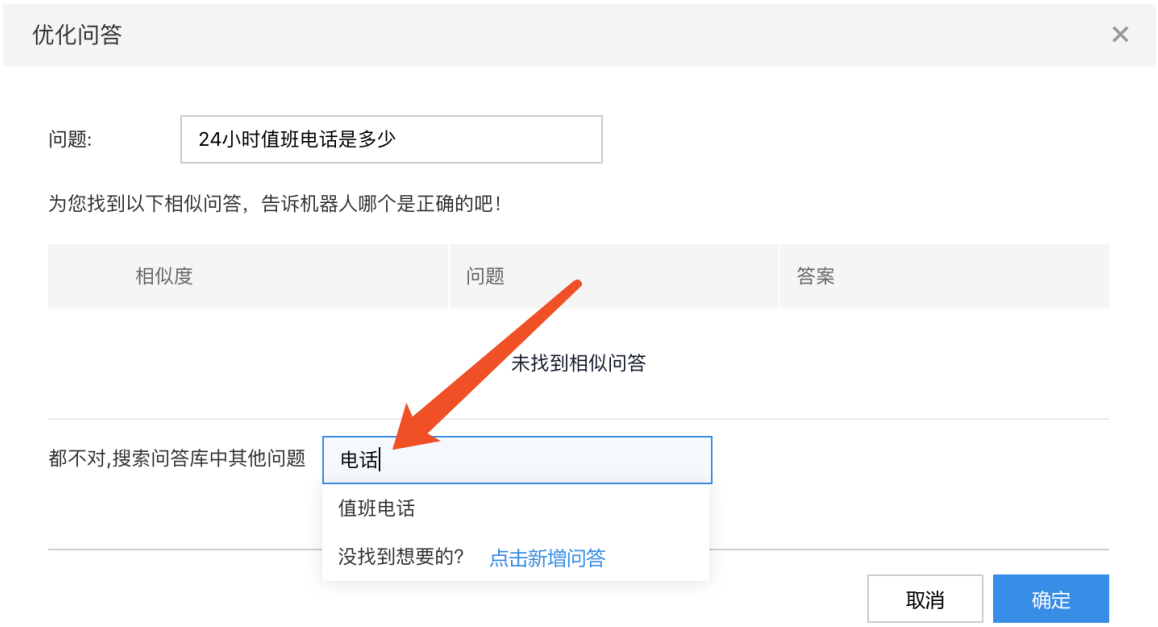
1. 对于问答库中不存在的问题，可以在标注过程中往问答库补充新知识。
2. 对于问答库中存在，但机器人未能回答的问题，可以将该问法补充到已有知识，下次机器人就能回复该问题。
3. 对于问答库命中的问题，也可以将该问法补充到已有知识，提高问答库的语义泛化能力。

单条优化

1. 登录ABC Robot平台管理控制台，进入一个项目，左侧导航栏点击“**效果优化>日志标注**”，进入“**日志标注**”页面。

2. 在列表中找到相优化的问题，点击列表项操作区的“**单条优化**”按钮，系统会为您推荐问答库中最相似的top3问题，选择其中一项，点击“**确定**”。

3. 如果系统推荐的相似问答没有您想要的，您可以在搜索框中输入关键词，搜索问答库中已经添加的知识。



4. 选择其中正确的问题，点击“**确认**”按钮。

5. 如果这是一个新知识，您可以点击“**点击新增问答**”按钮，创建新知识并保存。

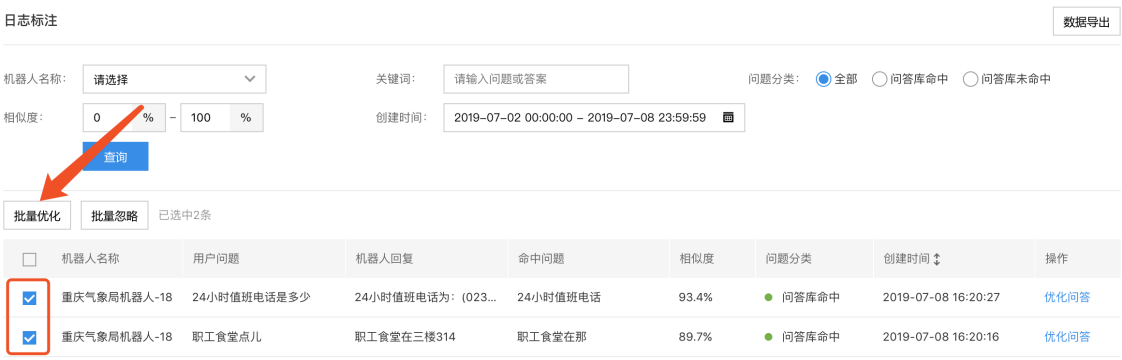
批量优化

如果用户问题和命中问题在语义上属于同一个问题，您可以直接勾选对应的选项，点击“**批量优化**”按钮，即可批量优化选中

项。批量优化需满足以下条件：

1. 命中问题在问答库中没有被删除

2. 批量优化的问题，问题分类必须都为“问答库命中”。



批量忽略

对于不需要优化的问题，您可以勾选对应的选项，点击“**批量忽略**”按钮，选中项就不会展示在列表页中。

数据统计

数据统计简介

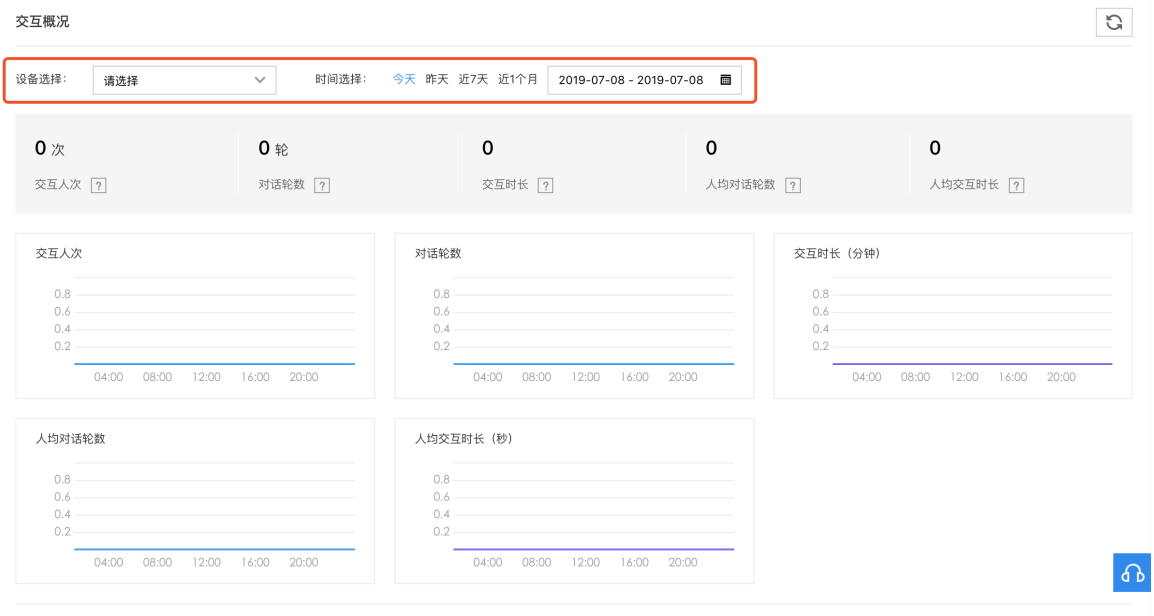
数据统计功能提供多维度的数据指标和图表信息，可以帮助开发者掌握机器人的核心数据指标和走势，为开发者进一步开发和优化效果提供有效的数据支持。ABC Robot为开发者提供的数据统计主要包括交互概况、知识分析和历史对话记录三种维度的数据，支持按照不同的设备和时间范围进行查询。

功能	介绍
交互概况	展示机器人与用户的交互情况，主要数据包括交互人次、对话轮数、交互时长、人均对话轮数、人均交互时长。
知识分析	展示知识的使用情况、知识库健康度、热门／冷门知识的分布情况，进而指导开发者有针对性地优化知识库和自定义技能。主要数据包括答案来源分布、问答库问答总数、问答库有效问答占比、问答库和自定义技能的命中明细。
历史对话记录	展示用户与机器人对话的内容明细。

交互概况

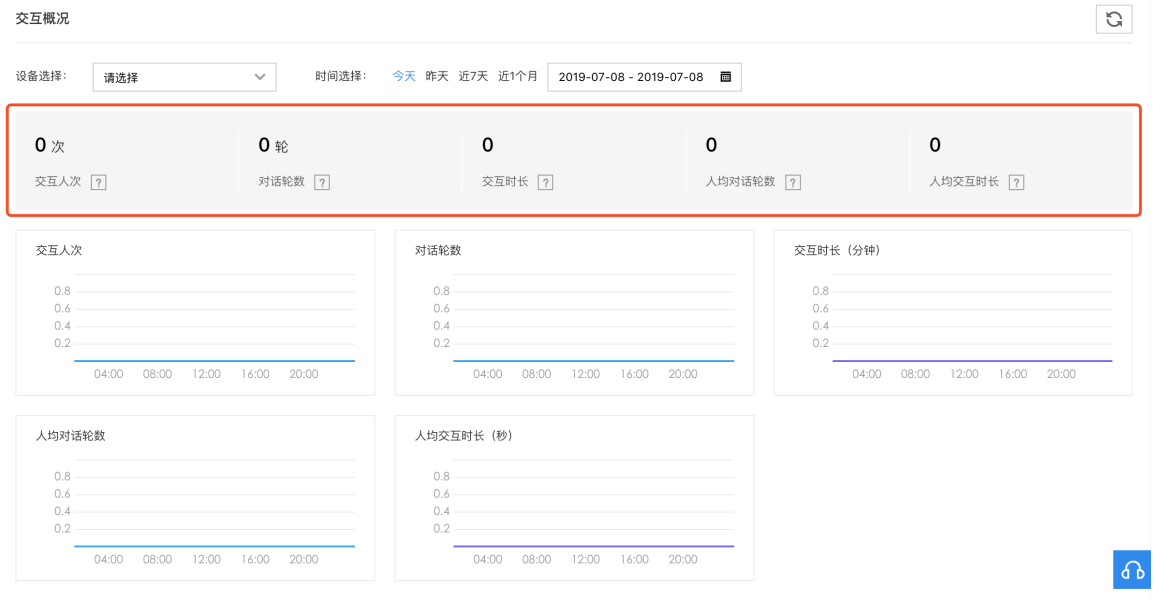
条件筛选

支持筛选某个设备后查看指定设备的数据，同时支持时间选择，如今天、昨天、近7天、近1个月，用户也可以自定义时间段进行筛选。



数据概览

包括交互人次、对话轮数、交互时长、人均对话轮数、人均交互时长。

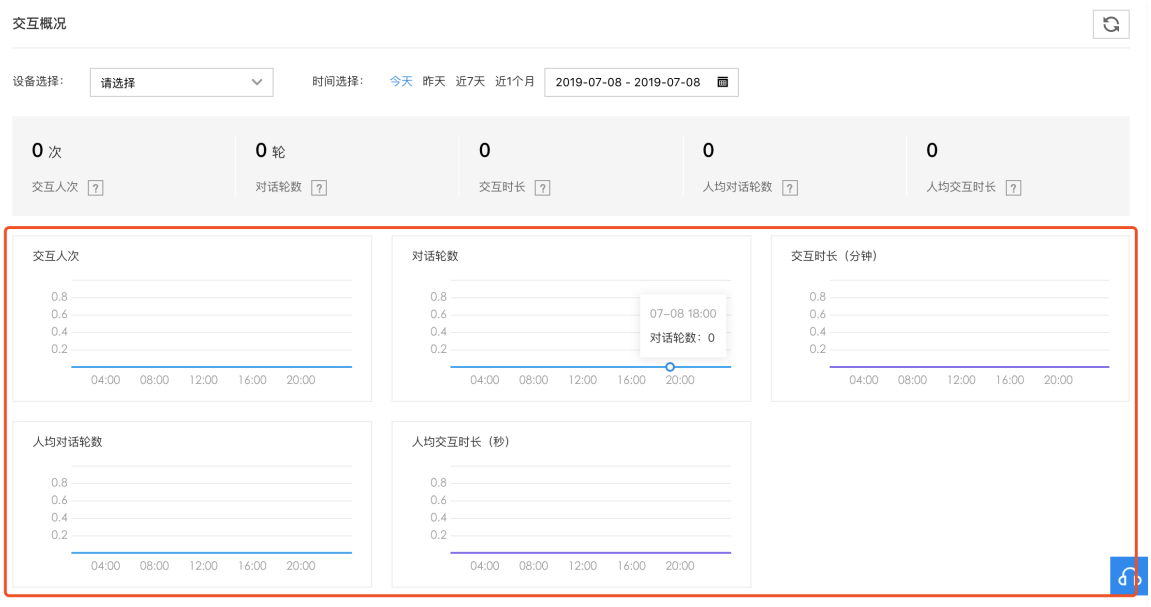


其中，各指标的解释如下：

- 交互人次：在指定时间、指定设备范围内（下同），服务的总人次，一个session为一人次。
- 对话轮数：机器人的总对话量，一个用户对话请求为一轮。通常一个session中包含多轮对话。
- 交互时长：机器人跟用户交互的总时长，各session结束时间减开始时间的和
- 人均对话轮数：平均每个用户的对话轮次，一个用户对话请求为一轮。用来衡量用户的粘性，轮数越大，表明用户对话的粘性越强。
- 人均交互时长：每个session的平均时长。

数据趋势

在指定时间、指定设备范围内，交互人次、对话轮数、交互时长、人均对话轮数、人均交互时长的变化趋势。



数据明细

在指定时间、指定设备范围内，按日期或小时（时间选择跨度为1天时）的维度展示机器人的交互数据明细。包括交互人次、对话轮数、交互时长、人均对话轮数、人均交互时长。

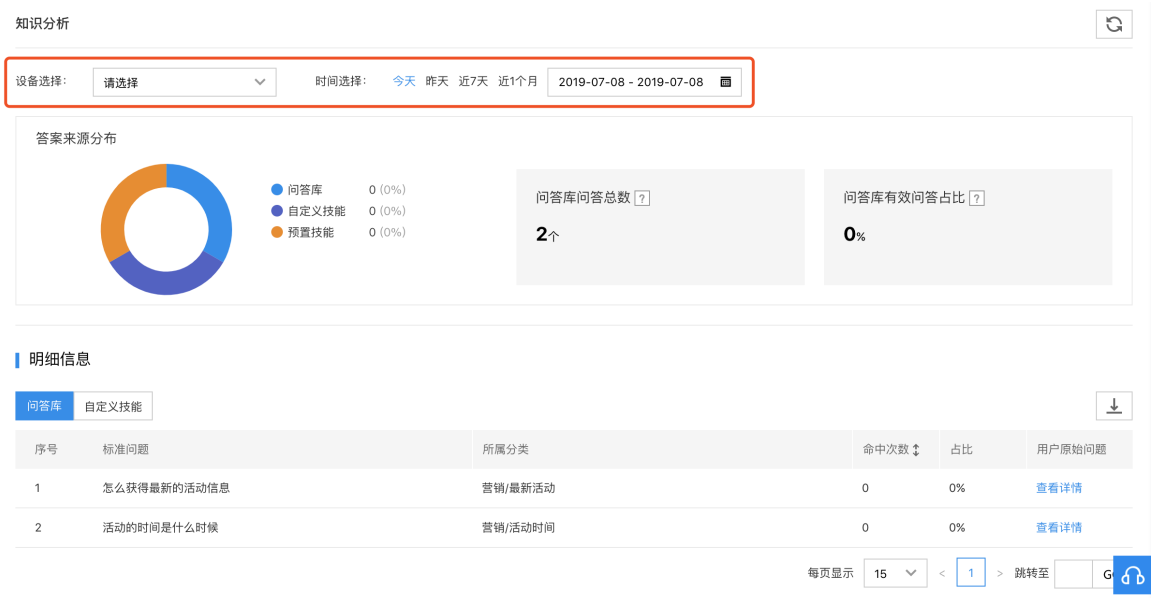
同时，支持以Excel形式导出数据明细。

时间	交互人次	对话轮数	交互时长（分钟）	人均对话轮数	人均交互时长（秒）
07-08 23:00	0	0	0	0	0
07-08 22:00	0	0	0	0	0
07-08 21:00	0	0	0	0	0
07-08 20:00	0	0	0	0	0
07-08 19:00	0	0	0	0	0
07-08 18:00	0	0	0	0	0
07-08 17:00	0	0	0	0	0
07-08 16:00	0	0	0	0	0
07-08 15:00	0	0	0	0	0
07-08 14:00	0	0	0	0	0
07-08 13:00	0	0	0	0	0
07-08 12:00	0	0	0	0	0
07-08 11:00	0	0	0	0	0

知识分析

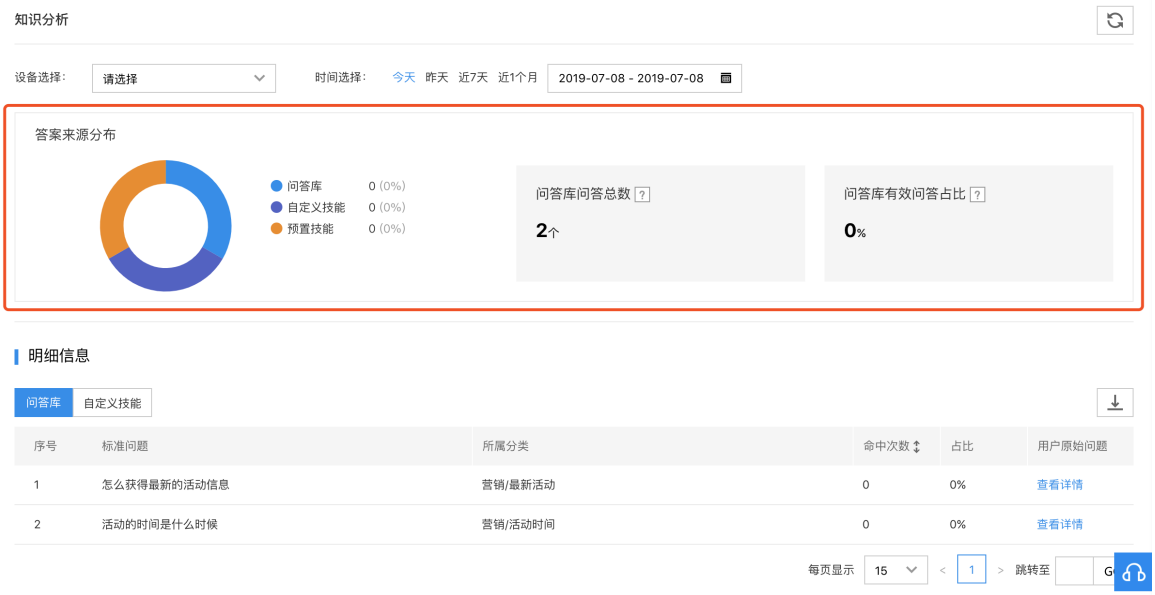
条件筛选

支持筛选某个设备后查看指定设备的数据，同时支持时间选择，如今天、昨天、近7天、近1个月，用户也可以自定义时间段进行筛选。



数据详情

在指定时间、指定设备范围内，知识的有效性、使用情况分析，帮助运营人员判断问答库构建的健康程度，针对性进行优化。



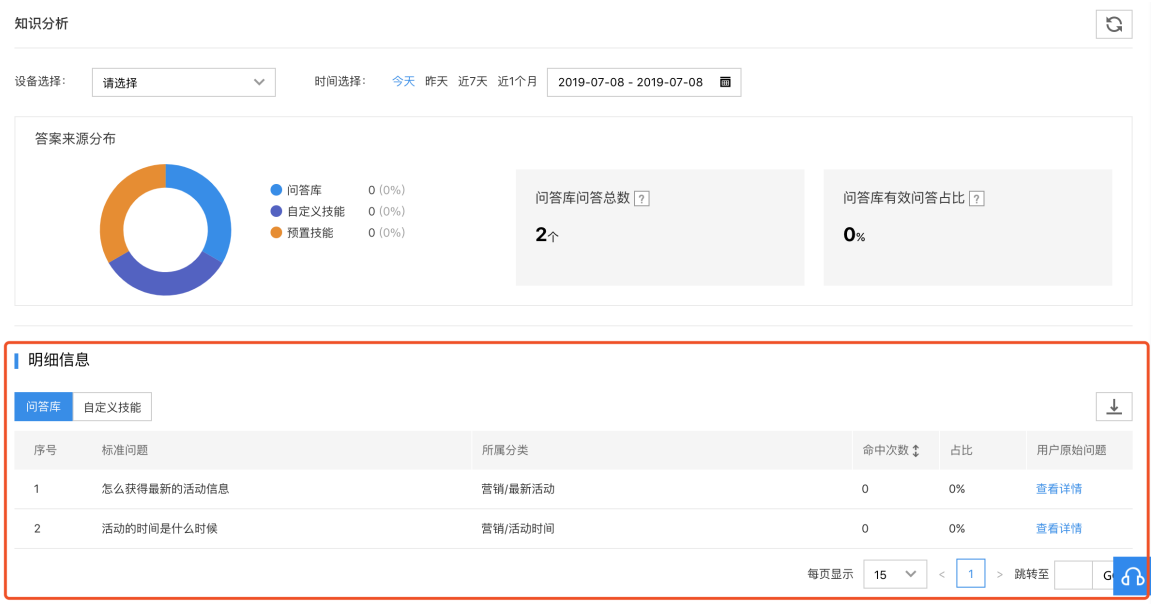
其中，各指标的解释如下：

- 答案来源分布：机器人的答案中，各个来源所占的比例。
- 问答库问答总数：当前问答库中标准问题的数量。
- 问答库有效问答占比：指定时间范围内有命中的标准问数量／问答库问答总数。反映问答库中有多少问答是被用户真实提问到的，用来衡量问答库构建的有效性和健康度。

数据明细

在指定时间、指定设备范围内，问答库中各个标准问、自定义技能中各个意图的命中明细，支持按命中次数排序。反映标准问题和意图的热度情况。

同时，支持以Excel形式导出数据明细。



1. **问答库明细**主要字段包括：
    - 标准问题：问答库中的所有标准问题。
    - 所属分类：该标准问题在问答库中对应的分类。
    - 命中次数：该标准问题及其相似问法被命中的次数。
    - 占比：命中次数／所有标注问题命中次数之和。
    - 用户原始问题：命中该标准问题及其相似问的用户原始问法。
2. **自定义技能明细**主要字段包括：
    - 意图名称：自定义技能中定义的意图名称。
    - 所属自定义技能：该意图对应的自定义技能。
    - 命中次数：同问答库。
    - 占比：同问答库。
    - 用户原始问题：同问答库。

历史对话记录

通过历史对话记录功能，您可以查看人机交互历史记录，了解机器人的对话效果。

操作步骤

1. 登录ABC Robot平台管理控制台，选择一个项目，左侧导航栏点击“**数据统计>历史对话记录**”，进入“**历史对话记录**”页面。

2. 组合筛选：根据下面的查询字段进行组合筛选。

参数名称	描述
问答分类	1. 默认全部分类 ；2. 问答库命中：用户问题命中问答库中的业务知识；3. 问答库未命中：用户问题未命中问答库中的业务知识，可能命中预置技能、自定义技能、兜底金句等。
机器人名称	您可以选择该项目下一台或多台机器人来过滤对应的对话记录。
创建时间	支持按分钟级时间段查询历史对话记录。
相似度	命中问答库的问题有相似度字段，相似度为用户问题和问答库中已有问题的语义相似度。
关键词	支持按关键词搜索历史对话记录中包含该关键词的问题或回复。

3. 数据导出：
   - 默认导出全部数据。
   - 根据筛选条件，导出满足条件的数据。

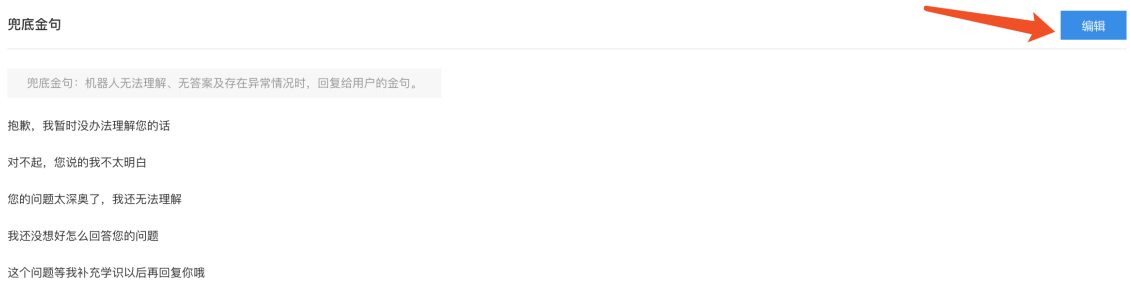
项目配置

兜底金句

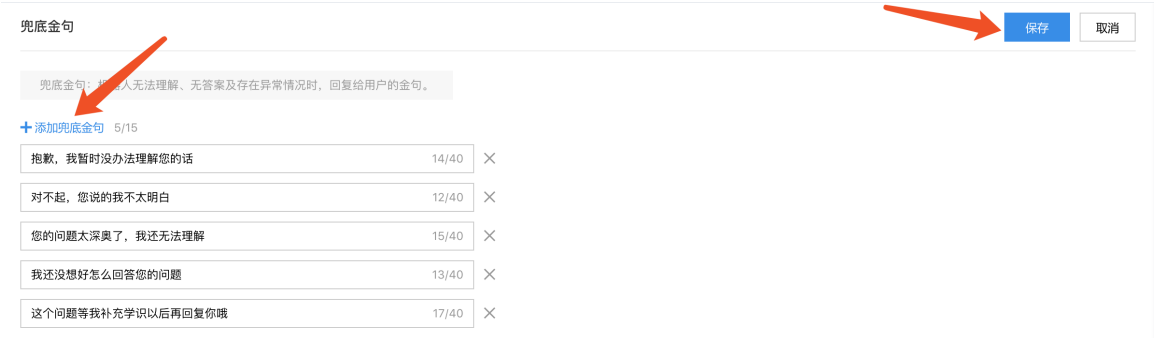
兜底金句是机器人无法理解、无答案及存在异常情况时，回复给用户的话术。最多可添加15条兜底金句，系统会进行随机回复，提升回复多样性。

操作步骤

1. 登录ABC Robot平台管理控制台，进入一个项目，左侧导航栏点击“**项目配置>兜底金句**”，进入“**兜底金句**”页面。
2. 在“**兜底金句**”页面，点击“**编辑**”按钮，进入编辑页面。



3. 点击“**添加兜底金句**”按钮，在输入框中输入兜底金句后，点击“**保存**”按钮，编辑成功。



权限管理

您可以将项目的权限开放给其它用户，进行多人协作。权限授予通过百度云用户ID进行唯一用户标识，您可以在“**页面右上角头像>用户中心>用户ID**”，获取用户ID。

角色权限定义

创建者（每个项目仅一个，创建该项目的账号自动拥有该项目的创建者权限）

- 拥有项目下所有页面的查看、新建、编辑、删除权限
- 创建者不可被删除
- 创建者的权限不可修改

编辑

- 拥有该项目下所有页面（**除权限管理页面**）的查看、新建、编辑、删除权限
- 无删除项目权限
- 权限管理页面的查看权限，无新建、编辑、删除权限

只读

- 拥有该项目下所有页面的查看权限，无新建、编辑、删除权限

添加用户

1. 登录ABC Robot平台管理控制台，进入一个项目，左侧导航栏点击“**权限管理**”，进入“**权限管理**”页面。
2. 点击列表上方的“**新增**”按钮。

3. 按照弹框提示填写信息。

字段名称	说明
用户ID	被授权用户的百度云用户ID，请在“**页面右上角头像>用户中心>用户ID**”获取。
姓名	被授权用户的姓名。
权限	包括编辑和只读。
备注	备注用户的公司、身份等信息。

4. 点击“**确定**”，添加完成。

编辑用户

1. 在“**权限管理**”页面，找到要编辑的用户，点击列表操作项中的“**编辑**”按钮，即可编辑该用户信息。
2. 编辑操作同“**添加用户**”。

> 注：用户ID不可编辑；创建者的权限不可编辑。

删除用户

1. 方法一：单个删除。在“**权限管理**”页面，找到要删除的用户，点击列表操作项中的“**删除**”按钮，即可删除该用户。

2. 方法二：批量删除。在“**权限管理**”页面，勾选多个列表项，点击列表上方的“**删除**”按钮，即可批量删除选中的用户。

> 注：创建者不能被删除。

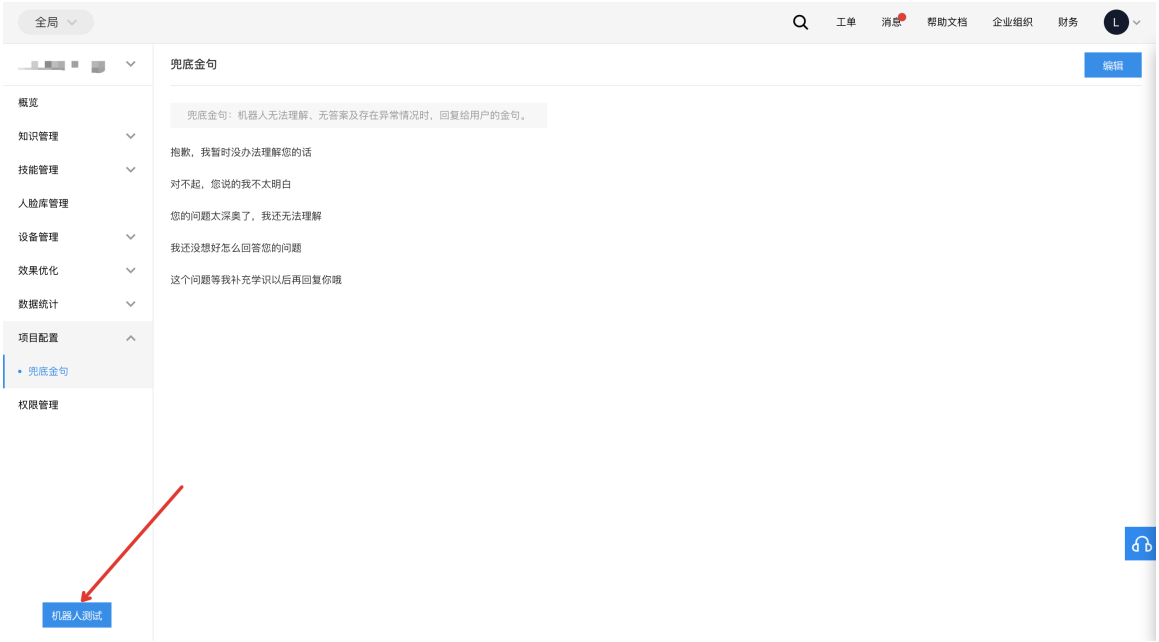
测试验证

ABC Robot平台提供轻量化的测试验证工具，帮助开发者低成本地完成大多数场景的功能验证。通过模拟用户的真实对话请求，开发者可以对全局或单个模块进行调试，并查看系统的返回结果。

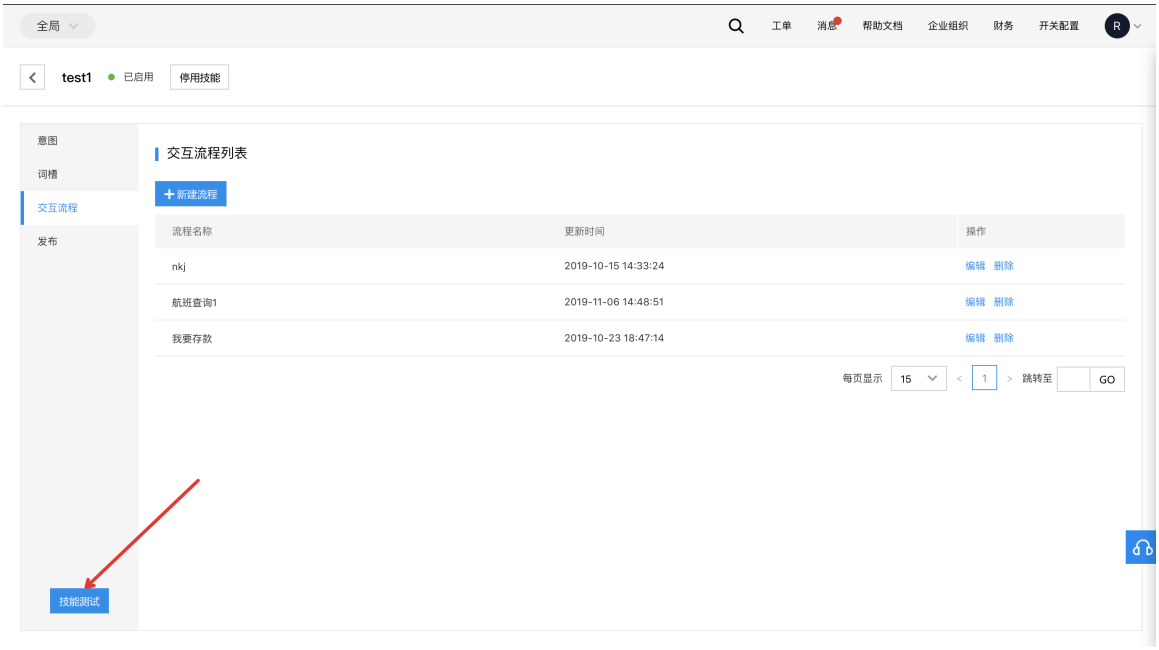
> 注：部分能力（如运动指令等）无法在网页上模拟，请通过真实设备测试。

操作步骤

1. 登录ABC Robot平台管理控制台，进入一个项目，可通过以下两个入口进入“**测试验证**”功能。
   - 入口一：左侧导航栏点击“**机器人测试**”。



- 入口二：进入**“图形化技能”**或**“填槽型技能”**，左侧导航栏点击“**技能测试**”。



2. 抽屉浮层中可选择测试范围、测试设备、测试技能或流程、测试环境，选中后会生效该配置项。选择完毕后关闭浮层，您也可点击**“测试范围”**按钮，重新选择配置项。

> 测试技能时，建议选择具体的技能或交互流程。

3. 在对话框中输入测试query，进行测试验证。

4. 结果中会展示指令中各类型的答案，以及Debug信息。Debug卡片可在抽屉右上角关闭，卡片中展示具体的过程信息，更多内容请点击下方的**“查看data详情”**按钮。

运动管理

导览讲解

导览讲解功能介绍

导览讲解功能是面向展厅场景，让机器人为参观者提供讲解服务。项目管理者可以通过编排导览线路、配置讲解内容的方式，替代部分讲解员的重复性较高的导览讲解工作，同时提升展厅科技形象。

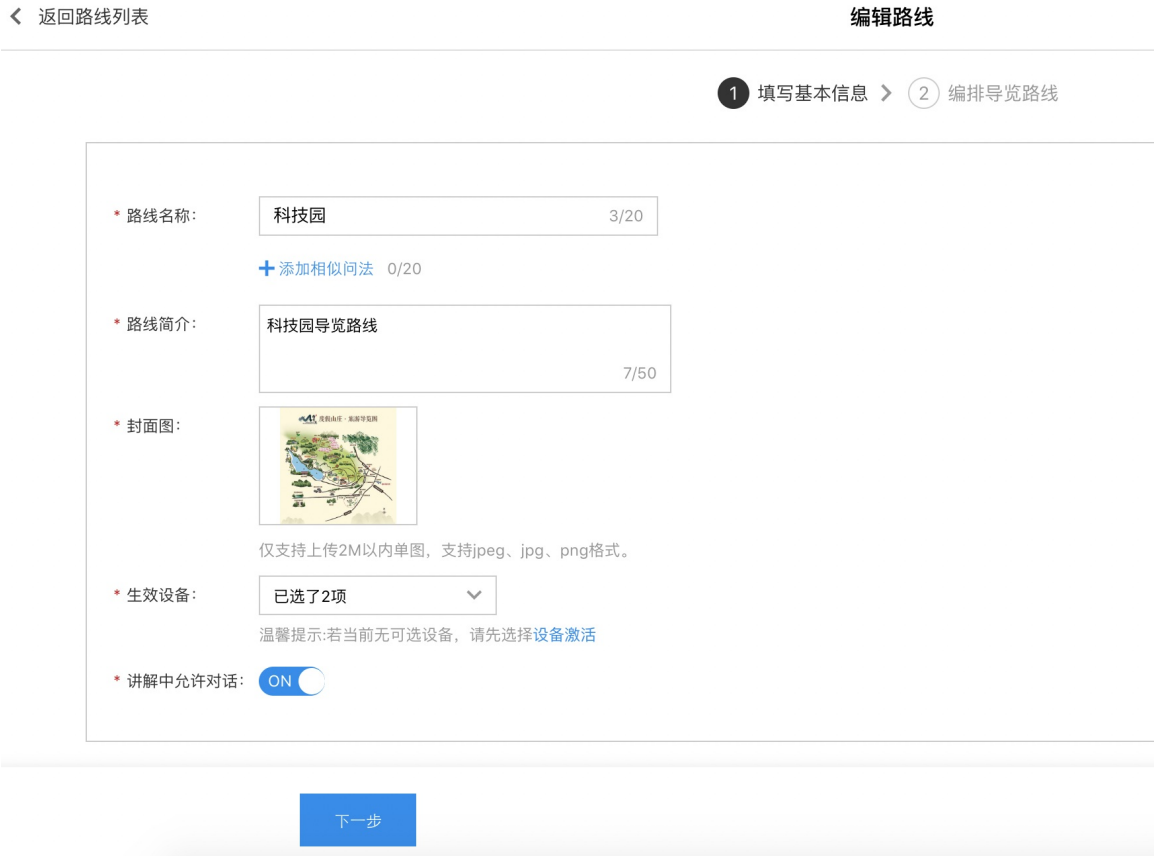
触发方式

1. 开始导览：用户可以在机器人端通过“开始导览”，“开始参观”，“我要参观”，“带我参观这里”，“带我参观科技园”等话术触发导览讲解功能。触发后，机器人播报“好的，请跟我来”，随即前往起始点，按已经编排好的路线进行导览讲解。
2. 继续导览：在机器人导览过程中，如果因为交互导览导览流程暂停，可以通过“继续导览”等话术，使机器人继续进行导览流程。
3. 下一个导览点：在机器人导览过程中，可以通过“下一个点”，“下一个导览点”等话术，让机器人结束当前点的讲解，直接到下一个点进行讲解。
4. 结束导览：在机器人导览过程中，可以通过“结束导览”，“停止导览”等话术，让机器人退出导览状态。

添加路线

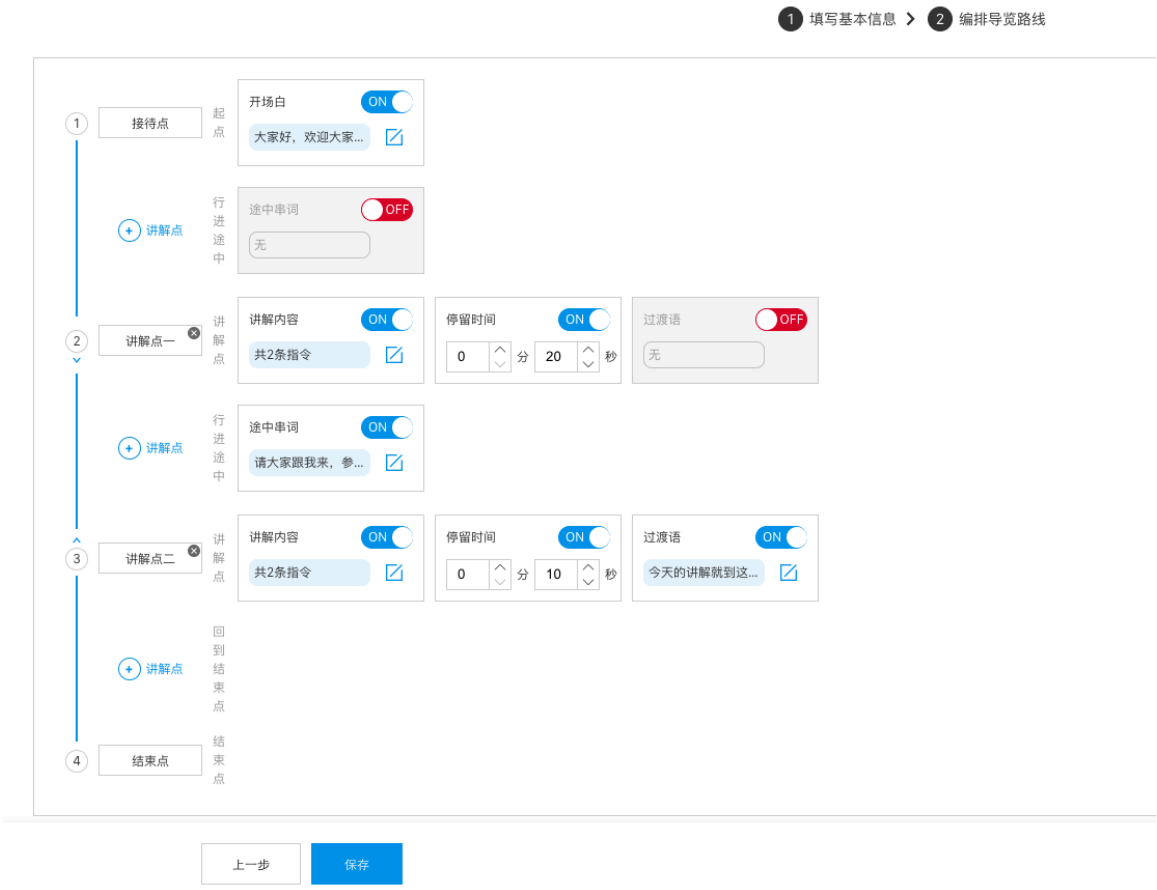
1. 登录ABC Robot平台管理控制台，进入一个项目，左侧导航栏点击“**运动管理>导览讲解**”，进入“**导览讲解**”页面。
2. 点击列表上方的“**新增**”按钮，进入路线编辑流程。

3. 在路线新增页面，填写路线基本信息。



字段名称	说明
路线名称	可添加路线名称的相似问法，用于直接触发对应导览路线，比如“带我参观科技园”，科技园即为线路名称或线路名称相似问法
路线简介	用于路线描述，用于路线说明
封面图	支持jpg、jpeg、png格式，大小5m以内
生效设备	多选下拉菜单，选中后，该路线只对选中设备生效
讲解中允许对话	关闭后，导览过程中机器人不响应用户的对话，只进行讲解；开启后，机器人在导览过程中会响应用户的对话，但不会打断当前的导览流程，在响应用户的对话后，会继续进行导览。

4. 在路线新增页面，填写完基本信息后，点击下一步，编排导览路线信息。



4.1 导览路线第一个点默认为“接待点”，不可以删除或调整顺序，接待点可以添加开场白，开场白为多行文本输入框。

4.2 导览路线最后一个点为“结束点”，不可以删除或调整顺序。

4.3 途中串词用于两个讲解点之间的过渡串讲。

4.4 点击“添加讲解点”，出现添加讲解点弹框。输入讲解点名称，点击确定，即可完成讲解点添加。注意讲解点名称必须在机器人地图上存在。

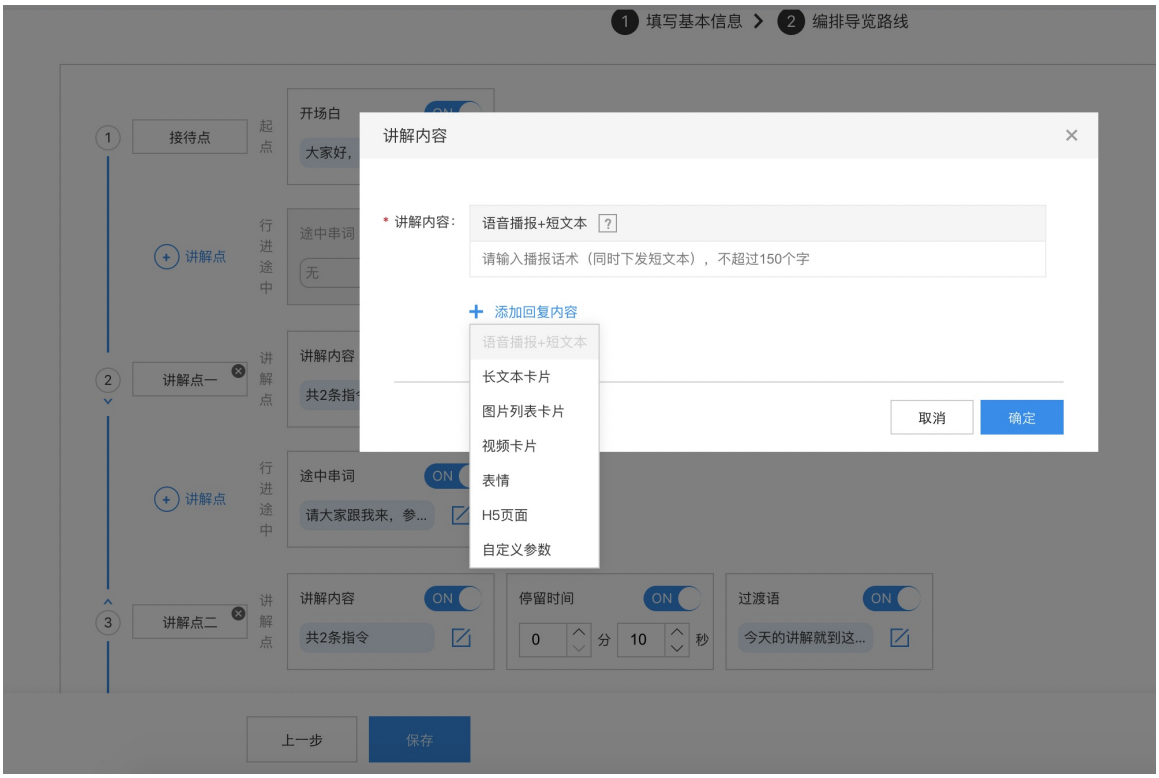
4.5 添加讲解点后，路线上出现对应的讲解点，如下图所示：

4.5.1 路线左侧可对讲解点位置上下移动（移动时该点对应的内容也移动），以及删除讲解点。

4.5.2 路线右侧为机器人在该讲解点时，对应的交互内容，包括：讲解内容、停留时间、过渡语。每张卡片都可设置开关状

态。当卡片开关状态为开时，必须填写卡片的内容，否则无法保存。

（1）讲解内容：当未编辑时，提示文字显示“无”，编辑后，提示文字显示“共X个指令”；点击讲解内容编辑按钮，出现如下弹框：



讲解内容支持短文本、长文本、图片、视频、H5页面等。

（2）停用时间：机器人在该讲解点讲解完毕后，需要停留的时间。

（3）过渡语：当未编辑时，提示文字显示“无”，编辑后，提示文字显示对应的文本内容。

5. 点击“**保存**”，添加完成。如果需要修改基本信息，可点击“**上一步**”，返回基本信息页面进行修改。
6. 新增路线后，需要重启该路线关联的机器人app，该路线数据才会被加载到机器人端。

编辑路线

1. 在“**运动管理>导览讲解**”页面，找到要编辑的路线，点击列表操作项中的“**编辑**”按钮，即可编辑路线信息。
2. 编辑操作同“**添加路线**”。
3. 编辑路线后，需要重启该路线关联的机器人app，该路线的更新数据才会被加载到机器人端。

停用路线

1. 在“**运动管理>导览讲解**”页面，找到要停用的路线，点击列表操作项中的“**停用**”按钮，即可停用路线。
2. 停用后，需要重启该路线关联的机器人app，机器人端才会清除该路线数据缓存。

删除路线

1. 方法一：单个删除。在“**运动管理>导览讲解**”页面，找到要删除的路线，点击列表操作项中的“**删除**”按钮，即可删除该路线。

2. 方法二：批量删除。在“**运动管理>导览讲解**”页面，勾选多个列表项，点击列表上方的“**删除**”按钮，即可批量删除选中的路线。

3. 删除路线后，重启该路线关联的机器人app，机器人端才会清除该路线数据缓存。

问路导航

问路导航功能介绍

通过平台提供的问路导航功能，开发者通过简单配置，即可完成导航模块的定制，满足不同设备的位置问询和导航需求。

触发方式

场景	触发话术
直接导航	带我去XXX、领我们去XXX等话术
位置问询	XXX在哪儿、XXX在什么地方
功能支持问询	你能导航吗
位置范围问询	你能查询什么地点，你能导航哪些地方

添加地点

1. 登录ABC Robot平台管理控制台，进入一个项目，左侧导航栏点击“**运动管理>问路导航**”，进入“**问路导航**”页面。
2. 点击列表上方的“**新增地点**”按钮，打开地点新增弹框。

字段名称	说明
地点名称	和机器人扫图后，在地图上创建的点的名称一致
同义词	用户在问路导航过程中，询问该地点时，可能用到的地点同义词，同义词无需和地图上点的名称一致
问路回复	问路导航命中该地点时回复的话术
生效设备	选择该地点生效的设备标签，选中后，该地点只对选中设备标签下的设备生效
是否支持导航	默认打开，如果关闭，则只播报问路回复内容，机器人不会触发导航动作
类型	可选类型为“普通点”，“充电点”，“迎宾点”；每个设备只能有一个充电点，一个迎宾点，但可以有多个普通点。

> 注：当机器人导航到某一个地点后，在指定的时间内没有发生任何交互，则机器人会自动回到迎宾点；当下发充电指令时，机器人会自动回到充电点。
> 注：如果修改了点的类型，则需要重启对应的机器人app，点类型才能生效。

3. 点击“**确定**”，添加完成。

编辑地点

1. 在“**运动管理>问路导航**”页面，找到要编辑的地点，点击列表操作项中的“**编辑**”按钮，即可编辑地点信息。
2. 编辑操作同“**新增地点**”。

删除地点

1. 方法一：单个删除。在“**运动管理>问路导航**”页面，找到要删除的地点，点击列表操作项中的“**删除**”按钮，即可删除该地点。

2. 方法二：批量删除。在“**运动管理>问路导航**”页面，勾选多个地点列表项，点击列表上方的“**删除**”按钮，即可批量删除选中的地点。

智能推荐

首页推荐

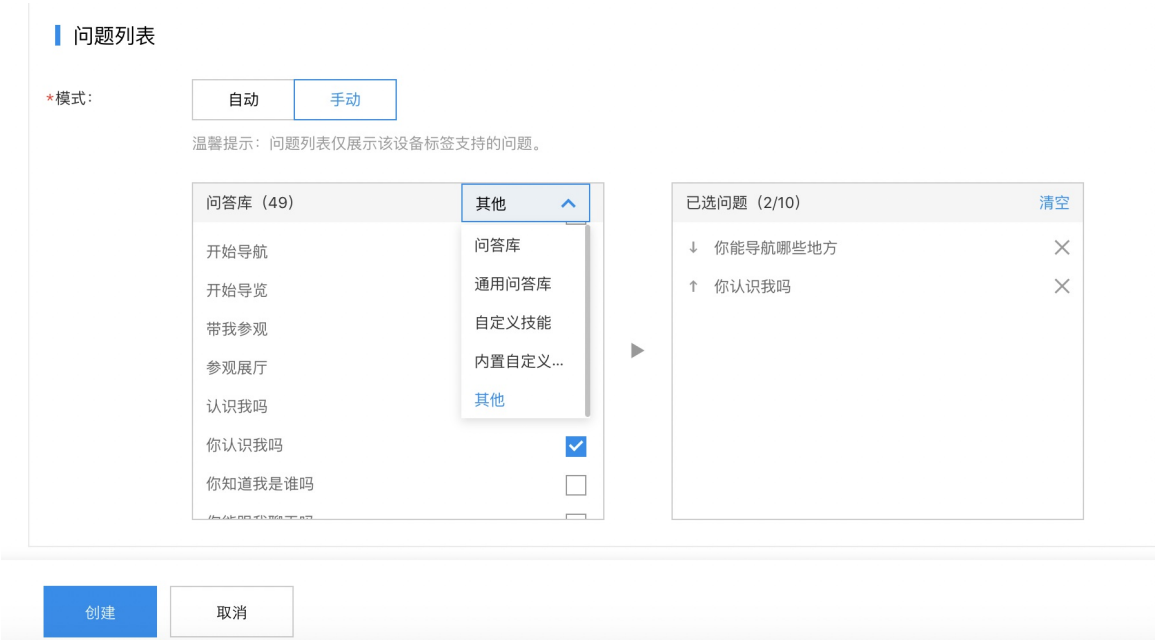
在首次交互时，告知用户机器人支持的能力，引导用户提问系统支持的问题，降低无结果比例、提升用户活跃度。

创建首页推荐

1. 登录ABC Robot平台管理控制台。
2. 在“**智能推荐>首页推荐**”页面，点击“**新建推荐**”按钮，按照弹框提示填写信息。

> 创建项目时会自动创建一个默认的首页推荐数据，用户可以修改或删除该数据。

- 基本信息
  - 推荐名称信息：用于检索和区分推荐列表中的数据。
  - 生效设备标签：如果绑定标签，则该推荐数据只对与该标签关联的设备生效。否则，对所有设备生效。
- 问题列表
  - 自动模式：该模式下，平台统计最近一个月内用户问的最多的问答库数据，自动生成推荐数据。
  - 手动模式：该模式下，用户可以通过下拉框选择候选数据来源，目前候选列表类型支持：问答库、通用问答库、自定义技能、内置自定义技能、其它(预置技能、导航、导览、熟人识别等技能)。



点击候选列表问题，该问题会自动移动至“已选问题”列表。在“已选问题”展示框中，可以点击上下箭头，来调整推荐数据在机器人端的显示顺序。也可以点击“X”来取消已选数据。

3. 点击“**创建**”按钮，完成首页推荐信息创建。同一个标签下，只能创建一个首页推荐数据。

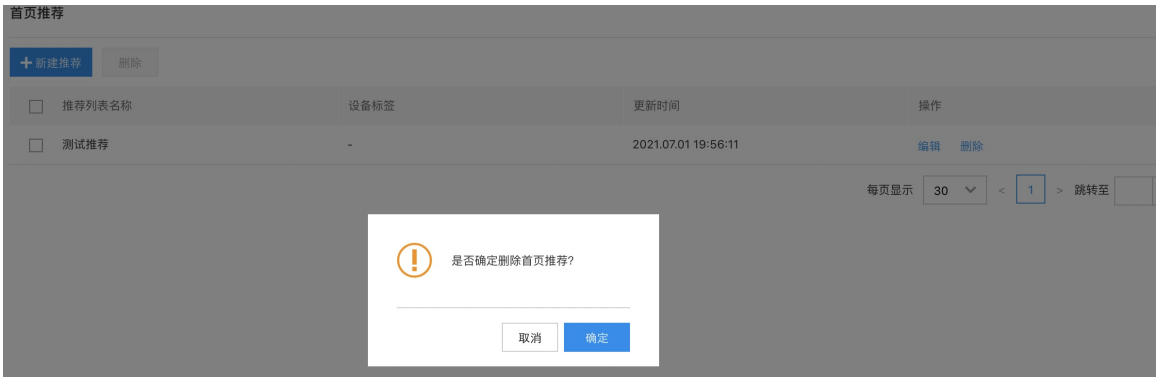
编辑首页推荐

1. 在“**智能推荐>首页推荐**”页面，找到要编辑的首页推荐数据，点击“**编辑**”按钮。

2. 按照弹框提示填写信息，操作同“**创建首页推荐**”。

删除首页推荐

1. 单条删除：在“**智能推荐>首页推荐**”页面，找到要删除的列表项，点击“**删除**”按钮，在弹框中点击“**确定**”即可删除推荐数据。



2. 批量删除：在“**智能推荐>首页推荐**”页面，勾选多个列表项，点击列表上方的“**删除**”按钮，即可批量删除选中的推荐数据。

> 注意：修改推荐数据后，重启标签关联的机器人端APP，推荐数据即可生效。

端云交互协议

这套协议是客户端和ABC Robot云端的通信协议，客户端借助此协议，实现端云交互，完成请求发送和返回值处理。我们将端云交互抽象成事件、指令和端状态的形式，客户端的请求由事件和端状态组合而成，云端处理客户端传输的事件和端状态，将处理结果以指令集的形式下发给机器人，客户端执行收到的指令集，完成交互逻辑。

ClientContext（端状态）

ClientContext（端状态）反映的是客户端发起请求时的状态，客户端把端状态告知云端，云端可以根据设备端当前状态做出不同的逻辑决策，从而下发不同的指令集。例如，当前设备端正在语音播报，客户端收音并发起请求时将语音播报状态带上，这样云端就可以智能选择是否要求客户端打断当前语音播报。

示例

```
{
  "clientContextV2":{
     "TTS_PLAYING": "true",
     "APP_VERSION": "2",
     "WAKEUP_TYPE": "easyTalk|wakeupWord|touchBody|touchScreen",
     "MANUAL_REPLY": "true|false",
     "DIRECTION_INVALID": "0",
     "NAV_RUNNING":"true|false",
     "GUIDE_RUNNING":"true|false",
     "CHAT_IN_GUIDING":"true|false",
     "KNOW_ME_RUNNING":"true|false",
  },
  "event": {
     ...
  }
}
```

参数说明

字段名	类型	是否必需	说明
TTS_PLAYING	string	否	客户端语音播报状态，取值为 "true" 和 "false"，表示是否在语音播报
APP_VERSION	string	否	客户端Base App版本，目前取值为 "2"
WAKEUP_TYPE	string	否	唤醒方式,取值为easyTalk、wakeupWord、touchBody、touchScreen
MANUAL_REPLY	string	否	人工回复是否开启, 取值为 "true" 和 "false"
DIRECTION_INVALID	string	否	语音角度，0表示角度内，其它表示角度外
NAV_RUNNING	string	否	是否处于导航状态，取值为 "true" 和 "false"
GUIDE_RUNNING	string	否	是否处于导览状态，取值为 "true" 和 "false"
CHAT_IN_GUIDING	string	否	导览状态下是否支持聊天，取值为 "true" 和 "false"。当用户通过“开始导览”等话术触发导览，请注意对话接口Playload中的chatInGuiding字段的值。在导览过程中，将chatInGuiding字段的值进行回传即可；如果CHAT_IN_GUIDING没有回chatInGuiding的值，那么服务端将以客户端传值为准进行业务逻辑判断
KNOW_ME_RUNNING	string	否	是否处于熟人识别状态，取值为 "true" 和 "false"。在项目支持熟人识别的情况下，用户通过“你认识我吗？”等话术触发熟人识别流程，在熟人识别过程中，需要将该字段的值设置为"true"，其他情况下可以不传该字段的值或传"false"

Event（事件）

我们把客户端和云端交互的请求数据组装成Event（事件）发给云端，常见的事件例如：人脸识别事件等。

TextInput事件

TextInput事件是以文本的方式请求ABC Robot对话接口，这个过程不包含ASR。

示例

```
{
  "header": {
     "namespace": "baidu.abcrobot.event.text_input",
     "name": "TextInput"
  },
  "payload": {
     "query": "今天天气怎么样"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
query	string	是	请求文本

FaceRecognition事件

人脸识别事件，整个人脸识别请求由两部分组成：人脸识别事件和人脸图片。

```
{
  "header": {
     "namespace": "baidu.abcrobot.event.face_recognition",
     "name": "FaceRecognition"
  },
  "payload": {
     "group":"face group id"
  }
}
```

字段名	类型	是否必需	说明
group	string	否	人脸库ID。若不填写，则云端会默认使用default人脸库进行人脸识别。

二进制图片附件

```
 --this-is-a-boundary
Content-Disposition: form-data; name="image"; filename="image"
Content-Type: application/octet-stream
{{binary image attachment}}
--this-is-a-boundary
```

FaceComparison事件

人脸1:1比对事件，整个人脸1:1比对请求由两部分组成：人脸1:1比对事件和两张人脸图片。

```
{
  "header": {
     "namespace": "baidu.abcrobot.event.face_recognition",
     "name": "FaceComparison"
  },
  "payload": {
     "imageTypes":[
        1,
        2
     ]
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
imageTypes	list of int	是	图片类型值，取值范围详见下面说明。

imageTypes参数说明

图片类型值	类型	是否必需
1	生活照	通常为手机、相机拍摄的人像图片、或从网络获取的人像图片等
2	身份证芯片照	二代身份证内置芯片中的人像照片
3	带水印证件照	一般为带水印的小图，如公安网小图
4	证件照片	如拍摄的身份证、工卡、护照、学生证等证件图片，注：需要确保人脸部分不可太小，通常为100px*100px

二进制图片附件

```
 --this-is-a-boundary
Content-Disposition: form-data; name="image"; filename="image"
Content-Type: application/octet-stream
{{binary image 1 attachment}}
--this-is-a-boundary

--this-is-a-boundary
Content-Disposition: form-data; name="image"; filename="image"
Content-Type: application/octet-stream
{{binary image 2 attachment}}
--this-is-a-boundary
```

FaceImageInput事件

人脸图片上传事件，人脸图片上传请求由两部分组成：人脸图片上传事件和人脸图片。

```
{
  "header": {
     "namespace": "baidu.abcrobot.event.image_input",
     "name": "FaceImageInput"
  },
  "payload": {}
}
```

二进制图片附件

```
 --this-is-a-boundary
Content-Disposition: form-data; name="image"; filename="image"
Content-Type: application/octet-stream
{{binary image attachment}}
--this-is-a-boundary
```

RobotConfig事件

客户端拉取配置事件，云端收到这个事件后，会将云端配置下发给客户端。

```
{
  "header": {
     "namespace": "baidu.abcrobot.event.system",
     "name": "RobotConfig"
  },
  "payload": {}
}
```

Directive（指令）

云端把下发给客户端的操作封装成指令的格式，客户端收到指令后，依据指令的namespace、name和payload做出相应的响应。比如，Text指令，客户端收到这个指令后显示指令里的文本内容。

header参数说明

字段名	类型	是否必需	说明
namesace	string	是	指令的命名空间
name	string	是	指令的名字
interactionId	string	是	客户端生成的，本次请求的交互ID
messageId	string	是	云端生成的消息ID

常规指令

Speak指令

语音播报指令，客户端将指令的文本内容转换成语音并播报。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.voice_output",
     "name": "Speak",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "content": "今天晴，天气不错！"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
content	string	是	用于语音播报的文本内容

Text指令

文本显示指令，一般用于显示语音播报对应的文本，长文本的显示请使用卡片显示中的TextCard指令。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.text_output",
     "name": "Text",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "content": "为您找到长城的介绍"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
content	string	是	用于显示的文本内容

Hints指令

提示信息指令，用于引导用户进行下一轮交互。如用户问“天气怎么样”，云端在返回天气答案的同时，会下发Hints指令，如“明天呢”、“天津呢”，引导用户进行多轮对话。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.hint",
     "name": "Hints",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "hints": [
        "我要办理值机"
     ]
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
hints	list of string	是	提示信息列表

> Text指令和Hints指令排版示例如下，其中“为您找到长城的介绍”为Text指令，“我要办理值机”为Hints指令。

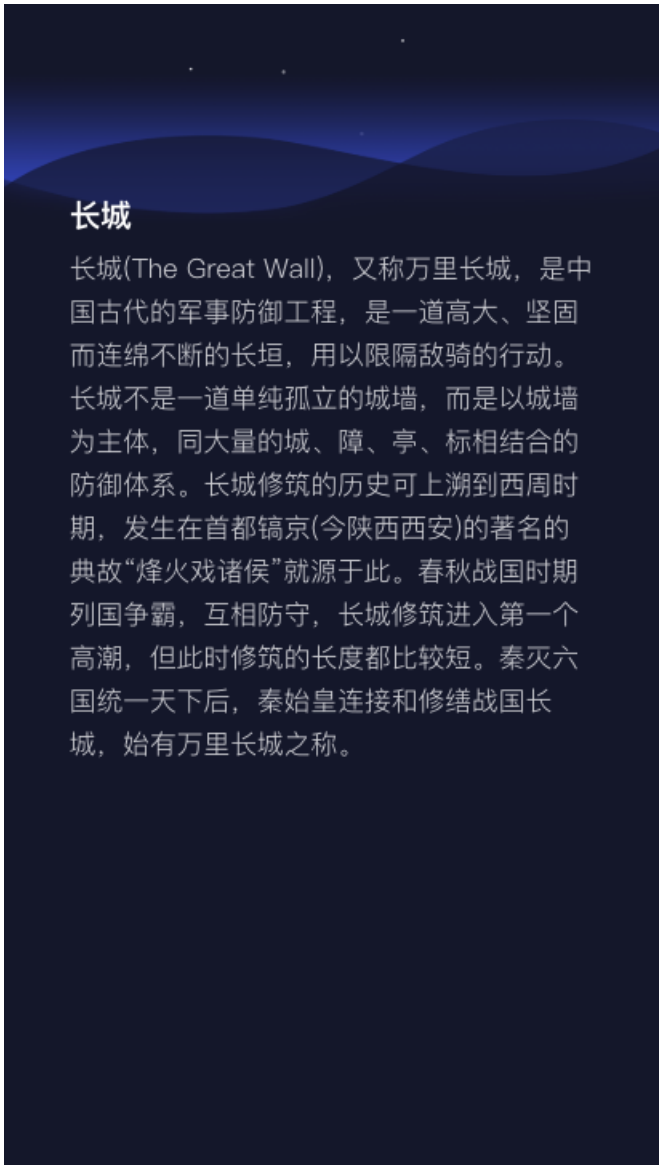
卡片显示

下面介绍的指令都是涉及客户端卡片显示的，客户端收到这些指令后，根据不同的类型展示不同的卡片。卡片显示指令统一使用RenderCard指令header，通过不同的payload来区分不同的卡片。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.screen",
     "name": "RenderCard",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     {{不同的payload}}
  }
}
```

TextCard类型卡片

文本卡片。参考排版方式：标题、内容上下排列，左对齐。



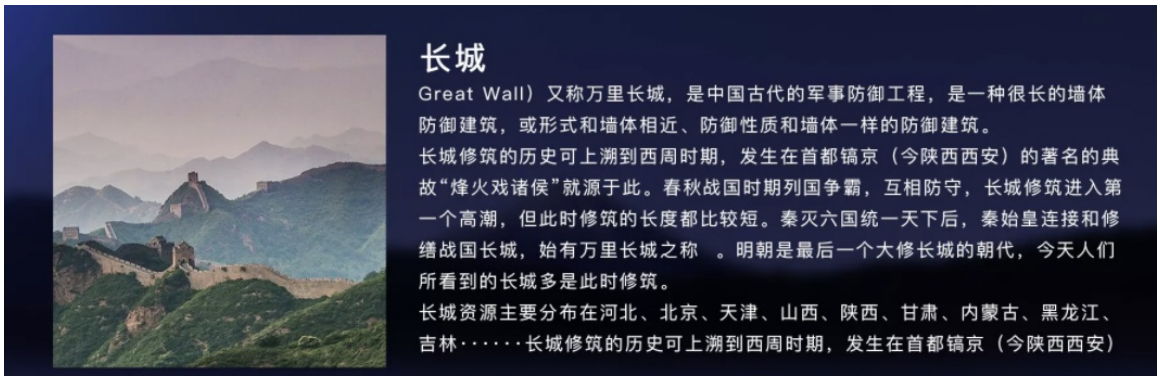
```
{
  "type": "TextCard",
  "title": "长城",
  "content": "${{长城|link0}}(The Great Wall)，又称万里长城...",
  "url": "https://baike.baidu.com/item/%E9%95%BF%E5%9F%8E/14251",
  "links": {
    "link0": {
      "type": "renderHtml",
      "url": "http://robot.baidu.com/"
    }
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
type	string	是	模板类型，固定取值为TextCard
title	string	否	标题
content	string	是	内容
url	string	否	点击卡片的跳转
links	json	否	详见links说明

ImageCard类型卡片

图片卡片，包含图片和对应的文本信息。参考排版方式：图片在左侧，标题、内容上下排列，三者左对齐。




```
{
  "type": "ImageCard",
  "title": "长城",
  "image": "https://timgsa.baidu.com/timg?20100603_96286_1.jpg",
  "content": "Great Wall）又称${{万里长城|link0}}，是中国古代的军事防御工程，是...",
  "url": "https://baike.baidu.com/item/长城/14251",
  "links": {
    "link0": {
      "type": "renderHtml",
      "url": "http://robot.baidu.com/"
    }
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
type	string	是	模板类型，固定取值为ImageCard
title	string	否	标题
content	string	是	内容
image	string	是	图片
url	string	否	点击卡片的跳转
links	json	否	详见links说明

ListCard类型卡片

列表卡片格式一，参考排版方式：列表包行多项，上下依次排列。每一项的排列方式和ImageCard的相同。

柳宗元

是中国古代的军事防御工程，是一道高大、坚固而连绵不断的长垣，

曾巩

是中国古代的军事防御工程，是一道高大、坚固而连绵不断的长垣，坚固而连绵不断的长垣坚固而连绵不断的长垣…

王安石

是中国古代的军事防御工程，是一道高大、坚固而连绵不断的长垣，

欧阳修

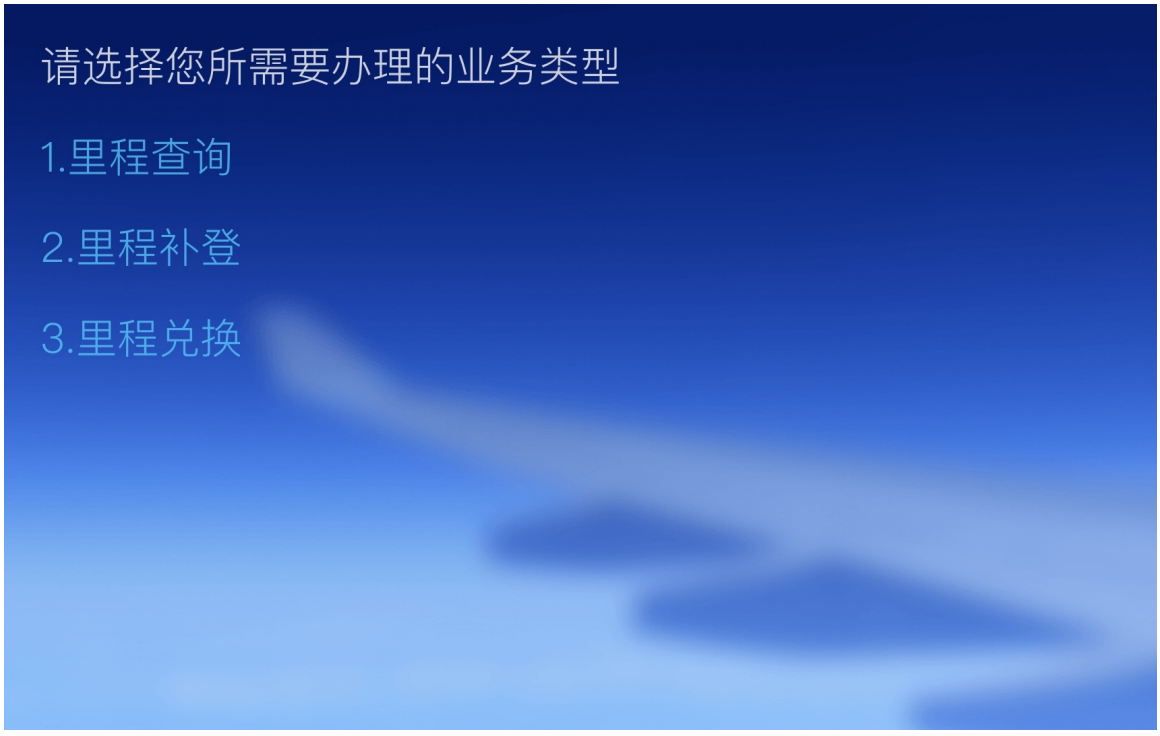
是中国古代的军事防御工程，是一道高大、坚固而连绵不断的长垣，

苏轼

是中国古代的军事防御工程，是一道高大、坚固而连绵不断的长垣，

```
{
  "type":"ListCard",
  "links":{
     "link0":{
        "speak":"播报话术",
        "type":"sendTextEvent",
        "query":"防御工程"
     },
     "link1":{
        "speak":"播报话术",
        "type":"sendTextEvent",
        "query":"防御工程"
     }
  }
  "title": "",
  "list":[
     {
        "title": "柳宗元",
        "content":"${{防御工程|link0}}",
        "image": "https://ss2.baidu.com/6ONYsjip0QIZ8tyhnq/it/u=2541339440,4048181160&fm=58",
        "url": ""
     },
     {
        "title": "曾巩",
        "content":"${{防御工程|link1}}",
        "image": "https://ss2.baidu.com/6ONYsjip0QIZ8tyhnq/it/u=2541339440,4048181160&fm=58",
        "url": ""
     },
     ...
  ]
}
```

列表卡片格式二，参考排版方式：卡片主标题和列表项上下依次排列，左对齐。



```
{
  "type":"ListCard",
  "links":{
     "link0":{
        "speak":"播报话术",
        "type":"sendTextEvent",
        "query":"里程查询"
     },
     "link1":{
        "speak":"播报话术",
        "type":"sendTextEvent",
        "query":"里程补登"
     },
     "link1":{
        "speak":"播报话术",
        "type":"sendTextEvent",
        "query":"里程兑换"
     }
  }
  "title": "请选择您所需要办理的业务类型",
  "list":[
     {
        "content":"${{1.里程查询|link0}}",
     },
     {
        "content":"${{2.里程补登|link1}}",
     },
     {
        "content":"${{3.里程兑换|link2}}",
     }
  ]
}
```

payload参数说明

字段名	类型	是否必需	说明
type	string	是	模板类型，固定取值为ListCard
title	string	否	标题
list	json	array	是
links	json	否	详见links说明

列表项

字段名	类型	是否必需	说明
title	string	否	标题
content	string	是	内容
image	string	否	图片
url	string	否	点击列表项的跳转

SimpleImageListCard类型卡片

简单图片列表卡片，为多个图片的集合，在仅展示图片的场景下，建议用此卡片。参考排版方式：这种类型只包含图片列表，图片排列依据屏幕宽度换行排列。

```
{
  "type": "SimpleImageListCard",
  "list": [
    {
      "image": "image_src",
      "url": "http://robot.baidu.com/"
    },
    {
      "image": "image_src",
      "url": "http://robot.baidu.com/"
    }
  ]
}
```

payload参数说明

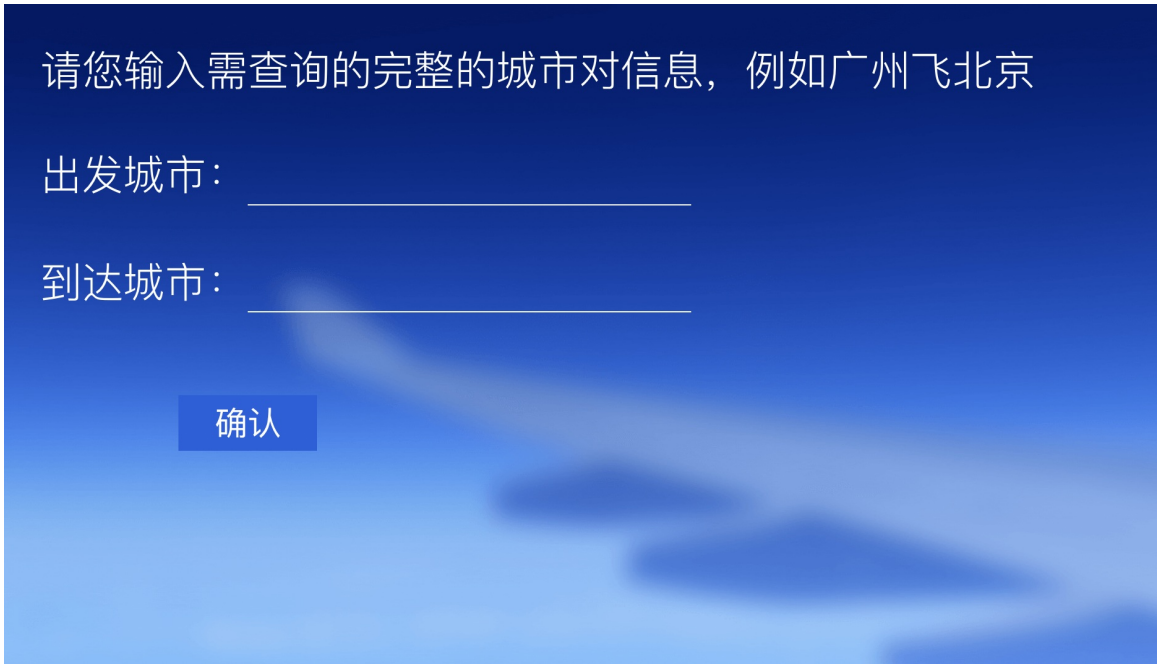
字段名	类型	是否必需	说明
type	string	是	模板类型，固定取值为SimpleImageListCard
list	json	array	是

列表项

字段名	类型	是否必需	说明
image	string	否	图片
url	string	否	点击列表项的跳转

InputCard类型卡片

输入卡片，参考排版格式：标题、内容、输入说明列表项上下依次排列，左对齐。




```
{
  "type": "InputCard",
  "content" : "请您输入需查询的完整的城市对信息，例如广州飞北京',
  "inputList" : [
     "出发城市：",
     "到达城市："
  ]
  "queryFormat": "${{input0}}飞${{input1}}"
}
```

payload参数说明

字段名	类型	是否必需	说明
type	string	是	模板类型，固定取值为InputCard
title	string	否	标题
content	string	是	内容
inputList	string array	是	输入列表说明，inputList的数量，对应输入框数量。
queryFormat	string	是	输入内容编排格式，对应格式为 ${{input0}} ${{input1}}，input的序号对应inputList数组的index。

HintsCard类型卡片

信息提示卡片，list中为提示内容列表信息。


```
{
  "type": "HintsCard",
  "content" : "小度能带你到这些地方',
  "list": [
{
 "content": "带我去第七展厅"
},
{
 "content": "带我去7号楼"
},
    {
 "content": "带我去卫生间"
},
    {
 "content": "带我去电梯口"
},
    {
 "content": "带我去净化器"
 }
]
}
```

payload参数说明

字段名	类型	是否必需	说明
type	string	是	模板类型，固定取值为HintsCard
content	string	否	标题
list	string	是	内容

复合类型卡片

复合类型卡片支持更复杂的卡片点击、卡片间跳转等操作，目前我们为用户航班信息查询场景，提供里程查询、消费查询等卡片间的跳转。如果开发者有接入需求，请提交工单联系我们。

links字段说明

文本点击操作，文本整段或部分显示可点击。自定义协议：**${{长城|link0}}**，link0为links json里的key。

字段名	类型	json key	json value类型	json value具体内容	说明
links	json				
		link0	json	{"type": "renderHtml","url": "http://xxxx"}	renderHtml点击获取网页。link0为点击文本对应的ID，实际使用时取值不定。type和url都是必须的。
		link1	json	{"type": "sendTextEvent","speak": "播报内容","query": "具体query"}	sendTextEvent点击发起event请求。speak为点击时的播报。query为event发起的文本query。
		link2	json	{"type": "renderCard","speak": "播报内容","cardIndex": 1}	renderCard点击跳转到本地卡片。speak为点击时的播报。cardIndex为卡片的ID。
		...			

音量控制

AdjustVolume指令

调节音量指令，客户端根据指令内的字段调节本地系统音量。

```
{
  "header": {
    "namespace": "baidu.abcrobot.directive.speaker_controller",
    "name": "AdjustVolume",
    "interactionId": "",
    "messageId": ""
  },
  "payload": {
    "volume_control": "down",
    "volume_value": "5.0"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
volume_control	string	是	声音调高或调低，取值为up和down。
volume_value	string	否	声音调高或调低的数值，例如："5.0"。当此字段为空时（如用户说“声音大一点”，没有包含具体的音量值），客户端可按照固定的音量值进行调节。

SetMute指令

静音指令，客户端根据指令设置打开或取消静音。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.speaker_controller",
     "name": "SetMute",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "switch":"on"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
switch	string	是	静音，取值为on和off，on表示打开静音，off表示取消静音。

运动控制

下面介绍的都是涉及客户端运动控制的指令，客户端收到这些指令后，可以控制客户端做出相应的动作。

Walk指令

行走指令，控制设备行走一段距离。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "Walk",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "direction": "forward",
     "distance": "5.0",
     "distance_unit": "米"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
direction	string	是	移动的方向，取值为forward、backward、left、right。
distance	string	否	移动的距离，例如："5.0"。
distance_unit	string	否	移动距离的单位，例如："米"。

Turn指令

转动指令，控制设备转动一定角度。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "Turn",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "direction": "forward",
     "angle": "90.0",
     "angle_unit": "度"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
direction	string	是	转动的方向，取值为forward、backward、left、right。
angle	string	否	转动的角度，例如："90.0"。
angle_unit	string	否	移动距离的单位，例如："度"。

RaiseHands指令

抬手指令，设备可以抬起手臂。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "RaiseHands",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "hands": "hand"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
hands	string	是	手臂选择，取值为hand、left_hand、right_hand。

ShakeHands指令

握手指令，设备可以做出握手动作。例如当用户说“握个手”时，云端会下发该指令。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "ShakeHands",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {}
}
```

Hug指令

拥抱指令，设备可以做出拥抱动作。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "Hug"
  },
  "payload": {}
}
```

TwistHead指令

摇头指令，设备可以做出摇头动作。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "TwistHead"
  },
  "payload": {}
}
```

TurnHeadLeft指令

向左看指令，客户端可以做出向左看动作。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "TurnHeadLeft",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {}
}
```

TurnHeadRight指令

向右看指令，客户端可以做出向右看动作。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "TurnHeadRight",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {}
}
```

Stop指令

停止运动指令，设备收到该指令后停止当前动作。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "Stop"
  },
  "payload": {}
}
```

Wave指令

打招呼指令。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "Wave",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {}
}
```

DoAnAction指令

动一下指令。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "DoAnAction",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {}
}
```

Charge指令

充电指令，控制设备开始充电或停止充电。例如，当用户说“回去充电”时，可控制机器人进行自主充电。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "Charge",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "switch": "on"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
switch	string	是	充电，取值为on和off，on表示开始充电， off表示停止充电。

巡航指令

巡航指令，控制设备开始巡航或停止巡航。例如，当用户说“开始巡航”时，可控制机器人按编排的路线进行巡航。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.action",
     "name": "Cruise",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "switch": "on"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
switch	string	是	巡航，取值为on和off，on表示开始巡航， off表示停止巡航。

导航指令

当用户说“带我去XXX”时，可控制机器人导航至XXX地点。

```
{
  "header": {
     "namespace": "baidu.airport.directive.navigation",
     "name": "Navigate",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "target": "卫生间"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
target	string	是	用户要导航的地点，要求该地点在机器人地图上存在

导览指令

开始导览指令

当用户说“开始导览”时，可控制机器人按编排的路线进行导览讲解。

```
{
  "header": {
     "namespace": "baidu.standard.directive.guide",
     "name": "StartGuideFlow",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "routeName":"导览路线1",
     "chatInGuiding" : true
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
flowName	string	是	导览路线名称
thumbnailsUrl	string	是	导览路线缩略图
chatInGuiding	string	是	在该路线的导览讲解过程中是否支持聊天

结束导览指令

当用户说“结束导览”时，可控制机器人结束导览讲解。

```
{
  "header": {
     "namespace": "baidu.standard.directive.guide",
     "name": "EndGuideFlow",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
  }
}
```

继续导览指令

当机器人在导览过程中被打断后处于暂停导览状态的情况下，用户说“继续导览”时，可控制机器人继续暂停前的导览讲解。

```
{
  "header": {
     "namespace": "baidu.standard.directive.guide",
     "name": "ProceedCurrentStep",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
  }
}
```

下一个导览点

当机器人在某一个导览点进行讲解时，可以通过“下一个讲解点”话术，控制机器人结束当前点的讲解，直接到下一个导览点进行讲解。

```
{
  "header": {
     "namespace": "baidu.standard.directive.guide",
     "name": "ProceedNextStep",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
  }
}
```

人脸指令

FaceRecognitionResult指令

人脸识别结果指令，云端下发人脸识别结果给客户端。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.face_recognition",
     "name": "FaceRecognitionResult",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "name": "name",
     "sex": "MALE",
     "age": 30，,
     "uid": "N1088",
     "faceliveness": true,
     "type": "VIP"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
name	string	否	人脸的名字
sex	string	否	人脸的性别，取值位MALE和FEMALE。
age	int	否	人脸的年龄
uid	string	否	人脸ID
faceliveness	boolean	是	检测是否是活体
type	string	是	人脸类型，取值为VIP和UNKNOWN。

FaceComparisonResult指令

人脸比对结果指令，云端下发人脸比对结果给客户端。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.face_recognition",
     "name": "FaceComparisonResult",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "score": 89.9
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
score	float	是	比对分数，80分以上可以认为是同一个人

GetFaceImage指令

上传人脸图片指令，客户端收到这个指令后，上传人脸图片到云端，与FaceImageInput事件配合使用。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.image_input",
     "name": "GetFaceImage",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {}
}
```

系统指令

SyncRobotConfig指令

云端配置下发指令，由RobotConfig事件触发。

```
{
  "header": {
    "namespace": "baidu.abcrobot.directive.system",
    "name": "SyncRobotConfig",
    "interactionId": "",
    "messageId": ""
  },
  "payload": {
    "welcomeSpeeches": [
      "您好,很高兴为您服务",
      "${name}您好,很高兴为您服务"
    ],
    "faceWakeUpType": 1,
    "sensitiveWords":[
      "abc",
      "edf"
    ]
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
welcomeSpeeches	list of string	是	欢迎语列表
faceWakeUpType	int	是	人脸唤醒类型，1在线，2代表离线。
sensitiveWords	list of string	否	敏感词列表

InquireEnergy指令

电量查询指令，客户端收到这个指令后，显示客户端电量。

```
{
  "header": {
    "namespace": "baidu.abcrobot.directive.instruction",
    "name": "InquireEnergy",
    "interactionId": "",
    "messageId": ""
  },
  "payload": {}
}
```

ShowFeatures指令

功能页展示指令，客户端收到这个指令后，展示功能介绍页。

```
{
  "header": {
    "namespace": "baidu.abcrobot.directive.instruction",
    "name": "ShowFeatures",
    "interactionId": "",
    "messageId": ""
  },
  "payload": {}
}
```

媒体资源控制指令

媒体播放指令

用于播放媒体资源，如唱一首没有共产党就没有新中国。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.media_control",
     "name": "PlayMedia",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "name":"没有共产党就没有新中国",
     "type":"SING_SONG"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
type	string	是	播放媒体类型，目前只支持“SING_SONG”类型
name	string	是	播放内容名称

媒体播放控制指令

用于媒体播放控制，如暂停，继续，下一个，换一个等

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.media_control",
     "name": "PlayControl",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "action":"play_next"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
action	string	是	播放控制动作，可选值：play_pause(暂停), play_continue(继续), play_next(下一个), play_switch(切换)

页面切换控制指令

用于页面切换控制，比如返回主页，返回上一页，下一页等

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.media_control",
     "name": "PageControl",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "pageName":"MAIN_PAGE"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
pageName	string	是	页面名称，可选值：MAIN_PAGE(返回主页),PREVIOUS_PAGE(返回父级页面),LAST_PAGE(上一页),NEXT_PAGE(下一页)

按索引选择指令

用于在媒体列表中，按索引选择要操作的媒体项。如第一个/页，倒数第一个/页等，

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.media_control",
     "name": "SelectByIndex",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "order": "POS",
     "index": 1,
     "unit": "page"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
order	string	是	选择顺序，如，第一个、倒数第一个等。可选值：POS(顺序)、NEG(倒序)
index	int	是	选择索引，整数，可选值：1、2、3...
unit	string	是	索引单位，如第一页、第一个等。可选值：page(页)、pcs(个)

其它指令

NLUResult指令

语义理解结果指令，开发者使用自定义技能时，可以在技能回复中配置成直接获取NLU结果，并通过这个指令下发。

```
{
  "header": {
    "namespace": "baidu.abcrobot.directive.nlu",
    "name": "NLUResult",
    "interactionId": "",
    "messageId": ""
  },
  "payload": {
    "skill": {
      "name": "CUSTOMER"
    },
    "intent": {
      "name": "RECHARGE"
    },
    "slots": [
      {
        "name": "cellphone",
        "originalWord": "13810102020",
        "normalizedWord": "13810102020"
      }
    ]
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
skill.name	string	是	技能名称
intent.name	string	是	意图名称
slots	json array	否	槽位列表
slots[i].name	string	是	槽位名称
slots[i].originalWord	string	是	槽位值
slots[i].normalizedWord	string	是	归一化槽位值

CustomData指令

自定义参数下发指令，通常为控制台配置的KV集合，例如问答库中的自定义参数。

```
{
  "header": {
    "namespace": "baidu.abcrobot.directive.custom",
    "name": "CustomData",
    "interactionId": "",
    "messageId": ""
  },
  "payload": {
    "data": {
      "key": "value",
      ...
    }
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
data	json	是	自定义参数json结构，json里的取值是自定义的key和value，目前问答库支持自定义参数下发。

技能分发描述指令

用于表明用户问题命中技能和答案来源技能。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.skill_dispatch",
     "name": "SkillInfo",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "type": "Navigation",
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
type	String	是	问题命中技能信息，目前只支持Navigation，在没有明确下发导航指令，但用户问题命中导航意图时下发，用于机器人端打开导航相关页面的判断条件

End指令

再见指令，客户端收到这个指令后，客户端重置和云端交互的session。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.session",
     "name": "End",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {}
}
```

Error指令

错误指令，客户端请求云端返回此指令，表示请求失败，客户端根据相应的错误码做错误处理。

```
{
  "header": {
     "namespace": "baidu.abcrobot.directive.error",
     "name": "Error",
     "interactionId": "",
     "messageId": ""
  },
  "payload": {
     "code": 50300,
     "message": "……"
  }
}
```

payload参数说明

字段名	类型	是否必需	说明
code	int	是	错误码
message	string	是	错误描述

错误码

错误码	描述	类别	备注
40001	客户端http请求缺少参数,或者参数错误	请求数据数据问题(4xxxx)	详细信息见message字段
40002	没有对应的处理器处理该event	请求数据数据问题(4xxxx)	详细信息见message字段
50100	协议模块通用错误	协议模块问题(501xx)	详细信息见message字段
50200	对话模块通用错误	下游对话服务问题(502xx)	详细信息见message字段
50300	人脸模块通用错误	下游人脸服务问题(503xx)	详细信息见message字段
50000	其他通用错误	通用错误	详细信息见message字段

Android SDK开发指南

SDK-V2.1

简介

ABC Robot Android SDK是百度机器人平台面向开发者提供的一套低成本接入和定制化开发方案，通过本SDK可灵活调用平台提供的语音、语义和视觉能力，结合机器人平台管理控制台提供的可视化管理能力，让机器人具备类人的人机交互体验。

SDK面向的主要开发者如下：

类型	描述	场景
本体商	机器人、智能屏幕和XTM终端等泛机器人本体制造商	预置百度机器人平台ABC Robot核心技术和平台能力，方便于其他开发者基于该本体进行二次开发，实现软硬能力的强强结合
集成商	有较强软件开发能力的行业集成商伙伴	基于SDK定制开发适用于各类行业场景的人机交互应用，例如专门做医疗行业辅助诊断的机器人集成商和方案商
企业客户	有较强软件开发能力的企业客户	企业基于平台开发自己的机器人产品，例如某家庭机器人企业，移动营业厅对话式智能终端等。

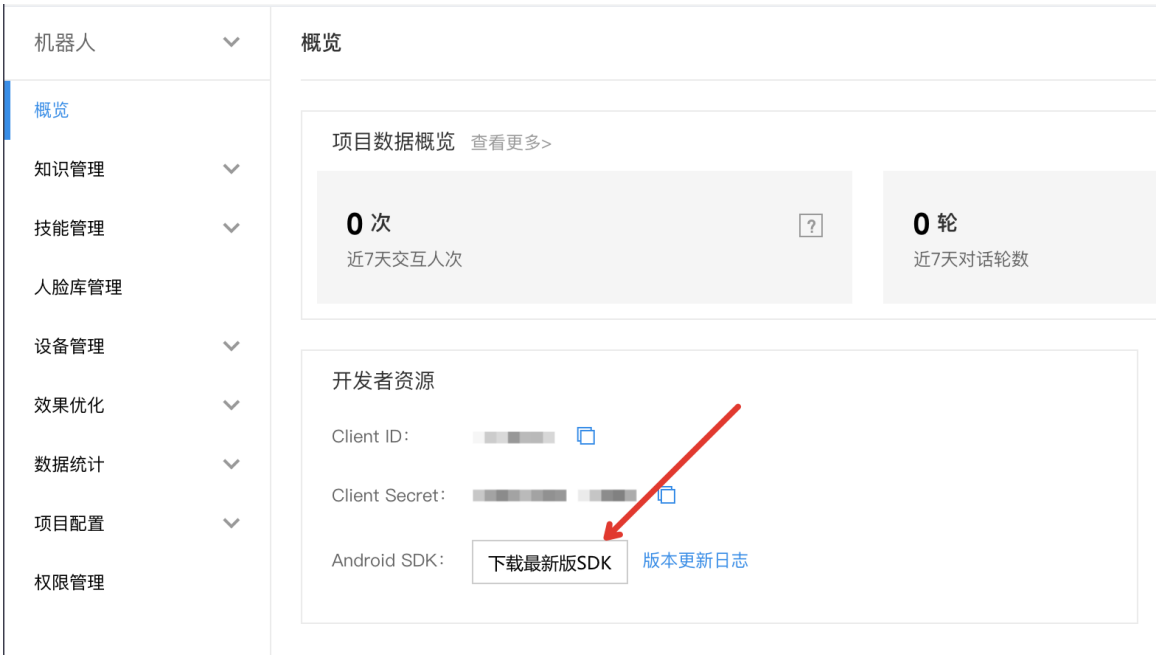
SDK具备的主要业务功能：

功能	描述
语音唤醒	默认唤醒词是"小度小度"，可付费定制其他唤醒词。如果机器人使用的是百度麦克风阵列，还可获取唤醒角度，实现声源定位
语音识别	通过自定义语音模块，可支持除内置麦克风和百度麦克风阵列之外的第三方麦克风接入的语音识别功能
语音合成	离在线TTS默认支持标准男、标准女两种音色
语音对话	语音识别+语义理解+语音合成可实现语音对话功能，通过管理控制台可管理知识库、训练对话模型和编排对话逻辑，含语音指令的解析
人脸检测	人脸检测landmarks数据及位图信息，获得表情、性别、年龄等人脸属性信息，人脸情绪信息，以及活体打分，可基于人脸检测实现更自然的唤醒
人脸识别	支持离/在线人脸识别，包括1:1和1:N，SDK端支持本地离线人脸库管理，在线人脸库可通过管理控制台在线管理

集成准备

下载SDK

1. 下载SDK需要申请并开通平台服务，具体请参考开通服务。
2. 开通完成后，在管理控制台完成项目创建，具体参考项目管理。
3. 进入一个项目，左侧导航栏点击“**概览**”，进入“**概览**”页面。
4. 点击“**下载最新版SDK**”即可。

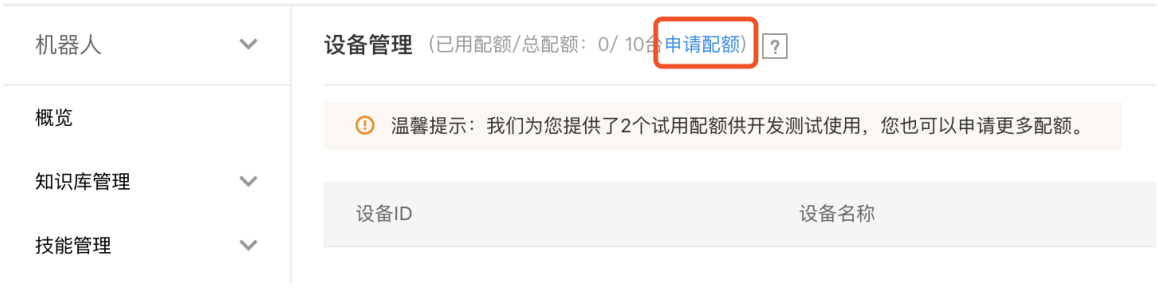


获取Client ID、Client Secret

在集成SDK之前，您还需要在“**概览**”页面获取该项目的Client ID和Client Secret。

设备配额

我们为您提供了2个试用配额供开发测试使用，1个配额可激活1台设备，您也可以在管理控制台—设备管理页面申请更多配额。



环境准备

将AAR包放入项目的libs文件夹下，位置如下：

```
RobotApp // 开发者的项目根目录
├──── app
│    ├──── build.gradle
│    ├──── libs
│    │    └──── AbcRobotSDK-v2.x.x.aar <--------- 请把机器人SDK的aar放到这里
│    │    └──── lib-liantian-releaseLogenable-3.1.6.7.aar <--------- 机器人SDK依赖aar资源
│    ├──── proguard-rules.pro
│    └──── src
├──── build.gradle
├──── gradle
│    └──── wrapper
├──── gradle.properties
├──── gradlew
├──── gradlew.bat
├──── local.properties
└──── settings.gradle
```

兼容性

- minSdkVersion 大于等于 21
- 建议使用最新版本Android Studio进行开发，对Eclipse开发过程中遇到的问题暂不提供支持。

快速上手

概述

ABC Robot Android SDK主要包括语音组件、人脸组件、指令组件三大功能，初始化SDK成功以后，开发者可以通过API调用这三个功能。

SDK初始化

SDK初始化阶段是指对各大功能组件进行初始化，所以接入SDK必须要进行初始化，才能保证各大组件的正常使用。
开发者先通过ABC Robot管理控制台，申请对应的Client ID和Client Secret，然后根据硬件平台，设置对应的初始化类型。

语音组件

RobotSdk支持多种语音方案：RK3326、 RK3399、手机和自定义语音方案，开发者根据自己的硬件类型，选择对应的语音方案初始化SDK。
具体的使用流程，根据开发者的场景需求，自主选择搭配。

人脸组件

不同硬件平台摄像头数量和位置不一样，开发者在使用人脸功能之前，需要在SDK初始化的时候配置摄像头和图像相关参数。
目前我们支持人脸的离线和在线识别，在线识别需要配置ABC Robot管理控制台人脸库，离线识别需要在机器人的sdcard中导入人脸图片。

指令组件

端云交互依赖于指令组件，开发者可以在ABC Robot管理控制台配置相关功能，云端会通过指令将返回结果下发给SDK，SDK通过指令组件将数据传输给APP层。

初始化

我们在 SDK 的压缩文件夹中提供了一个简单的集成示例 Sample App 。初次使用，推荐参考 Sample App 配合官网文档来进行 SDK 的集成。

Sample App 中包含了最小程度的集成代码示例，开发者可以直接将 SDK 的 AAR 文件导入 Sample 中运行。 如果 Sample App 中各项功能均能够运行成功，则说明硬件和系统环境已经集成完毕。开发者可以参考 Sample App 中的代码来使用 SDK 的各项功能。

接下来介绍手动集成 AAR 到项目中的步骤：

集成步骤

1. 在项目的build.gradle中进行依赖配置，配置如下：

```
 android {
    repositories {
        flatDir {
            dir 'libs'
        }
    }
}

dependencies {
    // AbcRobotSDK-v2.x.x为RobotSDK aar文件的名称。
    implementation(name: 'AbcRobotSDK-v2.x.x', ext: 'aar')
    // SDK依赖库
    implementation(name: 'lib-liantian-releaseLogenable-3.1.6.7', ext: 'aar')
}
```

2. 需要添加的远程依赖如下，如您项目中已经包含则无需添加：

```
 // Android Official libs
implementation 'com.android.support:appcompat-v7:28.0.0'
implementation 'com.android.support:recyclerview-v7:28.0.0'

// Third party libs
implementation 'com.squareup.okio:okio:1.14.0'
implementation 'com.squareup.okhttp3:okhttp:3.10.0'
implementation 'com.alibaba:fastjson:1.2.67'
implementation 'org.greenrobot:greendao:3.2.2'
```

初始化

配置初始化参数

SDK初始化参数配置包括SDK全局参数和SDK组件参数，分别通过全局参数对象和SDK各组件对应的组件参数对象进行配置。

SDK全局参数配置

通过RobotSDKConfig对象完成SDK全局参数的配置，这里列出主要部分：

```
 RobotSDKConfig sdkConfig = new RobotSDKConfig();
// 配置项目ClientID，项目鉴权时使用ID
sdkConfig.setClientId(RobotApplication.CLIENT_ID);
// 配置项目ClientSecret，项目鉴权时使用鉴权码
sdkConfig.setClientSecret(RobotApplication.CLIENT_SECRET);
```

SDK组件参数配置

通过SDK组件配置对象，完成对应组件的参数配置，SDK包括的组件和对应的组件配置对象：

组件名称	配置对象
语音功能组件	SpeechModuleConfig
人脸功能组件	FaceModuleConfig
事件指令组件	DirectiveModuleConfig
数据打点组件	DataTrackingConfig

组件参数配置方法，以SDK的人脸功能组件为例：

```
 // 初始化人脸组件配置对象
FaceModuleConfig faceModuleConfig = new FaceModuleConfig();
// 摄像头设备类型（TYPE_INTERNAL_FRONT设备摄像头）
faceModuleConfig.cameraType = TYPE_INTERNAL_FRONT;
// 摄像头设备ID
faceModuleConfig.cameraId = "1";
// 摄像头图片帧方向
faceModuleConfig.videoDirection = 0;
// 图片帧人脸识别前，图片需要旋转的角度
faceModuleConfig.faceAngle = ANGLE_270;
// 人脸识别方式（在线识别，离线识别）
faceModuleConfig.recType = REC_TYPE_ONLINE;
// 在线识别对应的服务端人脸库名称
faceModuleConfig.faceGroup = "VIPFace";
```

设置组件配置对象

完成组件配置对象设置，以SDK的人脸功能组件配置为例：

```
 RobotSDKConfig sdkConfig = new RobotSDKConfig();
...
FaceModuleConfig faceModuleConfig = new FaceModuleConfig();
...
// 设置人脸组件配置对象
sdkConfig.addModuleConfig(faceModuleConfig);
```

执行SDK初始化

初始化方法 使用SDK初始化方法和RobotSDKConfig配置对象完成SDK初始化，通过实现ISDKInitListener初始化回调接口监听初始化过程。 初始化方法内部为异步操作不会阻塞当前线程。

```
 // 获取SDK引擎对象
RobotSDKEngineBuilder robotSDKEngineBuilder =
        new RobotSDKEngineBuilder();
IRobotSDKEngine iRobotSDKEngine = robotSDKEngineBuilder.build();
// 初始化SDK
iRobotSDKEngine.init(context, sdkConfig, initListener);
```

初始化回调接口 实现ISDKInitListener接口，可以监听SDK初始化状态。onInitFinish执行结束后，返回的参数initCode >= 0表明SDK初始化成功。

```
 public interface ISDKInitListener {

    /**
     * 初始化过程回调方法
     *
     * @param initModuleName 完成初始化的功能或组件名称
     * @param initCode       功能或组件初始化状态（大于0为初始化成功状态，小于0为初始化失败状态）
     * @param initMsg        功能或组件初始化状态信息
     */
    void onInitProcess(String initModuleName, int initCode, String initMsg);

    /**
     * 初始化完成回调方法
     *
     * @param initCode SDK初始化状态码（大于0为初始化成功状态，小于0为初始化失败状态）
     * @param initMsg  SDK初始化状态信息
     */
    void onInitFinish(int initCode, String initMsg);

}
```

获取设备ID

SDK激活完成后，您可以在管理控制台—设备管理的列表中查看到最新激活的设备ID、设备名称等信息。同时，SDK端可以通过调用`IRobotSDKEngine.getDeviceId()`函数获取设备ID，以建立Console端和SDK端的对应关系。

初始化状态码 错误码为负数，状态码为正数

错误码/状态码	描述
10000	SDK初始化成功
10001	SDK权限检测初始化成功
10002	SDK运行状态设置成功
10003	SDK网络状态设置成功
10004	SDKlog设置成功
10005	SDK本地资源设置成功
10006	SDK在线激活成功
10007	SDK获取人脸鉴权文件成功
10008	SDK语音组件初始化成功
10009	SDK人脸组件初始化成功
10010	SDK指令组件初始化成功
-10000	SDK初始化失败
-10001	SDK运行状态设置失败
-10002	SDK网络状态设置失败
-10003	SDKlog设置失败
-10004	SDK本地资源设置失败
-10005	SDK在线激活失败
-10006	SDK获取aipe人脸鉴权文件失败
-10007	SDK组件初始化失败
-10008	SDK人脸组件初始化失败
-10009	SDK语音组件初始化失败
-10010	SDK指令组件初始化失败
-10013	语音组件状态码，串口设备为null
-10014	语音组件状态码，IO接口错误
-10015	SDK权限检测异常

语音组件

语音组件整体介绍

本文档是RobotSdk中语音组件的开发指南，包含了组件配置，使用流程，方法API，自定义语音方案接入等相关说明。
RobotSdk支持三种语音方案，分别是 RK3399语音方案，Android手机自带语音和自定义的语音方案。 开发者可以根据所使用的硬件平台，来初始化不同的语音方案。
我们同样支持自定义语音方案，开发者可以根据自己语音方案的实际情况，通过我们提供的接口进行接入。

组件配置

1. 获取语音组件配置对象

```
SpeechModuleConfig speechModuleConfig = new SpeechModuleConfig(context);
```

2. 语音识别引擎类型

```
/**
* 设置语音方案前，确认使用SDK设备支持的语音引擎类型
* SDK支持语音识别引擎列表：
* RK3399: SPEECH_TYPE_NUWA
* Android手机： SPEECH_TYPE_INTERNAL
* 自定义语音方案： SPEECH_TYPE_CUSTOMER
**/
speechModuleConfig.speechType = SPEECH_TYPE_NUWA;// NUWA语音方案
```

3.是否开启语音角度抑制功能

```
// 使机器人只倾听正前方目标说话人的声音，优化其他人声和嘈杂环境的干扰。
speechModuleConfig.enableSpeechDirectionCheck = true;// 开启角度抑制
```

4.语音播报时，服务端控制是否打断

```
speechModuleConfig.isTtsIntercept = true;// 开启语音播报时，服务端控制是否打断
```

5.语音识别log级别设置

```
speechModuleConfig.speechLogLevel = 6;// 0~6, 数字越大log内容越详细
```

组件使用流程

下面四个方法是语音常用的四个API，通过这四个方法，开发者就可以使用整个语音功能。
开启唤醒监听

```
RobotSDKManager.getInstance().startWakeupListening();
```

关闭唤醒监听

```
RobotSDKManager.getInstance().stopWakeupListening();
```

打开语音识别

```
RobotSDKManager.getInstance().startSpeechListening();
```

关闭语音识别

```
RobotSDKManager.getInstance().stopSpeechListening();
```

具体的使用流程，根据开发者的场景需要，自主选择。下面是我们给出的两个场景示例

1. 开始唤醒->用户通过唤醒词唤醒->关闭唤醒->开启识别->语音交互->交互结束->关闭识别->开启唤醒->等待用户唤醒
   这种场景下，需要用户通过唤醒词唤醒后进行语音交互，在打开识别的同时，建议开发者关闭唤醒。

2. 开启识别->语音交互->交互结束->关闭识别
   这种场景下，不需要唤醒词唤醒，直接开启识别，开启识别的时机和方式开发者自己定义。

其它API说明

1. void adjustVolume(int direction);
调节音量

- SpeechEngine.VOLUME_UP 调大音量
- SpeechEngine.VOLUME_DOWN 调小音量

```
RobotSDKManager.getInstance().adjustVolume(SpeechEngine.VOLUME_UP);
```

2. void speak(String speech);

语音播报

```
RobotSDKManager.getInstance().speak("播报内容");
```

3. void stopSpeaking();

停止语音播报

```
RobotSDKManager.getInstance().stopSpeaking();
```

4. void cancelSpeechListening();

取消语音识别和stopSpeechListening的效果不同

- stopSpeechListening() 停止语音识别，不会立即结束当前识别，等待当前识别结束并返回结果
- cancelSpeechListening() 取消语音识别，会打断当前识别，立即结束

```
RobotSDKManager.getInstance().cancelSpeechListening();
```

5. void switchLanguage();

切换中英文识别结果

- RobotSpeechConstant.LAN_CHINESE 中文识别
- RobotSpeechConstant.LAN_ENGLISH 英文识别

```
RobotSDKManager.getInstance().switchLanguage(RobotSpeechConstant.LAN_ENGLISH);
```

5. void registerSpeechListener(ISpeechCallback callback);

注册语音回调

```
RobotSDKManager.getInstance().registerSpeechListener(this);
```

ISpeechCallback 各个接口回调的含义

```
 /**
 * 语音功能接口
 */
public interface ISpeechCallback {

    /**
     * 语音识别状态回调
     * @param status 状态码
     */
    public void onStatus(int status);

    /**
     * 语音识别结果回调
     * @param speech 识别到的语音所转换的文本数据
     */
    public void onNewSpeech(SpeechModel speech);

    /**
     * 语音唤醒事件回调
     * @param word 唤醒词
     */
    public void onWakeUp(String word);

    /**
     * 唤醒角度回调
     * @param angle 唤醒角度
     */
    public void onWakeAngle(int angle);

    /**
     * 错误事件回调
     * @param errorCode 错误码
     */
    void onSpeechRunningStatus(int statusCode, String statusMsg);

    /**
     * 配置语音识别语种（目前支持中文 zh-CN，英文 en-us）.
     * @param 语音识别语种
     */
    void onConfigObtained(String languageId);

    /**
     * 播报音量变化回调
     * @param volumePercent 当前音量的相对值（0-100）
     * @param volume        当前音量
     */
    void onAsrVolume(int volumePercent, int volume);
}
```

6. void unregisterSpeechListener(ISpeechCallback callback);

解除注册语音回调

```
 RobotSDKManager.getInstance().unregisterSpeechListener(this);
```

自定义语音方案

RobotSdk支持接入自定义语音方案

1. 创建语音识别引擎SpeechEngine

继承抽象类SpeechEngine并实现对应的方法，根据自己语音方案提供相应的功能，主要方法有 startTTS， stopTTS，

startASR， stopASR， startWakeUp， stopWakeUp， setVolume

```
// 第三方语音功能实现类，下面为示例代码，实际初始化类以使用的第三方语音功能实现类为准
AbcSpeechControlInterface speechEngine = new AbcSpeechControlInterfaceImpl(context);
new CustomSpeechListener(speechEngine, context);
```

2. 实现SpeechEngine接口并将步骤1中的第三方语音功能实现类传入进来

```
public class CustomSpeech extends SpeechEngine {~~~~
    // 第三方语音功能实现类
    private AbcSpeechControlInterface mSpeechEngineImpl;
    private static final String ACTION_FLAG = "KLYLSpeech";

    public CustomSpeech(AbcSpeechControlInterface speechEngine) {
        this.mSpeechEngineImpl = speechEngine;
    }

    @Override
    public synchronized void speak(String speech) {
        mSpeechEngineImpl.startTTS(speech, true);
    }

    @Override
    public void stopSpeaking() {
        mSpeechEngineImpl.stopTTS();
    }

    @Override
    public void startListening() {
        mSpeechEngineImpl.startASR();
    }

    @Override
    public void cancelListening() {
        mSpeechEngineImpl.stopASR();
    }

    @Override
    public void stopListening() {
        mSpeechEngineImpl.stopASR();
    }

    @Override
    public void startWakeUp() {
        mSpeechEngineImpl.startWakeUp();
    }

    @Override
    public void stopWakeUp() {
        mSpeechEngineImpl.stopWakeUp();
    }

    @Override
    public void queryConfiguration() {

    }

    @Override
    public void adjustVolume(int value) {
        if (VOLUME_DOWN == value) {
            mSpeechEngineImpl.setVolume(AbcSpeechControlInterface.VOLUME_TYPE_DECREASE, 0, null);
        } else if (VOLUME_UP == value) {
            mSpeechEngineImpl.setVolume(AbcSpeechControlInterface.VOLUME_TYPE_INCREASE, 0, null);
        }
    }
}
```

3. 对自定义语音方案进行初始化

```
public class CustomSpeechListener {

    // 第三方语音功能实现类
    private AbcSpeechControlInterface mSpeechEngineImpl;
    private Context mContext;

    public CustomSpeechListener(AbcSpeechControlInterface speechEngine, Context mContext) {
        this.mSpeechEngineImpl = speechEngine;
        this.mContext = mContext;
        init();
    }

    public void init() {
        KLYLSpeech speech = new KLYLSpeech(mSpeechEngine);
        // 注册自定义语音方案
        RobotSDKManager.getInstance().registerSpeech(mContext, speech);
        // 创建回调listenner
        final SpeechRecListener mListener = new SpeechRecListener(speech);
        // 对自定义语音方案进行回调监听
        mSpeechEngineImpl.registerSpeechLisenter(new AbcSpeechControlInterface.ThirdSpeechCallBack() {
            // 语音状态回调
            @Override
            public void onThirdStatus(int status) {
                mListener.setStatus(status);
                mListener.notifyStatus();
            }
            // 用户语音识别回调
            @Override
            public void onThirdNewSpeech(int id, String speech, boolean isFinal) {
                // 如果使用我们的云端服务，将识别结果通过listener回传
                mListener.onAsrFinalResult(speech, null);
            }
            // 如果使用的是我们的云端服务，可以忽略此方法
            @Override
            public void onThirdNewAnswer(String answer) {
            }
            // 唤醒回调
            @Override
            public void onThirdWakeUp(String word, int wakeUpAngle) {
                mListener.onWpSuccess(word);
                mListener.onWpAngle(wakeUpAngle);
            }
            // 错误码回调
            @Override
            public void onError(int errorCode) {
                mListener.onError(errorCode, "");
            }
        });
    }
}
```

3. 在App的Application中将我们的语音方案进行初始化

```
AbcSpeechControlInterface speechEngine = new AbcSpeechControlInterfaceImpl(context);
new CustomSpeechListener(speechEngine, context);
```

状态码以及错误码

错误码为负数，状态码为正数

错误码/状态码	描述
-20000	RK3326语音板WIFI断开连接
-20001	RK3326语音板串口打开失败
-20002	RK3326语音板speech service语音读数据失败
-20003	RK3326语音板串口命令发送失败
-20004	RK3326语音板3326串口驱动为空
-20005	RK3326语音板asr识别发生错误
-20006	RK3326语音板串口通信超时
-20007	RK3326语音板串口重新连接
-20008	RK3326语音板speech service 重启，ASR重新连接
-20009	RK3326语音板speech service 重启，Wakeup重新连接
-20010	RK3326语音板speech service 其它服务重新连接
-20011	RK3326语音板WIFI已关闭
-20012	ASR退出，请稍等，SDK本身包含自动重启ASR机制
-20013	RK3326语音板ASR退出,正在重启
-20014	RK3326语音板Wakeup正在重试启动中，请勿多次调用
-20015	RK3326语音板Wakeup启动失败，原因是重复开启
-20016	RK3326语音板ASR正在重试启动中，请勿多次调用
-20017	RK3326语音板ASR启动失败，原因是重复开启
20000	RK3326语音板WIFI已连接
20001	RK3326语音板WIFI已开启
20002	唤醒已经开启
20003	RK3326语音板启动语音唤醒，正在重试
20004	ASR已经开启
20005	RK3326语音板启动语音识别，正在重试

人脸组件

人脸组件整体介绍

SDK支持通过机器人设备中摄像头采集到的图像信息，进行人脸检测，在线/离线人脸识别。

组件配置

使用SDK的人脸组件功能前，需要人脸组件功能进行配置。

获取配置对象

```
FaceModuleConfig faceModuleConfig = new FaceModuleConfig()
```

配置组件

```
// 相机类型
faceModuleConfig.cameraType = FaceModuleConfig.TYPE_INTERNAL_FRONT;
// 相机图像帧，人脸检测前旋转角度
// 正常完成人脸检测识别，需要图像帧内的人脸方向为正向
faceModuleConfig.faceAngle = FaceModuleConfig.ANGLE_90;
// 预览图片方向
faceModuleConfig.videoDirection = FaceModuleConfig.ANGLE_270;
// 人脸识别方式为在线
faceModuleConfig.recType = FaceModuleConfig.REC_TYPE_ONLINE;
// 人脸识别方式为离线
// faceModuleConfig.recType = FaceModuleConfig.REC_TYPE_OFFLINE;
// 离线人脸图片目录路径（属性默认路径为/storage/emulated/0/baidu_robot/UserPicture）
// faceModuleConfig.offlineFaceRecognizeImageDir = "/sdcard/faceImage"
// 在线1:N人脸识别，设置使用人脸库ID
// 如果不设置，会默认使用开放平台配置的默认人脸库，设置人脸库属性要确保开放平台已配置对应的库信息
faceModuleConfig.faceGroup = "VIPFace";
```

组件使用流程

启动/停止人脸识别功能 通过调用RobotSDKManager.getInstance().startFaceRecognize()接口来启动人脸识别功能。

```
/**
* 启动人脸识别功能
* @param textureView UI预览用TextureView，需要设置为可见，View宽高最小为1dp*1dp
* @param faceTaskType 人脸功能任务类型（人脸识别使用FaceModuleConfig.TASK_FACE_RECOGNIZE）
* @param faceDetectCallBack 人脸检测回调接口
* @return 是否启动成功
*/
boolean startFaceRecognize(TextureView textureView,
                           int faceTaskType,
                           IFaceDetectCallBack faceDetectCallBack);
```

通过RobotSDKManager.getInstance().stopFaceRecognize()接口来停止人脸识别功能。

```
/**
* 停止人脸识别
*/
void stopFaceRecognize();
```

获取人脸检测回调信息 开启人脸识别功能后，可以通过实现IFaceDetectCallBack接口来获取启动人脸识别功能后，人脸检测的回调信息。

```
/**
 * 人脸检测回调接口
 */
public interface IFaceDetectCallBack {

    /**
     * 人脸检测回调方法
     *
     * @param livenessModel 检测到人脸数据对象
     */
    void onFaceDetectCallback(FaceModel livenessModel);

    /**
     * 人脸检测状态回调方法
     *
     * @param code 人脸检测状态码
     * @param msg  人脸检测状态信息
     */
    void onFaceDetectStatus(int code, String msg);

    /**
     * 人脸检测UI渲染接口（预留功能接口，暂不使用）
     *
     * @param livenessModel 检测到人脸数据对象
     */
    void onFaceDetectDrawCallback(FaceModel livenessModel);

    /**
     * 离线人脸识别回调接口
     * 人脸组件配置为离线识别时使用faceModuleConfig.recType = REC_TYPE_OFFLINE;
     *
     * @param faceModel 检测到人脸数据对象
     */
    void onOfflineFaceRecognize(FaceModel faceModel);

    /**
     * 发现有人脸（没有人脸质量检测）
     * @param livenessModel
     */
    void onNoQualityDetectCallback(FaceModel livenessModel);
}
```

人脸检测回调接口实现样例：

```
private IFaceDetectCallBack mFaceDetectCallback = new IFaceDetectCallBack() {
    @Override
    public void onFaceDetectCallback(FaceModel livenessModel) {
        if (livenessModel == null || livenessModel.getFaceInfo() == null) {
            // 没有检测到人脸
        } else {
            // 检测到人脸
            FaceInfo faceInfo = livenessModel.getFaceInfo();
            // 人脸宽度
            float faceWidth = faceInfo.width;
            // 人脸高度
            float faceHeight = faceInfo.height;
            // 人脸分数
            float faceScore = faceInfo.score;
            // 性别
            BDFaceSDKCommon.BDFaceGender bdFaceGender = faceInfo.gender;
            // 年龄
            int age = faceInfo.age;
            // 是否带眼镜
            BDFaceSDKCommon.BDFaceGlasses bdFaceGlasses = faceInfo.glasses;
        }
    }

    @Override
    public void onFaceDetectStatus(int statusCode, final String statusMsg) {
        // 人脸检测时的状态码和状态信息，例如：statusCode = -20104，statusMsg = 图片比较模糊
        int code = statusCode;
        String msg = statusMsg;
    }

    @Override
    public void onOfflineFaceRecognize(final FaceModel faceModel) {
        if (faceModel != null) {
            // 获取离线识别人脸的用户名称
            String offlineUserName = faceModel.getOfflineRecognizeResult().mName;
        }
    }

    @Override
    public void onFaceDetectDrawCallback(FaceModel livenessModel) {
        // 预留接口，暂不使用
    }
    /**
     * 发现有人脸（没有人脸质量检测）
     * @param livenessModel
     */
    public void onNoQualityDetectCallback(FaceModel livenessModel) {
        // 做人脸质量检测前的人脸数据对象
    }
};
```

获取人脸识别结果 开启人脸识别功能后，想获取在线人脸识别结果，需要实现服务端指令回调接口IDirectiveCallback，实现的接口添加到SDK指令回调接口列表内：

```
IDirectiveCallback directiveCallback = new IDirectiveCallback() {
...
}
// 添加到SDK指令回调接口列表
RobotSDKManager.getInstance().addDirectiveListener(directiveCallback)
```

关于指令回调接口的具体介绍，可以参考 服务端和客户端交互指令说明 。

获取离线人脸识别结果 通过实现IFaceDetectCallBack接口onOfflineFaceRecognize方法获取离线人脸识别结果数据。

```
private IFaceDetectCallBack mFaceDetectCallback = new IFaceDetectCallBack() {
    @Override
    public void onFaceDetectCallback(FaceModel livenessModel) {
        if (livenessModel == null || livenessModel.getFaceInfo() == null) {
            // 没有检测到人脸
        } else {
            // 检测到人脸
            FaceInfo faceInfo = livenessModel.getFaceInfo();
        }
    }
    ...
    ...
    @Override
    public void onOfflineFaceRecognize(final FaceModel faceModel) {
        if (faceModel != null) {
            // 获取离线识别人脸的用户数据对象faceModel
            // 获取离线用户对象的用户名
            String offlineUserName = faceModel.getOfflineRecognizeResult().mName;
        }
    }
};
```

加载离线人脸库数据 1.加载离线人脸库
使用离线人脸识别功能，需要先录入离线人脸库数据，人脸图片路径通过faceModuleConfig.offlineFaceRecognizeImageDir属性配置。

- 属性默认路径为/storage/emulated/0/baidu_robot/UserPicture。

通过RobotSDKManager.getInstance().recordAllFaceFeature()方法完成离线人脸数据的录入，数据录入后会作为离线人脸识别时的离线人脸库使用。

```
/**
* 加载离线人脸识别照片信息
* @param featureType 照片类型，普通照片使用BDFACE_FEATURE_TYPE_LIVE_PHOTO属性
*        BDFaceSDKCommon.FeatureType.BDFACE_FEATURE_TYPE_LIVE_PHOTO（生活照）
*        BDFaceSDKCommon.FeatureType.BDFACE_FEATURE_TYPE_ID_PHOTO（证件照）
* @return
*/
int recordAllFaceFeature(BDFaceSDKCommon.FeatureType featureType);
```

2.离线人脸库图片要求

- 支持jpg和png图片格式
- 图片尺寸 > 640 * 480, 作为离线人脸库底图尽量采用清晰人脸图片

增量更新离线人脸库数据 通过recordDeltaFaceFeature()方法增量更新本地存储离线人脸的数据库。

```
RobotSDKManager.getInstance().recordDeltaFaceFeature()
```

清空离线人脸数据库 通过clearLocalFaceDatabase()清空离线人脸特征的数据库。

```
RobotSDKManager.getInstance().clearLocalFaceDatabase()
```

其他API功能

离线人脸1:1验证

```
 RobotSDKManager.getInstance().offlineCompareFace(@NonNull Bitmap src, @NonNull Bitmap dest,@NonNull IFaceCompareCallback callback)
```

IFaceCompareCallback的定义

```
 /**
 * 人脸比对回调接口
 */
public interface IFaceCompareCallback {
    /**
     * 人脸1:1识别结果回调
     *
     * @param score 对比相似度值（0-100）
     */
    void onCompareSuccess(float score);

    /**
     * 错误事件回调
     *
     * @param errorCode 错误码
     */
    void onCompareError(int errorCode);
}
```

在线人脸比对（1:1）相关接口

```
/**
 * 1：1 图片比对 （生活照）
 * 单次请求，两张图片的大小之和需要小于12M
 *
 * @param faceImg 待比对人脸照片
 * @param dailyPhotoImg 生活照，通常为手机、相机拍摄的人像图片、或从网络获取的人像图片等
 */
public void compareFaceWithDailyPhoto(Bitmap faceImg, Bitmap dailyPhotoImg)

/**
 * 1：1 图片比对 （身份证芯片照）
 * 单次请求，两张图片的大小之和需要小于12M
 *
 * @param faceImg 待比对人脸照片
 * @param idCardPhotoInChip 身份证芯片照，即二代身份证芯片中内置的人像照片
 */
public void compareFaceWithChipIDCard(Bitmap faceImg, Bitmap idCardPhotoInChip)

/**
 * 1：1 图片比对 （带水印证件照）
 * 单次请求，两张图片的大小之和需要小于12M
 *
 * @param faceImg 待比对人脸照片
 * @param docPhotoWithWatermark 带水印证件照，一般为带水印的小图，如公安网小图
 */
public void compareFaceWithWatermarkIDCard(Bitmap faceImg, Bitmap docPhotoWithWatermark)

/**
 * 1：1 图片比对 （普通证件照）
 * 单次请求，两张图片的大小之和需要小于12M
 *
 * @param faceImg 待比对人脸照片
 * @param docPhotoImg 证件照片，如拍摄的身份证、工卡、护照、学生证等证件图片，注：需要确保人脸部分不可太小，通常为100px*100px
 */
public void compareFaceWithDocPhoto(Bitmap faceImg, Bitmap docPhotoImg)
```

状态码以及错误码

错误码/状态码	描述
-20100	未检测到人脸
-20101	人脸左右偏转角度超出限制
-20102	人脸平行平面内的头部旋转角超出限制
-20103	人脸上下偏转角超出限制
-20104	人脸模糊值超出限制
-20105	人脸光照值低于阈值
-20106	人脸左眼遮挡
-20107	人脸右眼遮挡
-20108	人脸鼻子遮挡
-20109	人脸嘴遮挡
-20110	人脸左脸颊遮挡
-20111	人脸右脸颊遮挡
-20112	人脸下巴遮挡

指令交互组件

端云交互指令组件整体介绍

用户通过机器人输入语音、文字、图像等信息后，SDK将输入的信息包装成事件（event）发送给服务器，并接收服务器返回的指令（directive）集，开发者通过传入指令监听回调接口的实现来接收指令集，进行对应的逻辑处理。

- 指令和事件是完成机器人与人交互的最基本的要素，设备端上发生的变化都通过上报相应的事件来通知服务端，服务端通过下发指令给设备端，对用户请求进行响应。

事件（event） 是设备端上报给服务端，通知服务端在设备端发生的事情。比如语音识别到用户说的话，人脸功能检测到人脸等。

指令（directive） 是服务端下发给设备端，设备端需要执行的操作。比如根据语音识别到的用户语义下发相关的语义指令或图文信息，根据人脸识别的图片信息下发人脸图片对应的用户信息等。

指令监听回调接口分类

- IDirectiveCallback 指令回调基础接口，用于接收服务端下发的原始json指令信息，用户可以基于原始指令信息开发业务功能。
- IDirectiveListener 指令监听扩展接口，在指令回调基础接口上，对指令数据进行了分类，简化原始json指令信息解析，每个分类定义了对应的接口，方便用户快速实现业务功能。指令监听扩展接口开源代码可在SampleApp中查找修改。

组件配置

指令组件作为SDK基础组件功能，在SDK初始化时会默认一起完成指令组件的配置对象创建和组件初始化。

获取配置对象

```
DirectiveModuleConfig directiveModuleConfig = new DirectiveModuleConfig();
```

配置组件

```
// 私有化部署环境时，可以设置指令上传的服务端中转地址
directiveModuleConfig.robotRequestHostUrl = "http://172.0.0.1:8080/";
```

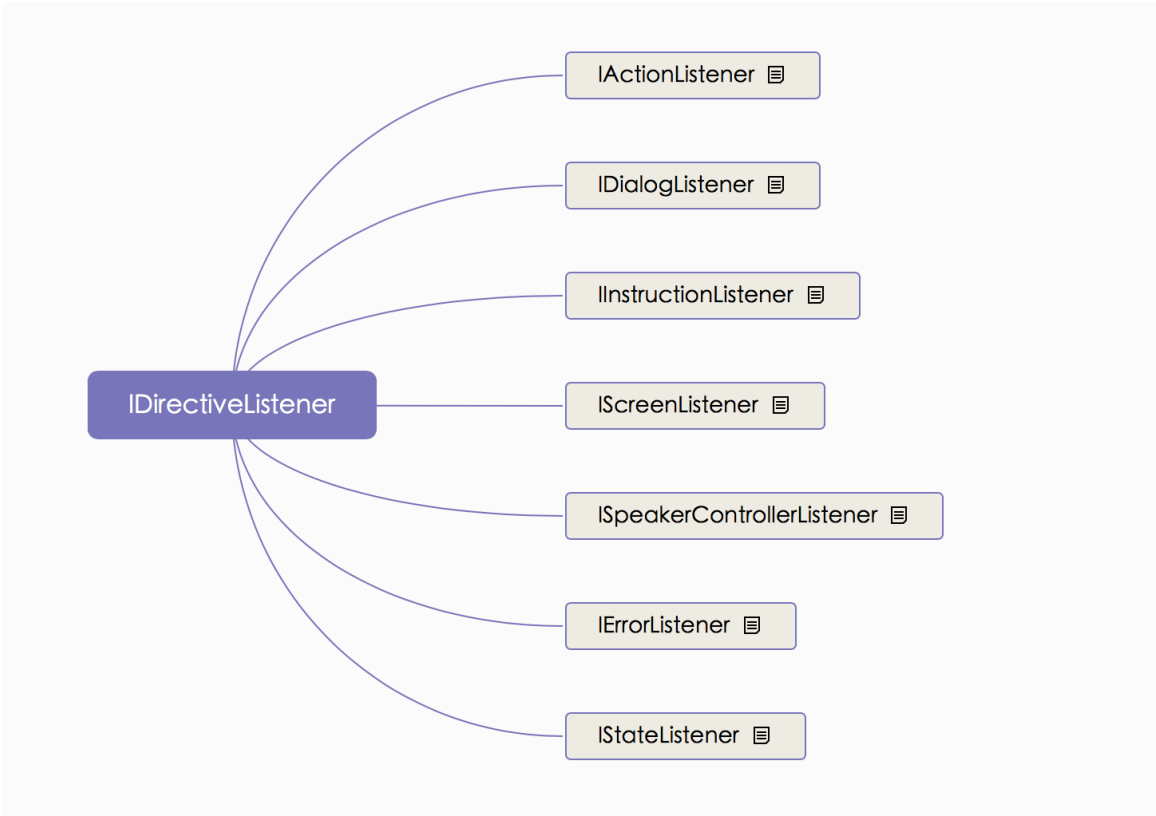
指令回调基础接口 通过实现IDirectiveCallback指令监听回调接口，进行对指令数据回调的监听。

```
/**
 * 指令回调基础接口
 */
public interface IDirectiveCallback {
    /**
     * 服务端指令下发接口
     *
     * @param directiveStatusCode 指令状态码
     * @param directiveStatusMsg  指令状态信息
     * @param directiveModel      指令数据对象
     */
    void onDirectiveReceived(int directiveStatusCode,
                             String directiveStatusMsg,
                             DirectiveModel directiveModel);
}
```

指令数据对象

```
public class DirectiveModel {
    // 请求数据：指令命名空间
    public String requestNamespace = "";
    // 请求数据：指令名称
    public String requestName = "";
    // 请求数据：指令信息ID
    public String requestMessageId = "";
    // 请求数据：指令交互ID用于维护多轮对话
    public String requestInteractionId = "";

    // 返回数据：指令请求状态码
    public int responseStatusCode = 0;
    // 返回数据：指令请求状态信息
    public String responseStatusMsg = "";
    // 返回数据：指令返回原数据json格式
    public String responseDirectives = "";
    // 返回数据：指令返回图片数据
    public byte[] responseByteData = null;
}
```

指令监听扩展接口 目前SDK内置的IDirectiveListener接口如下图：



如图，目前共有七种IDirectiveListener的实现。

开发者需要在SDK中注册自己相应的IDirectiveListener来获取SDK的指令回调，步骤如下：

1. 开发者编写IDirectiveListener的接口实现类

2. 通过RobotSDKManager 的 addDirectiveListeners方法传入 IDirectiveListener 实现类到SDK中

3. 当接口对应事件触发时，在 IDirectiveListener 的回调方法中获得对应数据并实现相应业务逻辑

可以通过以下代码片段来理解这段过程：

```
 // 1. 实现 IDirective 的接口实现类，此处以 IDialogListener 为例
IDialogListener mDialogListener = new IDialogListener() {
        @Override
        public void onVoiceOutput(String content) {
            // 处理语音回复播放相关逻辑
        }

        @Override
        public void onTextOutput(String content) {
            // 处理文字对话显示相关逻辑
        }

        @Override
        public void onHints(List <string> hints) {
            // 处理提示文案显示相关逻辑
        }

    @Override
        public void onEnd() {
            // 对话结束相关逻辑
        }
    };
.....

// 2. 将接口实现类传入 SDK
RobotSDKManager.getInstance().addDirectiveListener(NamespaceGroup.DIALOG, mDialogListener);
```

各个Listener说明如下：

IDialogListener

IDialogListener ：对话相关指令监听器

```
public interface IDialogListener extends IDirectiveListener {
  /**
   * 语音播报指令
   * @param content 需要播报的语音内容
   */
  void onVoiceOutput(String content);

  /**
   * 文字回复展示指令
   * @param content 回复用户的对话内容
   */
  void onTextOutput(String content);

  /**
   * 提示文本展示指令
   * @param hints 需要展示的提示文本 List
   */
  void onHints(List <string> hints);

  /**
   * 对话结束指令
   * 当用户说出对话结束关键字（例如"再见"、"拜拜"）时触发回调
   */
  void onEnd();
}
```

IScreenListener

IScreenListener - 屏幕内容展示相关指令

```
public interface IScreenListener extends IDirectiveListener {

    /**
     * 文本卡片展示指令
     * @param card 文本类型卡片 model
     */
    void onRenderTextCard(TextCard card);

    /**
     * 图片卡片展示指令
     * @param card 图片卡片 model
     */
    void onRenderImageCard(ImageCard card);

    /**
     * 普通列表卡片展示指令
     * @param listCard 普通列表卡片 model
     */
    void onRenderNormalList(ListCard <normalcarditem> listCard);

    /**
     * 图片列表卡片展示指令
     * @param listCard 图片列表卡片 model
     */
    void onRenderSimpleImgList(ListCard<simpleimagecarditem> listCard);

    /**
     * 视频列表卡片展示指令
     * @param listCard 视频列表卡片 model
     */
    void onRenderVideoList(ListCard<videocarditem> listCard);

    /**
     * 用户卡片展示指令
     * @param card 用户卡片 model
     * @param data 用户照片的二进制数据
     */
    void onRenderPerson(UserCard card, byte[] data);

    /**
     * 天气卡片展示指令
     * @param card 天气卡片 model
     */
    void onRenderWeather(WeatherInfoCard card);

    /**
     * 表情展示指令
     * @param expression 表情关键字，例如 Sad、Happy
     */
    void onRenderExpression(String expression);
}
```

IActionListener

IActionListener - 动作控制指令

```
public interface IActionListener extends IDirectiveListener {
    /**
     * 行走指令
```

```
     * @param direction 行走方向
     * @param distance 行走距离
     * @param distanceUnit 距离单位
     */
    void onWalk(String direction, String distance, String distanceUnit);

    /**
     * 转向指令
     * @param direction 转向方向
     * @param angle 转向角度
     * @param angleUnit 角度单位
     */
    void onTurn(String direction, String angle, String angleUnit);

    /**
     * 举手指令
     * @param hands hand字段可为以下内容：hand, right_hand, left_hand, double_hand
     */
    void onRaiseHands(String hands);

    /**
     * 巡航指令
     * @param switchStatus 状态开关
     */
    void onCruise(String switchStatus);

    /**
     * 充电指令
     * @param switchStatus 状态开关
     */
    void onCharge(String switchStatus);

    /**
     * 握手指令
     */
    void onShakeHands();

    /**
     * 拥抱指令
     */
    void onHug();

    /**
     * 摇头指令
     */
    void onTwistHead();

    /**
     * 向左看指令
     */
    void onTurnHeadLeft();

    /**
     * 向右看指令
     */
    void onTurnHeadRight();

    /**
     * 停止运动指令
     */
    void onStop();
```

```
    /**
     * 打招呼指令
     */
    void onWave();

    /**
     * 动一下指令
     */
    void onDoAnAction();
}
```

ISpeakerControllerListener

ISpeakerControllerListener - 扬声器控制指令

```
public interface ISpeakerControllerListener extends IDirectiveListener {
    /**
     * 静音指令
     */
    void onSetMute();

    /**
     * 音量调节指令
     * @param volumeControl
     * @param volumeValue
     */
    void onAdjustVolume(String volumeControl, String volumeValue);
}
```

IInstructionListener

IInstructionListener - 特殊指令

```
public interface IInstructionListener extends IDirectiveListener {

    /**
     * 电量查询指令
     */
    void onInquireEnergy();

    /**
     * 展示功能指令
     */
    void onShowFeatures();
}
```

IErrorListener

IErrorListener - 错误监听

```
public interface IErrorListener extends IDirectiveListener {
    /**
     * 错误监听
     * @param code 错误码
     * @param errorDescription 错误描述
     */
    void onError(int code, String errorDescription);
}
```

IStateListener

IStateListener - 指令处理状态监听

```
public interface IStateListener extends IDirectiveListener {
    /**
     * 当NLU能力识别完成本次对话请求中的用户事件，
     * SDK 开始向开发者回传本次对话请求返回的事件指令（Directives）时回调
     */
    void onDirectiveStart();

    /**
     * 当 SDK 完成了本次对话请求返回的全部事件指令的回传时回调
     */
    void onDirectiveStop();
}
```

错误码及状态码列表

表中为SDK初始化和运行时的错误码和状态码，错误码为负数，状态码为正数

错误码/状态码	描述
10000	SDK初始化成功
10001	SDK语音组件初始化成功
10002	SDK运行状态设置成功
10003	SDK网络状态设置成功
10004	SDKlog设置成功
10005	SDK本地资源设置成功
10006	SDK在线激活成功
10007	SDK获取人脸鉴权文件成功
10008	SDK语音组件初始化成功
10009	SDK人脸组件初始化成功
10010	SDK指令组件初始化成功
20000	3326语音板WIFI已连接
20001	3326语音板WIFI已开启
20002	3326语音板startwakeup 唤醒已经开启
20003	3326语音板startwakeup 正在重试
20004	3326语音板asr 唤醒已经开启
20005	3326语音板asr 正在重试
-10000	SDK初始化失败
-10001	SDK运行状态设置失败
-10002	SDK网络状态设置失败
-10003	SDKlog设置失败
-10004	SDK本地资源设置失败
-10005	SDK在线激活失败
-10006	SDK获取aipe人脸鉴权文件失败
-10007	SDK组件初始化失败

-10008	SDK人脸组件初始化失败
-10009	SDK语音组件初始化失败
-10010	SDK指令组件初始化失败
-10013	语音组件状态码，串口设备为null
-10014	语音组件状态码，IO接口错误
-10015	SDK权限检测异常
-20000	3326语音板WIFI断开连接
-20001	3326语音板串口打开失败
-20002	3326语音板speech service语音读数据失败
-20003	3326语音板串口命令发送失败
-20004	3326语音板3326串口驱动为空
-20005	3326语音板asr识别发生错误
-20006	3326语音板串口通信超时
-20007	3326语音板串口重新连接
-20008	3326语音板speech service 重启，asr重新连接
-20009	3326语音板speech service 重启，wakeup重新连接
-20010	3326语音板speech service 其它服务重新连接
-20011	3326语音板WIFI已关闭
-20012	ASR退出，请稍等，sdk本身包含自动重启ASR机制
-20013	3326语音板ASR退出,正在重启
-20014	3326语音板wakeup正在重试启动中，请勿多次调用
-20015	3326语音板wakeup启动失败，原因是重复开启
-20016	3326语音板asr正在重试启动中，请勿多次调用
-20017	3326语音板asr启动失败，原因是重复开启
-20100	未检测到人脸
-20101	人脸左右偏转角度超出限制
-20102	人脸平行平面内的头部旋转角超出限制
-20103	人脸上下偏转角超出限制
-20104	人脸模糊值超出限制
-20105	人脸光照值低于阈值
-20106	人脸左眼遮挡
-20107	人脸右眼遮挡
-20108	人脸鼻子遮挡
-20109	人脸嘴遮挡
-20110	人脸左脸颊遮挡
-20111	人脸右脸颊遮挡
-20112	人脸下巴遮挡

版本更新记录

版本号	更新时间	更新内容
2.1.5	2022-4-20	1.多模态语音交互方案，实现配置云控 2.增加SDK使用限制策略 3.设备ID算法升级 4.修复已知bug
2.1.2	2022-3-24	1.增加多模态语音交互方案，并且支持戴口罩交互 2.音视频通话功能 3.离线TTS语音播报 4.全新设备指纹算法 5.增加日志上报功能 6.增加消息推送功能
2.1.0	2020-7-07	1.增加语音识别角度抑制功能：只识别正前方特定角度的语音 2.增加语音合成动态打断控制功能：服务端控制打断时机，实现更自然的打断 3.增加语音识别纠错干预功能：在管理控制台配置干预词条，实现语音识别纠错 4.增加语音合成二合一功能：降低语音对话端到端时延 5.升级人脸检测/识别功能，优化人脸检测，离线人脸识别性能，支持离线口罩检测 6.修复已知BUG
2.0.0	2020-3-10	1.SDK功能框架升级，规范功能接口命名，修改部分SDK接口的命名 2.SDK初始化监听接口升级 3.增加语音识别，人脸识别功能独立功能配置对象 4.人脸检测/识别功能升级，提升人脸检测，离线人脸识别性能，修改人脸检测功能回调接口 5.语音识别功能升级，提升语音识别效果 6.优化客户端和服务端交互指令监听接口 7.修复已知BUG
1.0.2	2019-8-10	1.对初始化相关流程进行了说明细化，新增 Sample Application 的简介和链接跳转 2.优化了SDKConfig的方法和接口介绍
1.0.1	2019-5-8	优化了对于人脸识别相关接口的说明
1.0.0	2019-3-10	SDK第一个正式版本
Beta版	2019-2-26	SDK第一个功能验证版本

SDK-V1.0

简介

ABC Robot Android SDK是百度机器人平台面向开发者提供的一套低成本接入和定制化开发方案，通过本SDK可灵活调用平台提供的语音、语义和视觉能力，结合机器人平台管理控制台提供的可视化管理能力，让机器人具备类人的人机交互体验。

SDK面向的主要开发者如下：

类型	描述	场景
本体商	机器人、智能屏幕和XTM终端等泛机器人本体制造商	预置百度机器人平台ABC Robot核心技术和平台能力，方便于其他开发者基于该本体进行二次开发，实现软硬能力的强强结合
集成商	有较强软件开发能力的行业集成商伙伴	基于SDK定制开发适用于各类行业场景的人机交互应用，例如专门做医疗行业辅助诊断的机器人集成商和方案商
企业客户	有较强软件开发能力的企业客户	企业基于平台开发自己的机器人产品，例如某家庭机器人企业，移动营业厅对话式智能终端等。

SDK具备的主要业务功能：

功能	描述
语音唤醒	默认唤醒词是"小度小度"，可付费定制其他唤醒词。如果机器人使用的是百度麦克风阵列，还可获取唤醒角度，实现声源定位
语音识别	通过自定义语音模块，可支持除内置麦克风和百度麦克风阵列之外的第三方麦克风接入的语音识别功能
语音合成	离在线TTS默认支持标准男、标准女、情感男、情感女、米朵和鸽子六种音色
语音对话	语音识别+语义理解+语音合同可实现语音对话功能，通过管理控制台可编排对话逻辑、管理知识库和训练对话模型，含语音指令的解析
人脸检测	人脸特征landmarks及位图信息，表情、性别、年龄等人脸属性信息，人脸情绪信息，以及活体打分，可基于人脸检测实现更自然的唤醒
人脸识别	支持离/在线人脸识别，包括1:1和1:N，SDK端支持本地离线人脸库管理，在线人脸库可通过管理控制台在线管理

集成准备

申请SDK

使用服务之前，请先申请Android SDK。申请SDK请发送正式邮件到robot-bd@baidu.com，并说明以下信息：

- 公司/组织名称
- 项目名称
- 联系人姓名
- 联系人手机号
- 背景说明

获取Client ID、Client Secret

在集成SDK之前，您还需要前往管理控制台 创建一个项目，并在“概览”页面获取该项目的Client ID和Client Secret。

设备配额

我们为您提供了2个试用配额供开发测试使用，1个配额可激活1台设备，您也可以在管理控制台—设备管理页面申请更多配额。

环境准备

将AAR包放入项目的libs文件夹下，位置如下：

```
RobotApp // 开发者的项目根目录
├──── app
│    ├──── build.gradle
│    ├──── libs
│    │    └──── robotsdk.aar <--------- 请把机器人 SDK 的 aar 放到这里
│    ├──── proguard-rules.pro
│    └──── src
├──── build.gradle
├──── gradle
│    └──── wrapper
├──── gradle.properties
├──── gradlew
├──── gradlew.bat
├──── local.properties
└──── settings.gradle
```

兼容性

- minSdkVersion 大于等于 21
- 建议使用最新版本Android Studio进行开发，对Eclipse开发过程中遇到的问题暂不提供支持。

初始化SDK

我们提供了一个简单的集成示例。初次使用，推荐参考 Sample Application 配合官网文档来进行 SDK 的集成。

Sample 中包含了最小程度的集成代码示例，开发者可以直接将 SDK 的 AAR 文件导入 Sample 中运行。 如果 Sample 中各项功能均能够运行成功，则说明硬件和系统环境已经集成完毕。开发者可以参考 Sample 中的代码来使用 SDK 的各项功能。

接下来介绍手动集成 AAR 到项目中的步骤：

集成步骤

1. 在项目的build.gradle中进行依赖配置，配置如下：

```
 android {
   repositories {
      flatDir {
         dir 'libs'
      }
   }
}

dependencies {
   implementation(name: <在这里输入放在libs文件夹下robotsdk的文件名（不含扩展名）>, ext:'aar')
}
```

2. 需要添加的远程依赖如下，如您项目中已经包含则无需添加：

```
    // Android Official libs
   implementation 'com.android.support:appcompat-v7:28.0.0'
   implementation 'com.android.support.constraint:constraint-layout:1.1.3'

   // Third party libs
   implementation 'com.google.android.exoplayer:exoplayer:2.8.2'
   implementation 'com.elvishew:xlog:1.6.1'
   implementation 'com.squareup.okio:okio:1.14.0'
   implementation 'com.squareup.okhttp3:okhttp:3.10.0'
   implementation 'com.alibaba:fastjson:1.2.56'
   implementation 'com.github.bumptech.glide:glide:4.9.0'
```

初始化

配置 SDK 初始化参数

SDK 有大量 configuration 供开发者自行配置，这里列出主要部分：

```
 // SDK配置项
SDKConfig.Builder builder = new SDKConfig.Builder(); // SDKConfig构建对象
builder.context(context) // SDK需要的上下文
    .clientid(<input-client_id-here>) // 设置项目使用的clientId 在云端新建项目时获取
    .clientSecret(<input-client_secret-here>) // 设置项目使用的clientSecret 在云端新建项目时获取
    .sdkType(<input-sdk-type-here>) // SDK 所使用的的功能类型
    .wifiSSID(<input-wifi-ssid-here>) // 设置WiFi账户和密码,使用麦克风阵列时，设置阵列WiFi时使用；使用内置麦克风时不必填写。
    .wifiPWD(<input-wifi-password-here>)
    .wifiType(<input-wlan-secure-type-here>) // 设阵列网络时候要设置的网络安全类型
    .setFaceGroup(<input-face-group-here>) // 设置在线人脸识别使用的人脸库
    .speechServiceType(<input-speech-type-here>) // 设置麦克风输入类型
    .faceAngle(<input-recognize-face-angle-here>) // 使用人脸识别功能时需要调整的角度
    .faceRecognizeType(<input-recognize-type-here>) // 人脸识别功能类型：在线或离线
```

关于参数的具体介绍，可以参考 高级配置项 -> SDKConfig 的接口说明。

创建SDK激活结果的回调监听

初次使用SDK会在初始化时向Server端确认当前ClientID下的机器人应用是否还有可用配额。 开发者需要通过该回调监听激活状态。

```
 RobotSDKEngine.DeviceActivationCallback mActivateCallback = new RobotSDKEngine.DeviceActivationCallback() {
    @Override
    public void onActivateSuccess() {
        Log.d(TAG, "onActivateSuccess: ");
    }

    @Override
    public void onActivateFailed(int errorCode, String errorMsg) {
        Log.d(TAG, "onActivateFailed: " + errorMsg);
    }
};

.....

RobotSDKEngine.getInstance().registerDeviceActivationCallback(mActivateCallback);
```

- SDK 内部对于接口回调实例统一以WeakRefeence形式持有，请请避免在这里传入临时变量或者匿名内部类，开发者需要自己持有回调实例。
- 如果SDK激活失败，请登录管理控制台，确认当前应用还有可用配额。
- 开发者必须在调用 initSDK 之前注册该回调方法，只有在收到成功回调onActivateSuccess()之后，才能开始正常的调用SDK API提供的各项功能。

执行 SDK 的初始化

```
 try {
  RobotSDKEngine.getInstance().initSDK(builder.build());
} catch (Exception e) {
 e.printStackTrace();
}
```

通过调用initSDK方法进行初始化之后，可以在之前注册的DeviceActivationCallback中收到设备激活结果的回调。初始化成功之后就可以开始正式使用SDK了。

获取设备ID

SDK激活完成后，您可以在管理控制台—设备管理的列表中查看到最新激活的设备ID、设备名称等信息。同时，SDK端可以通过调用RobotSDKEngine.getInstance().getSerialNumber()函数获取设备ID，以建立Console端和SDK端的对应关系。

快速上手

概述

ABC-Robot系统的整体交互逻辑：端上面对用户的各种交互或操作，会以“事件”的形式发送到ABC-Robot云端服务，云端服务会根据事件生成响应指令（Directive），将指令下发到客户端。客户端需要根据这些指令，来执行相应的动作。

语音唤醒核心流程

唤醒（WakeUp）是语音交互场景中常用的交互方式。用户通过说出唤醒词向机器人发出通知。机器人在收到唤醒词后会通过唤醒事件回调通知开发者，开发者可以在回调中实现自己的业务逻辑。

唤醒场景操作流程：

1. 机器人摆放在开放大厅场景中，无人交互时，会关闭语音识别功能，以减少无意义的信息输入和网络交互；
2. 当机器人听到用户说出唤醒关键词（例：小度小度），会开启语音识别，正式进入语音可交互状态。与用户进行语音对话；
3. 在适当的时机下（例如用户主动退出，或者长时间没有新的语音对话输入），机器人会关闭语音识别，退出语音可交互状态。回到步骤1，等待下一次唤醒。

操作步骤如下：

语音唤醒注册监听回调

调用RobotSDKEngine.getInstance().registerSpeechListener(SpeechCallBack listener)方法，传入语音交互监听器RobotSDKEngine.SpeechCallBack的接口实现。

开启唤醒词监听

调用RobotSDKEngine.getInstance().startWP()开启唤醒监听，此时麦克风打开，设备开始接收用户语音输入。开发者通过在语音唤醒注册监听回调中注册的回调来获得唤醒相关事件回调。

资源释放

- 在需要关闭语音识别状态时，开发者需要调用RobotSDKEngine.getInstance().stopWP()
- 在需要停止回调监听时，开发者需要调用RobotSDKEngine.getInstance().unRegisterSpeechListener(SpeechCallBack callBack)来移除相应的监听回调。

语音对话核心流程

语音对话流程举例如下：

1. 机器人开启语音识别，用户向机器人发起对话，麦克风阵列开始接受用户的语音输入数据；
2. 由 SDK 的 ASR 能力将语音输入转成对应的文字输入（Text），并通过语音监听回调通知给开发者；
3. 同时 SDK 利用云端的自然语言理解（NLU）能力将分析出用户的文字输入（Text）对应的意图，云端中控会生成响应此意图的指令（Directive），通过指令回调将指令通知给开发者。

实现一个语音对话流程的核心步骤如下：

语音对话注册监听回调

- 调用RobotSDKEngine.getInstance().addDirectiveListener()方法，根据产品业务场景传入意图识别指令监听器。目前SDK内置了六种指令监听接口，开发者可以根据自己的业务场景进行对应实现。详见指令回调处理。
- 调用RobotSDKEngine.getInstance().registerSpeechListener(SpeechCallBack listener)方法，传入语音对话事件监听器RobotSDKEngine.SpeechCallBack的接口实现。

开启语音识别

调用RobotSDKEngine.getInstance().startListening()开启ASR语音识别，此时麦克风打开，设备开始接收用户语音输入。开发者通过在语音对话注册监听回调中注册的回调来获得语音输入的ASR识别返回，并进行意图识别指令处理。

资源释放

- 在需要关闭语音识别状态时，开发者需要调用RobotSDKEngine.getInstance().stopListening()。
- 在需要停止回调监听时，开发者需要调用
  - RobotSDKEngine.getInstance().unRegisterSpeechListener(SpeechCallBack callBack)
  - RobotSDKEngine.getInstance().removeDirectiveListener(IDirectiveListener listener)来移除相应的监听回调。

人脸识别核心流程

人脸识别的常用场景举例：

> 1. 开发者在合适的时机调起人脸识别，机器人开启摄像头开始接受图像数据输入；
> 2. 由 SDK 的人脸检测能力从视频流中检测到含有完整人脸的图片帧，并对该人脸进行人脸识别；
> 3. 如果该人脸已在人脸库注册，则回传完整的身份信息，否则则通知开发者这是一张未经注册的人脸；

实现一个人脸识别流程的核心步骤如下：

人脸识别注册监听回调

- 调用RobotSDKEngine.getInstance().addDirectiveListener()方法，根据产品业务场景传入意图识别指令监听器。目前SDK内置了六种指令监听接口，开发者可以根据自己的业务场景进行对应实现。详见指令回调处理。
- 调用RobotSDKEngine.getInstance().registerFaceListener(RecognizeListener listener)方法，传入人脸识别事件监听器RobotSDKEngine.RecognizeListener的接口实现。

开始人脸识别

调用RobotSDKEngine.getInstance().startFaceRecognize(SurfaceView surfaceView, @FaceTaskType int taskType, CameraConfig config)开始人脸识别，开发者传入一个用于展示摄像头镜像画面的SurfaceView实例，SDK会开启摄像头自动检测相机帧流中包含的人脸并进行识别。

资源释放

- 在需要停止人脸识别时，开发者需要调用RobotSDKEngine.getInstance().stopTracking()。
- 在需要停止回调监听时，开发者需要调用
  - RobotSDKEngine.getInstance().unRegisterFaceListener()
  - RobotSDKEngine.getInstance().removeDirectiveListener(IDirectiveListener listener)来移除相应的监听回调。

语音交互

语音对话

该功能在SDK初始化之后进行，需要这个功能时，首先需调用函数registerSpeechListener注册监听器。之后开发者需实现本SDK提供的语音会话监听器接口（RecognizeListener），通过监听器中的回调函数来使用语音识别相关功能。本SDK会对语音识别结果，Q&A结果通过监听器的函数一一通知给开发者。如果对返回的答案的格式有特殊要求，可通过机器人平台完成智能对答内容（json内容）格式的定制化。

1. `public void registerSpeechListener(SpeechCallBack callBack)`：注册语音识别监听器 参数：`SpeechCallBack callBack`：一个语音识别的listener，详情点击这里(SpeechCallBack介绍)
2. `public void unRegisterSpeechListener(SpeechCallBack callBack)`：注销语音识别监听器 参数：`SpeechCallBack callBack`：一个语音识别的listener，详情点击这里(SpeechCallBack介绍)
3. `public void startListening()` ：语音识别开启函数
4. `public void stopListening()` ：语音识别关闭函数
5. `public boolean isAsrRunning()`：判断语音识别是否正在执行。 输出：True 正在执行 ； False 没有工作
6. `public void startConversation()` ：开始当前对话上下文（环境），即一次对话过程。
7. `public void resetConversation()`：取消当前对话上下文（环境），即初始化当前对话环境。
8. `public void switchLanguage(int languageType)`：切换识别语音的语种（目前支持中文、英文两种），必须在开启语音识别之后调用，具体参数如下： `public static final int LAN_CHINESE = 0;`：中文 `public static final int LAN_ENGLISH = 1;`：英文
9. `public void adjustVolume(int direction)`：调节播放tts文字的音量大小(目前仅支持3326方案)，必须在开启语音识别之后调用，具体参数如下： `public static final int VOLUME_ADD = 0x0100;`：增大音量 `public static final int VOLUME_SUB = 0x0200;`：减小音量

> **注意：**
> 当实现项目没有用到人脸识别模块时，可通过startConversation函数和resetConversation函数控制对话的上下文，即代表一次session会话。在调用startConversation () 方法之后到调用resetConversation()方法之前的这段时间里，语音功能模块会把使用者当做是同一个人，并且语境相同。调用resetConversation()方法之后，被认为当前对话模式结束。

语音合成

该功能在SDK初始化之后进行。

1. `public void speak(String speech)`：TTS播报字符串。 参数： `String speech`：播报的字符串。
2. `public void stopSpeaking()` ：TTS停止当前正在播报的字符串。

语音唤醒

1. `public void startWP()` ：开启唤醒功能，唤醒词设置点击这里。
2. `public void stopWP()` ：关闭唤醒词识别功能。

```
 public class WakeUpActivity implements RobotSDKEngine.SpeechCallBack
{
 private static final String TAG = logTag(RobotSDKEngine.class);
 private RobotSDKEngine robotSDKEngine;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_wake_up);
  // 获取SDK实例
  robotSDKEngine = RobotSDKEngine.getInstance();
  // 注册语音模块回调，this为当前类。
  robotSDKEngine.registerSpeechListener(this);
  // 开启语音识别
  robotSDKEngine.startListening();
  // 开启唤醒
  robotSDKEngine.startWP();
  // 语音合成
  robotSDKEngine.speak("欢迎使用机器人SDK");
 }
 @Override
 public void onError(int errorCode) {
  Log.d(TAG, "错误码：" + errorCode);
 }
 @Override
 public void onStatus(int status) {
  Log.d(TAG, "麦克风状态：" + status);
 }
 // asr识别结果回调
 @Override
 public void onNewSpeech(SpeechBean speechBean) {
  String speech = speechBean.speech;
        boolean isFinal = speechBean.isFinal;
  Log.d(TAG, "语音识别结果：" + speech);
  Log.d(TAG, "是否当前识别结束：" + isFinal);
  if (isFinal == ture) {
   if ("再见".equals(speech)) {
    robotSDKEngine.resetConversation();
   }
  }
 }

 // 唤醒回调，返回唤醒词
 @Override
 public void onWakeUp(String word) {
  Log.d(TAG, "唤醒词为：" + word);
  // 开启语音会话（上下文）
  robotSDKEngine.startConversation ();
  robotSDKEngine.speak("欢迎你");
 }
 // 唤醒角度回调
 @Override
 public void onWakeAngle(int angle) {
  Log.d(TAG, "角度：" + angle);
 }
 @Override
 protected void onDestroy() {
  super.onDestroy();
 }
}
```

语音交互监听器

Interface SpeechCallBack：语音识别器接口。具有五个回调函数：

1. public void onStatus(int status)：当前语音识别器状态 参数： int status：值为2时，表示当前不可用。其他值代表语音识别的不同状态 麦克风状态回调表：

状态名称	状态值	类别	麦克风类别
停止（WpStop）	203	唤醒	内置/阵列
准备就绪（WpSready）	202	唤醒	内置/阵列 （在阵列中只有经过一次停止唤醒之后，以后再开启就会有回调）
准备开始（AsrBegin）	5	识别	内置
结束（onAsrEnd）	6	识别	内置
结束（onAsrFinish）	7	识别	内置
最终识别结果（AsrFinalResult）	7	识别	内置
错误（AsrFinishError）	7	识别	内置
长语音识别结束（AsrLongFinish）	7	识别	内置
退出（AsrExit）	2	识别	内置/阵列
完成（TTS_FINISHED）	302	合成	内置/阵列
开始（TTS_START）	301	合成	阵列

2. public void onNewSpeech(SpeechBean speechBean)：SDK通过该回调函数，返回 语音识别的结果。SpeechBean 具有三个属性： int id：此段语音id String speech：语音识别到的字串 boolean isFinal：是否这句话识别结束，逐字识别模式下，当本句话还未识别结束回调的值为false，识别结束为true。

3. public void onWakeUp(String word)：SDK检测到唤醒词后回调该方法，并返回唤醒词字串。

   参数： String word：唤醒词

4. public void onWakeAngle(int angle)：SDK检测到唤醒词后回调该方法，并返回唤醒角度（当时用具有声源定位的麦克风阵列的时候，功能才有效）。

   参数： int angle：0~360之间（逆时针方向）

5. public void onError(int errorCode)：SDK检测到语音识别器发生错误时回调用该方法 参数：

   int errorCode：错误码为-2时，表示WiFi已经断开；错误码为-3时，表示麦克风阵列开启串口失败；错误码为-8时，表示麦克风阵列语音识别失败。

语音模块错误回调

状态名称	状态值	类别	麦克风类别
阵列的WiFi已经断开	-2	语音	阵列
麦克风阵列开启串口失败	-3	语音	阵列
解析麦克风阵列数据失败	-4	语音	阵列
SDK发送麦克风数据失败	-5	语音	阵列
麦克风驱动找不到	-6	语音	阵列
麦克风阵列 设备找不到	-7	语音	阵列
麦克风阵列语音识别失败	-8	语音	阵列
Android设备断网	-9	语音	内置（因为设备断网触发的语音识别错误回调）

自定义语音模块

本SDK还支持除内置麦克风和百度麦克风阵列之外的第三方麦克风的接入，具体接入过程如下：

1. 设置麦克风类型为`SDKConfig.SPEECH_TYPE_CUSTOMER`

```
public Class MyApplication extends Application { // 建议在 Application 中完成 SDK 初始化

    @Override
    public void onCreate() {
        super.onCreate();
        // SDK配置项
        SDKConfig.Builder builder = new SDKConfig.Builder();
        builder.faceApiKey(BuildConfig.FACE_KEY)
                .context(getApplicationContext())
                // 在这里设置麦克风类型为SDKConfig.SPEECH_TYPE_CUSTOMER
                .speechServiceType(SDKConfig.SPEECH_TYPE_CUSTOMER)
                .……;
        // 省略具体配置代码，初始化具体配置项代码可参考第三章节 - 3.1 引擎初始化
    }
}
```

2. 实现第三方语音类（基于代理模式，继承SDK提供的BaseSpeech类，并实现必要的方法）

```
public class CustomSpeech extends BaseSpeech {
    // 开发者需要在业务逻辑中自己编写自己的语音能力实现类
    // 此处以 CustomSpeechImp 举例，意指用户自定义的语音能力实现类
    CustomSpeechImp mSpeechImp;

    public CustomSpeech(CustomSpeechImp speechImp) {
        this.mSpeechImp = speechImp;
    }

    @Override
    public void queryConfiguration() {

    }

    @Override
    public void startListening() {
        mSpeechImp.startListening();
    }

    @Override
    public void cancelListening() {
        mSpeechImp.cancelListening();
    }

    @Override
    public void stopListening() {
        mSpeechImp.stopListening();
    }

    @Override
    public void startWakeUp() {
        mSpeechImp.startWakeUp();
    }

    @Override
    public void stopWakeUp() {
        mSpeechImp.stopWakeUp();
    }

    @Override
    public void speak(String speech) {
        mSpeechImp.speak(speech);
    }

    @Override
    public void stopSpeaking() {
        mSpeechImp.stopSpeaking();
    }
}
```

3. 把第三方语音服务run起来，并实现一个SpeechRecListener类的对象，最后在第三方语音服务的回调处添加SpeechRecListener的触发。 初始化一个开发者自定义的BaseSpeech类型对象、在SDK中注册、新建一个语音识别事件监听类，如下：

```
@Override
public void onCreate() {
  super.onCreate();
  // 1. 初始化开发者自定义的BaseSpeech类型对象
  mSpeech = new CustomSpeech(this);
  // 2. 注册开发者自定义的 CustomSpeech
  RobotSDKEngine.getInstance().registerSpeech(getApplicationContext(), mSpeech);
  // 3. 初始化语音识别事件监听器
  mListener = new SpeechRecListener(mSpeech);
}
```

4. 并在第三方语音服务的响应位置添加通知处理：

```
// 识别的语音通过该函数传递给语音监听器来通知开发者
// 通常会在语音监听器的 onNewSpeech(SpeechBean bean) 方法中进行回调
// msgData 为 SpeechBean 中的 speech 属性
mListener.onAsrFinalResult(msgData, null);
```

```
// 语音唤醒的结果通过该函数传递给语音监听器来通知开发者
// 该回调通常会触发语音监听器的 onWakeUp(String word) 方法
// msgData 与 word 对应，同为唤醒词
mListener.onWpSuccess(msgData);
```

5. 还可以继续拓展其他函数：

```
// 1. 识别的中间过程回调，对应onNewSpeech(SpeechBean speechBean)中，beand的isFinal属性为false的场景：
mListener.onAsrPartialResult(String result, RecogResult recogResult);
// 2. 唤醒角度回调，通过语音监听器的onWakeAngle(int angle)通知开发者
mListener.onWpAngle(int angle)
// 3. 错误状态回调，通过语音监听器的onError(int errorCode)通知开发者
mListener.onError(int errorCode, String errorMsg)
```

> **注意：**
>
> 以上注册要在下面这两个开启和注册语音语音之前完成。

```
robotSDKEngine.stopListening();
robotSDKEngine.registerSpeechListener(this);
```

人脸识别

人脸识别使用流程

当需要使用人脸识别时：

1. 调用`RobotSDKEngine.getInstance.startFaceRecognize()`来启动人脸识别任务，
2. 调用`RobotSDKEngine.getInstance.registerFaceListener(RecognizeListener listener)`传入`RobotSDKEngine.RecognizeListener`的实现来获取**本地人脸检测**的结果；
3. 调用`RobotSDKEngine.getInstance().addDirectiveListener()`，传入Screen类型 DirectiveListener 监听器实现，在`onRenderPerson(UserCard card, byte[] data)`回调中获取**在线人脸识别**的结果，详见指令回调处理（请联系我们进行在线人脸库的人脸信息添加）；
4. SDK支持配置人脸识别所使用的人脸 face group，需要调用`SDKConfig`的`Builder`类的`setFaceGroup(String faceGroup)`方法；

当需要结束人脸识别时：

1. 调用RobotSDKEngine.getInstance.unRegisterFaceListener()来取消人脸识别任务的回调监听，并移除监听器实例；

2. 调用RobotSDKEngine.getInstance.stopTracking()：结束人脸识别任务；

接下来对方法进行详细说明：

```
/**
 * 开始一次人脸识别，开发者需自行指定摄像头配置参数
 *
 * @param surfaceView 摄像头镜像展示所需要的view, 需要开发者自行在应用层维护其内存和生命周期。
 * @param taskType 任务类型，目前支持：
 *          RobotSDKEngine.TASK_FACE_LOGIN：人脸登录任务
 *          RobotSDKEngine.TASK_FACE_RECOGNIZE：人脸识别任务
 * @param config 摄像头配置参数，CameraConfig 具体说明见下文
 * @return 摄像头打开成功则返回true，反之则为false
 */
public boolean startFaceRecognize(SurfaceView surfaceView,
                          @FaceTaskType int taskType,
                          CameraConfig config);



/**
 * 开始一次人脸识别，SDK使用初始化时传入的默认的摄像头配置参数
 *
 * @param surfaceView 摄像头镜像展示所需要的view, 需要开发者自行在应用层维护其内存和生命周期。
 * @param taskType 任务类型，目前支持：
 *          RobotSDKEngine.TASK_FACE_LOGIN：人脸登录任务
 *          RobotSDKEngine.TASK_FACE_RECOGNIZE：人脸识别任务
 * @return 摄像头打开成功则返回true，反之则为false
 */
public boolean startFaceRecognize(SurfaceView surfaceView,
                          @FaceTaskType int taskType);
```

CameraConfig方法列表如下：

```
 /**
 * 构造方法，初始化一个摄像头参数
 * @param displayRotation 摄像头镜像画面的展示方向
 * @param cameraType 接入的摄像头类型
 * @param cameraParam 摄像头配置参数
 */
public CameraConfig(int displayRotation, int cameraType, String cameraParam)

/**
 * 通过SDK初始化时配置的参数获取默认的 CameraConfig
 * @return 默认CameraConfig 实例
 */
public static CameraConfig getDefault()
```

参数名	说明	可选值
displayRotation	SurfaceView中预览画面的显示角度	可选的参数值分别为0、1、2、3； 分别对应的为view控件展示的镜像逆时针旋转0°、90°、180°、270°
cameraType	相机类型	AbstractCamera.TYPE_RTSP ：网络摄像头 AbstractCamera.TYPE_USB ：外置USB摄像头 AbstractCamera.TYPE_INTERNAL_FRONT ：前置摄像头 AbstractCamera.TYPE_INTERNAL_REAR ：后置摄像头
cameraParam	相机配置参数	1、当摄像头类型为内置时：此参数设置为：0代表后置，1代表前置； 2、当摄像头为USB Camera类型时，此参数应该为usb camera的设备号

人脸逻辑调用代码示例：

```
public class FaceTestActivity implements RobotSDKEngine.RecognizeListener {

  private static final String TAG = "FaceTestActivity";

  public static final CameraConfig CAMERA_CONFIG
        = new CameraConfig(2, AbstractCamera.TYPE_INTERNAL_REAR, "0");

  // 在线人脸识别结果回调
  private class UserCardHandler implements IScreenListener {
     @Override
     public void onRenderPerson(UserCard card, byte[] data) {
        String name = card.getName();
        // todo
     }

     // other call back method goes here ...
  };

  private UserCardHandler mUserCardHandler = new UserCardHandler();

  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_wake_up);
     RobotSDKEngine.getInstance().startFaceRecognize(surfaceView, TASK_FACE_LOGIN, CAMERA_CONFIG);
     RobotSDKEngine.getInstance().registerFaceListener(callback);
     RobotSDKEngine.getInstance().addDirectiveListener(NamespaceGroup.SCREEN, mUserCardHandler);
  }

  @Override
  protected void onDestroy() {
     super.onDestroy();
     RobotSDKEngine.getInstance().stopTracking();
     RobotSDKEngine.getInstance().unRegisterFaceListener();
     RobotSDKEngine.getInstance().removeDirectiveListener(NamespaceGroup.SCREEN, mUserCardHandler);
  }

  @Override
  public void onNewFace(FaceInfos faceInfos, Bitmap faceImage) {

  }

  // 查询到人脸相关信息的回调。
  // 仅离线人脸识别会通过本接口回调数据
  // 开发者需要实现基于指令的监听器以获得完整回调信息，详见”数据回调处理”
  @Override
  public void onFaceResult(int faceid, String personInfo) {
```

```
    public void onFaceResult(int faceid, String personInfo) {
        String userName = "";
        if (personInfo != null) {
        JSONObject obj = JSONObject.parseObject(personInfo);
        int errorCode = obj.getIntValue("error_code");
         if (errorCode == 0) {
          if (obj.containsKey("customer")) {
           obj = obj.getJSONObject("customer");
           if (obj.containsKey("name")) {
            userName = obj.getString("name");
            if (userName == null || "null".equals(userName)) {
             userName = "";
            }
           }
          }
         }
        }
    }

    // 人脸移出回调，人脸从画面（摄像头镜像）中消失的时候触发
    @Override
    public void onFaceOut(int faceid) {
        Log.d(TAG, "人脸移出摄像头，给出人脸的id号：" + faceid);
    }

    // 人脸检测出现错误的时候触发
    @Override
    public void onError(int errorCode) {
    }
}
```

人脸检测事件监听器接口说明

`RecognizeListener` 为人脸识别监听器接口。具有如下回调函数：

public void onNewFace(FaceInfos faceInfos, Bitmap faceImage)

SDK通过摄像头检测到有人脸进入时回调该方法。
参数：
`FaceInfos faceInfos` 人脸信息，详情参考FaceInfos。
`Bitmap faceImage` 裁剪后的人脸图片（含背景）

public void onFaceTracker(FaceInfos faceInfos)

SDK通过摄像头检测每一帧图片中的人脸回调，当前图片帧有人脸存在时会回调该方法。
参数：

`FaceInfos faceInfos` 人脸信息，详情参考FaceInfos。

public void onFaceResult(int faceid, String personInfo)

SDK识别出onNewFace函数检测到的人脸的具体信息时回调该方法，返回人脸id和详细信息。
参数：
`int faceid`：人脸ID
`String personInfo`：personInfo 的返回值有如下情况：

1. 人脸匹配到人脸库中人脸且相似度分数高于 SDKConfig.setOfflineFaceRecogScore 中设置的阈值时：

```
 // name 字段对应本地人脸入库时，人脸照片的文件名，score 字段对应相似度分数；
"{'name':'<人脸照片文件名>', 'score':'<相似度分数>'}"
```

2. 当前人脸匹配到人脸库中的人脸但相似度分数低于 SDKConfig.setOfflineFaceRecogScore 中设置的阈值时：

```
 // name 字段对应一个空白字符串，score 字段对应相似度分数；
"{'name':'', 'score':'<相似度分数>'}"
```

3. 当前人脸没有匹配到人脸库中的人脸时：

```
 // name 字段对应一个空白字符串，score 字段为 -1 ；
{'name':'', 'score':'-1'}
```

public void onFaceOut(int faceid)

SDK检测到人脸离开摄像头能捕捉到的区域时回调用该方法

参数：

`int faceid`：人脸ID

`public void onError(int errorCode)`：发生错误时回调用该方法，并给出错误码

参数：

`int errorCode`：错误码，当errorCode为-1是表示face认证失败，请检查人脸认证信息。

状态名称	状态值	类别
face认证失败	-1	人脸

人脸FaceInfos说明

参数名称	类型	详细说明
faceInfo	FaceInfo	人脸基本信息
faceAttribute	BDFaceSDKAttribute	人脸基本属性
faceEmotion	BDFaceSDKEmotions	人脸表情信息
liveScore	float	活体验证得分

FaceInfo

参数名称	类型	详细说明
mWidth	float	人脸的宽度 （方形），与长度相等
mAngle	float	平面内旋转角[-180(逆时针), 180(顺时针)]
mCenter_y	float	人脸中心点Y值
mCenter_x	float	人脸中心点X值
mConf	float	人脸置信度
landmarks	int[]	4个关键点位置，左眼中心、右眼中心、鼻尖、嘴中心
face_id	int	Id值
headPose	float[]	俯仰角：headPose[0]代表抬头低头角度阈值，中间为0，上下为正负；headPose[1]代表左右角度阈值，中间为0左正右负、headPose[2]代表顺逆时针角度阈值
illum	float	人脸亮度
blur	float	人脸模糊度
occlu	float[]	人脸模糊度：0~6（不确保全部出现）分别代表左眼、右眼、鼻子、嘴巴、左脸、右脸、下巴的遮挡程度

BDFaceSDKAttribute

参数名称	类型	详细说明
age	float	年龄
race	BDFaceRace	种族（枚举类型）依次为：黄种人、白种人、黑种人、印度人
emotion	BDFaceEmotion	情绪（枚举类型）依次为：无表情、微笑、大笑
glasses	BDFaceGlasses	佩戴眼镜状况（枚举类型）依次为：不带眼镜、普通透明眼镜、墨镜
gender	BDFaceGender	性别（枚举类型）依次为：女、男

BDFaceSDKEmotions

参数名称	类型	详细说明
emotion	BDFaceEmotionEnum	表情（枚举类型）依次为：生气、厌恶、害怕、开心、伤心、惊讶、无表情
expression_conf	float	当前表情的置信度 （取值范围：0~1）
expression_conf_list	float[]	当前表情的置信度列表

离线人脸识别相关接口

- `RobotSDKEngine.getInstance().recordAllFaceFeature()`：
  全量更新本地存储离线人脸特征的数据库
  所有图片需要保存在如下目录：/sdcard/baidu_robot/UserPicture/
- `RobotSDKEngine.getInstance().recordDeltaFaceFeature()`：
  增量更新本地存储离线人脸特征的数据库
  所有图片需要保存在如下目录：/sdcard/baidu_robot/UserPicture/
- `RobotSDKEngine.getInstance().clearLocalFaceDatabase()`：清空离线人脸特征的数据库
- `RobotSDKEngine.getInstance().offlineCompareFace(Bitmap src, Bitmap dest, OfflineFaceCallBack callBack)`：离线人脸1:1验证，src，dest分别表示需要对比的两张图片，传入的图片大小必须小于1M，callback参数为对比结果回调，这个方法必

须等到callback有回调返回之后才能进行下一次调用这个方法，

```
/**
 *
 * 离线人脸1：1比对，图片必须小于1M
 *
 * @param src 第一张图片
 * @param dest 第二张图片
 * @param callBack 回调结果
 * @throws Exception: Throws Exception if :
 *      <p>1. last call of this function isn't finished<p>
 *      <p>2. callback is null<p>
 */
public void offlineCompareFace(@NonNull Bitmap src,
                               @NonNull Bitmap dest,
                               @NonNull OfflineFaceCallBack callBack) throws Exception
```

RobotSDKEngine.OfflineFaceCallBack定义如下：

```
/**
 * 离线人脸监听器接口
 */
public interface OfflineFaceCallBack {
    /**
     * 人脸1:1识别结果回调
     *
     * @param score 对比相似度值（0-100）
     */
    public void onCompareSuccess(float score);

    /**
     * 错误事件回调
     *
     * @param errorCode 错误码
     */
    public void onOfflineError(int errorCode);
}
```

在线人脸比对（1:1）相关接口

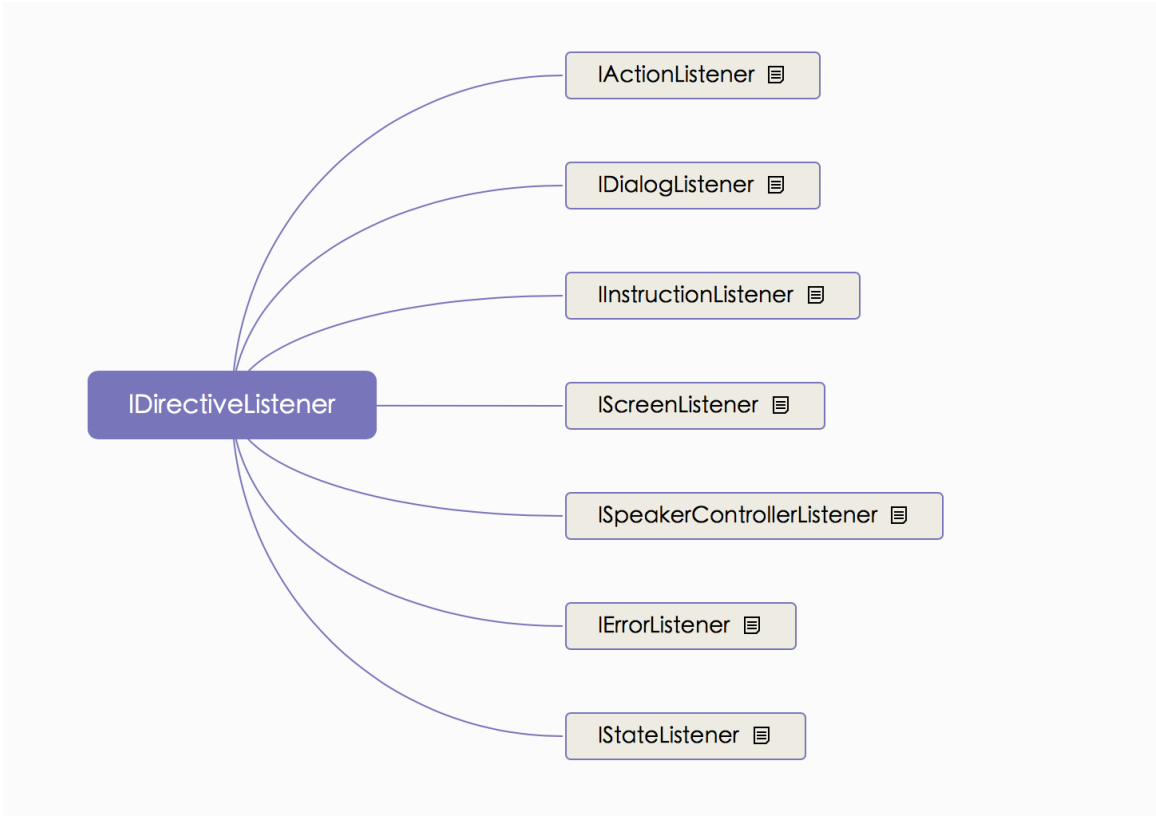
以下方法均可通过`RoboSDKEngine`调用,需要注意两张图片的大小之和需要小于12M：

```
    /**
     * 1：1 图片比对 （生活照）
     * 单次请求，两张图片的大小之和需要小于12M
     *
     * @param faceImg 待比对人脸照片
     * @param dailyPhotoImg 生活照，通常为手机、相机拍摄的人像图片、或从网络获取的人像图片等
     */
    public void compareFaceWithDailyPhoto(Bitmap faceImg, Bitmap dailyPhotoImg)

    /**
     * 1：1 图片比对 （身份证芯片照）
     * 单次请求，两张图片的大小之和需要小于12M
     *
     * @param faceImg 待比对人脸照片
     * @param idCardPhotoInChip 身份证芯片照，即二代身份证芯片中内置的人像照片
     */
    public void compareFaceWithChipIDCard(Bitmap faceImg, Bitmap idCardPhotoInChip)

    /**
     * 1：1 图片比对 （带水印证件照）
     * 单次请求，两张图片的大小之和需要小于12M
     *
     * @param faceImg 待比对人脸照片
     * @param docPhotoWithWatermark 带水印证件照，一般为带水印的小图，如公安网小图
     */
    public void compareFaceWithWatermarkIDCard(Bitmap faceImg, Bitmap docPhotoWithWatermark)

    /**
     * 1：1 图片比对 （普通证件照）
     * 单次请求，两张图片的大小之和需要小于12M
     *
     * @param faceImg 待比对人脸照片
     * @param docPhotoImg 证件照片，如拍摄的身份证、工卡、护照、学生证等证件图片，注：需要确保人脸部分不可太小，通常为100px*100px
     */
    public void compareFaceWithDocPhoto(Bitmap faceImg, Bitmap docPhotoImg)
```

指令回调处理

用户输入语音或者文字信息后，SDK会根据上下文识别出用户的意图，并返回对应的指令（directive），开发者通过传入指令监听器（IDirectiveListener）的实现来接收意图指令，并进行对应逻辑处理。

目前SDK内置的IDirectiveListener接口如下图：



如图，目前共有八种IDirectiveListener的实现。

开发者需要在SDK中注册自己相应的IDirectiveListener来获取SDK的指令回调，步骤如下：

1. 开发者编写IDirectiveListener的接口实现类
2. 通过RobotSDKEngine 的 addDirectiveListeners/addDirectiveListener方法传入 IDirectiveListener 实现类到SDK中
3. 当接口对应事件触发时，在 IDirectiveListener 的回调方法中获得对应数据并实现相应业务逻辑

可以通过以下代码片段来理解这段过程：

```
 // 1. 实现 IDirective 的接口实现类，此处以 IDialogListener 为例
IDialogListener mDialogListener = new IDialogListener() {
        @Override
        public void onVoiceOutput(String content) {
            // 3. 处理NLU解析返回的指令...
            // 处理语音回复播放相关逻辑
        }

        @Override
        public void onTextOutput(String content) {
            // 3. 处理NLU解析返回的指令...
            // 处理文字对话显示相关逻辑
        }

        @Override
        public void onHints(List <string> hints) {
            // 3. 处理NLU解析返回的指令...
            // 处理提示文案显示相关逻辑
        }

    @Override
        public void onEnd() {
            // 3. 处理NLU解析返回的指令...
            // 对话结束相关逻辑
        }
     };

......

// 2. 将接口实现类传入 SDK
RobotSDKEngine.getInstance().addDirectiveListener(mDialogListener);
RobotSDKEngine.getInstance().addDirectiveListener(NamespaceGroup.DIALOG, mDialogListener);</string>
```

各个Listener说明如下：

IDialogListener

IDialogListener ：对话相关指令监听器

```
public interface IDialogListener extends IDirectiveListener {
    /**
     * 语音播报指令
     *
     * @param content 需要播报的语音内容
     */
    void onVoiceOutput(String content);

    /**
     * 文字回复展示指令
     *
     * @param content 回复用户的对话内容
     */
    void onTextOutput(String content);

    /**
     * 提示文本展示指令
     *
     * @param hints 需要展示的提示文本 List
     */
    void onHints(List <string> hints);

    /**
     * 对话结束指令
     *
     * 当用户说出对话结束关键字（例如"再见"、"拜拜"）时触发回调
     *
     */
    void onEnd();
}</string>
```

IScreenListener

IScreenListener - 屏幕内容展示相关指令

```
public interface IScreenListener extends IDirectiveListener {

    /**
     * 文本卡片展示指令
     * @param card 文本类型卡片 model
     */
    void onRenderTextCard(TextCard card);

    /**
     * 图片卡片展示指令
     * @param card 图片卡片 model
     */
    void onRenderImageCard(ImageCard card);

    /**
     * 普通列表卡片展示指令
     * @param listCard 普通列表卡片 model
     */
    void onRenderNormalList(ListCard <normalcarditem> listCard);

    /**
     * 图片列表卡片展示指令
     * @param listCard 图片列表卡片 model
     */
    void onRenderSimpleImgList(ListCard<simpleimagecarditem> listCard);

    /**
     * 视频列表卡片展示指令
     * @param listCard 视频列表卡片 model
     */
    void onRenderVideoList(ListCard<videocarditem> listCard);

    /**
     * 用户卡片展示指令
     * @param card 用户卡片 model
     * @param data 用户照片的二进制数据
     */
    void onRenderPerson(UserCard card, byte[] data);

    /**
     * 天气卡片展示指令
     * @param card 天气卡片 model
     */
    void onRenderWeather(WeatherInfoCard card);

    /**
     * 表情展示指令
     * @param expression 表情关键字，例如 Sad、Happy
     */
    void onRenderExpression(String expression);
}</videocarditem></simpleimagecarditem></normalcarditem>
```

IActionListener

IActionListener - 动作控制相关指令

```
public interface IActionListener extends IDirectiveListener {
    /**
     * 行走指令
     * @param direction 行走方向
     * @param distance 行走距离
```

```
     * @param distanceUnit 距离单位
     */
    void onWalk(String direction, String distance, String distanceUnit);

    /**
     * 转向指令
     * @param direction 转向方向
     * @param angle 转向角度
     * @param angleUnit 角度单位
     */
    void onTurn(String direction, String angle, String angleUnit);

    /**
     * 举手指令
     * @param hands hand字段可为以下内容：hand, right_hand, left_hand, double_hand
     */
    void onRaiseHands(String hands);

    /**
     * 巡航指令
     * @param switchStatus 状态开关
     */
    void onCruise(String switchStatus);

    /**
     * 充电指令
     * @param switchStatus 状态开关
     */
    void onCharge(String switchStatus);

    /**
     * 握手指令
     */
    void onShakeHands();

    /**
     * 拥抱指令
     */
    void onHug();

    /**
     * 摇头指令
     */
    void onTwistHead();

    /**
     * 向左看指令
     */
    void onTurnHeadLeft();

    /**
     * 向右看指令
     */
    void onTurnHeadRight();

    /**
     * 停止运动指令
     */
    void onStop();

    /**
     * 打招呼指令
```

```
     */
    void onWave();

    /**
     * 动一下指令
     */
    void onDoAnAction();
}
```

ISpeakerControllerListener

ISpeakerControllerListener - 扬声器控制指令

```
public interface ISpeakerControllerListener extends IDirectiveListener {
    /**
     * 静音指令
     */
    void onSetMute();

    /**
     * 音量调节指令
     * @param volumeControl
     * @param volumeValue
     */
    void onAdjustVolume(String volumeControl, String volumeValue);
}
```

IInstructionListener

IInstructionListener - 特殊指令

```
public interface IInstructionListener extends IDirectiveListener {

    /**
     * 点亮查询指令
     */
    void onInquireEnergy();

    /**
     * 展示功能指令
     */
    void onShowFeatures();
}
```

IErrorListener

IErrorListener - 错误监听

```
public interface IErrorListener extends IDirectiveListener {
    /**
     * 错误监听
     * @param code 错误码
     * @param errorDescription 错误描述
     */
    void onError(int code, String errorDescription);
}
```

IStateListener

IStateListener - 指令处理状态监听

```
public interface IStateListener extends IDirectiveListener {
  /**
   * 当NLU能力识别完成本次对话请求中的用户意图，
   * SDK 开始向开发者回传本次对话请求返回的意图指令（Directives）时回调
   */
  void onDirectiveStart();

  /**
   * 当 SDK 完成了本次对话请求返回的全部意图指令的回传时回调
   */
  void onDirectiveStop();
}
```

ICustomListener

ICustomListener - 用户自定义能力监听

```
public interface ICustomListener extends IDirectiveListener {

  /**
   * 用户自定义意图回调监听
   *
   * @param namespace 指令 namespace
   * @param name 指令 name
   * @param payloads 指令参数 map
   * @return
   */
  boolean onDirectiveReceived(String namespace, String name, HashMap payloads);
}
```

公共UI

本SDK提供了下述几种自定义View，给开发中使用。

天气

1. 构造函数
   `WeatherLayoutView(Context context)`
2. 初始化函数
   `public void setWeatherLayoutView(WeatherInfoCard weatherInfoCard, int weight, int height)；` 参数：
   `WeatherInfoCard weatherInfoCard`：天气信息的实体类;
   `int weight`：为天气布局展示区域的宽；
   `int height`：为展示区域的高；
3. 实体类说明：

WeatherInfoCard:

参数名称	类型	详细说明
city	String	城市名，例如：“北京市”
time	String	日期，例如："周五 12月07日"
weatherInfo	List	天气信息(WeatherInfo)数组

WeatherInfo:

参数名称	类型	详细说明
type	String	有Common、Detail两种；当天日期为 Detail 类型，其他日期为 Common 类型
icon	String	天气图标url
temp	String	温度区间描述，例如"-10℃~-5℃"
time	String	日期，例如："周五 12月07日"
weather	String	天气描述，例如“晴”
wind	String	风向风力描述，例如“西北风微风”
pm25	String	pm25参数，返回一个字符串格式的整数，例如“10”、“42”
pmLevel	String	空气质量等级，例如“优”、“良好”
currentTemp	String	当前温度，例如“-7℃""

卡片

1. 构造函数
   `CardLayoutView(Context context)`
2. 初始化函数
   `public void setCardLayoutViewBean(final NormalCardItem cardItem)`
   参数：
   `NormalCardItem cardItem`：卡片信息的实体类;
3. 实体类说明：

参数名称	类型	详细说明
title	String	卡片标题
imageUrl	String	卡片图片url
content	String	卡片内容
url	String	卡片外链url

多图

1. 构造函数
   `MultigraphView(@NonNull Context context)`
2. 初始化函数
   `public void setMultigraph(ListCard listCard)`
   参数：
   `ListCard listCard`：为多图卡片实体类;
3. 实体类说明：
   ListCard

参数名称	类型	详细说明
list	List	卡片内容列表

SimpleImageCardItem

参数名称	类型	详细说明
image	String	图片链接
url	String	卡片外链url

长文本

1. 构造函数
   `TextLayoutview(Context context)`
2. 初始化函数
   `setCardContent(TextCard textCard)`
   参数：
   `TextCard textCard**`：为解析长文本json数据之后的实体类;
3. 实体类说明:
   SimpleImageCardItem

参数名称	类型	详细说明
content	String	文本内容
url	String	卡片外链url

高级配置项

SDK初始化

RobotSDKEngine getInstance() ： 获取SDK对象实例，需要先调用`RobotSDKEngine initSDK(SDKConfig config)`方法进行SDK的初始化。
RobotSDKEngine initSDK(SDKConfig config)：初始化RobotSDKEngine。返回初始化的SDK实例。
参数：`SDKConfig config`：SDK的一些基本配置信息，详细介绍见下文。

SDKConfig

`SDKConfig()`参数较多，我们提供并推荐以链式调用的方式进行参数配置，示例：

```
SDKConfig config = new SDKConfig.Builder()
            .context(mContext)
            .sdkType(SDKConfig.SDK_FACE_CONVERSATION)
            .clientid(CLIENT_ID)
            .clientSecret(CLIENT_SECRET)
            .build();
```

`SDKConfig`相关配置方法介绍见如下表格：

基础设置

方法名称	参数类型	必须	详细说明
context(Context context)	Context	是	SDK需要依赖应用程序Context，这里传入App的Context
clientid(String clientid)	String	是	机器人应用的ID 获取方法见集成准备 -> 获取ClientId、ClientSecret
clientSecret(String secret)	String	是	机器人应用的Secret 获取方法见集成准备 -> 获取ClientId、ClientSecret
sdkType(int type)	int	是	设置SDK模式，分为三种： SDKConfig.SDK_FACE：只有人脸； SDKConfig.SDK_CONVERSATION：只有对话 SDKConfig.SDK_FACE_CONVERSATION：对话与人脸

语音相关设置

方法名称	必须	详细说明
speechServiceType(int speechtype)	否（默认是使用USB阵列）	麦克风类型选择： SDKConfig.SPEECH_TYPE_U2S：USB麦克风阵列； SDKConfig.SPEECH_TYPE_INTERNAL：内置麦克风； SDKConfig.SPEECH_TYPE_CUSTOMER：自定义麦克风类型，详见 SPEECH_TYPE_NUWA：NuWa方案
wifiSSID(String ssid)	是（当使用USB麦克风阵列时）	给USB麦克风阵列设置账户名
wifiPWD(String pwd)	是（当使用USB麦克风阵列时）	给USB麦克风阵列设置密码
wifiType(String type)	否	给麦克风阵列设置Wi-Fi加密类型，可用参数如下： SDKConfig.SECURITY_NONE = "0"; SDKConfig.SECURITY_WEP = "1"; SDKConfig.SECURITY_WPA = "2"; SDKConfig.SECURITY_WPA2 = "3";
voiceType(String voicetype)	否	仅对内置麦克风有效，若使用使用麦克风阵列则无需填写。 SDKConfig.TTS_TYPE_FEMALE–普通女声， SDKConfig.TTS_TYPE_MALE–普通男声， SDKConfig.TTS_TYPE_GEZI–鸽子， SDKConfig.TTS_TYPE_MIDUO–情感男声米朵。

版本更新记录

V 1.0.2

更新时间：2019-05-27

1. 对初始化相关流程进行了说明细化，新增 Sample Application 的简介和链接跳转

2. 优化了SDKConfig的方法和接口介绍

V 1.0.1

更新时间：2019-05-22

优化了对于人脸识别相关接口的说明

V 1.0

更新时间：2019-04-26

Beta版

更新时间：2018-12-07

人脸库管理API参考

概述

本文档提供了调用人脸库管理的相关接口。通过token获取接口获取token，再调用人脸库管理接口和人脸管理接口时，需要携带token信息来验证调用者是否有权限调用相关接口。获取token时需要提供Client ID和Client Secret，每一组Client ID和Client Secret对应一个项目。要完成人脸的管理，首先要构建人脸库，然后向人脸库中添加人脸图片。项目、人脸库、人脸之间的关系是每个项目中包含多个人脸库，每个人脸库中包含多个人脸，同一个人脸可以在多个人脸库中。

接口介绍

OpenAPI Token获取接口

接口	描述
OpenAPI Token获取接口	调用人脸库管理接口和人脸管理接口时需要通过token认证，通过调用本接口可以获取token

人脸库管理接口

接口	描述
人脸库创建接口	创建一个人脸库
人脸库更新接口	更新人脸库信息，目前只支持更新人脸库名称
人脸库详情查询接口	查询人脸库详细信息
人脸库删除接口	删除一个人脸库及其包含的人脸信息
人脸库查询接口	查询人脸库列表

人脸管理接口

接口	描述
人脸创建接口	在指定的人脸库下创建人脸信息
人脸更新接口	修改人脸信息
人脸详情查询接口	查询人脸详细信息
人脸删除接口	删除一个指定的人脸，您可以选择在一个指定的人脸库中删除该人脸，或者在所有的人脸库删除该人脸。
人脸复制接口	将人脸信息在人脸库间复制，以允许在另一个人脸库下也能识别出当前人脸
人脸查询接口	查询指定人脸库下的人脸列表

通用说明

认证机制

访问人脸库管理接口和人脸管理接口时需要携带系统下发的token，token获取接口会返回两个字段，其中expires_in表示了token的有效期，如果token失效，调用接口不会成功，会有接口调用鉴权不通过的提示。

通信协议

支持HTTP和HTTPS两种调用方式。为了提升数据的安全性，建议通过HTTPS调用。

token获取接口说明

OpenAPI Token获取接口

接口描述

本接口用于获取OpenAPI access token。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/caas/oauth/token/v2`

Header如下：

头域	类型	说明	是否必须
Authorization	String	Basic认证编码后的Client ID和Client Secret	必须

> 注：Client ID和Client Secret从ABC Robot管理控制台的概览页获取。

请求参数

无

返回头域

除公共头域外，无其他特殊头域。

返回参数

名称	类型	描述
error_code	int	错误码
error_msg	long	错误信息
token	String	token 字符串
expires_in	long	token 有效时间

请求示例

```
 POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: Basic认证编码后的Client ID和Client Secret
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功",
    "token":"dihofspfhjwpfejwhefwaosdhifo1122djfn",
    "expires_in": 100000
}
```

人脸库管理接口

人脸库创建接口

接口描述

在添加人脸之前首先要创建人脸库，该接口提供创建人脸库的功能。

权限说明

请求发起人需要填写从token接口获取到的有效的token才能发起请求。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/openapi/v1/face-manage/group/create`

Header如下：

头域	类型	说明	是否必须
Authorization	String	填写根据token接口获取到的token	必须

请求参数

Body中放置请求参数，参数详情如下：

名称	类型	描述	是否必须
groupId	String	人脸库ID，项目级别唯一，提交后不可修改。数字、字母、下划线组成，长度不超过32个字符	必须
groupName	String	长度不超过40个字符，可修改，项目级别唯一。	必须

返回头域

除公共头域外，无其他特殊头域。

返回参数

参数名称	类型	描述
error_code	int	错误码
error_msg	String	操作结果提示信息
result	Object	操作结果展示Model

result的数据类型如下：

参数名称	类型	描述
groupId	String	人脸库ID
groupName	String	人脸库Name
createTime	long	创建时间
updateTime	long	更新时间

请求示例

```
 POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: authorization string
{
    "groupId":"face_group_id_1",
    "groupName":"face_group_name_1"
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功",
    "result": {
        "groupId":"face_group_id_1",
        "groupName":"face_group_name_1",
        "createTime":1547119839635,
        "updateTime":1547119839635
    }
}
```

人脸库修改接口

接口描述

修改指定人脸库ID的人脸库信息，目前只能修改人脸库名称。

权限说明

请求发起人需要具有合法的根据token接口获取到的token才能发起请求。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/openapi/v1/face-manage/group/update`

Header如下：

头域	类型	说明	是否必须
Authorization	String	填写根据token接口获取到的token	必须

请求参数

Body中放置请求参数，参数详情如下：

名称	类型	描述	是否必须
groupId	String	人脸库ID	必须
groupName	String	长度不超过40个字符，可修改，项目级别唯一。	必须

返回头域

除公共头域外，无其他特殊头域。

返回参数

参数名称	类型	描述
error_code	int	错误码
error_msg	String	操作结果提示信息

请求示例

```
 POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: authorization string
{
    "groupId":"face_group_id_1",
    "groupName":"face_group_name_1_update"
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功"
}
```

人脸库详情接口

接口描述

查询指定的人脸库详情信息。

权限说明

请求发起人需要具有合法的根据token接口获取到的token才能发起请求。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/openapi/v1/face-manage/group/detail`

Header如下：

头域	类型	说明	是否必须
Authorization	String	填写根据token接口获取到的token	必须

请求参数

Body中放置请求参数，参数详情如下：

名称	类型	描述	是否必须
groupId	String	人脸库ID	必须

返回头域

除公共头域外，无其他特殊头域。

返回参数

参数名称	类型	描述
error_code	int	错误码
error_msg	String	错误提示信息
result	Object	操作结果展示Model

result的数据类型如下：

参数名称	类型	描述
groupId	String	人脸库ID
groupName	String	人脸库Name
createTime	long	创建时间
updateTime	long	更新时间

请求示例

```
 POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: authorization string
{
    "groupId":"face_group_id_1"
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功",
    "result": {
        "groupId":"face_group_id_1",
        "groupName":"face_group_name_1_update",
        "createTime":1547119839635,
        "updateTime":1547119839635
    }
}
```

人脸库删除接口

接口描述

删除一个指定的人脸库和该人脸库下所有的人脸，如果一个人脸在多个人脸库中存在，该操作不会删除其他人脸库中的人脸。

权限说明

请求发起人需要具有合法的根据token接口获取到的token才能发起请求。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/openapi/v1/face-manage/group/delete`

Header如下：

头域	类型	说明	是否必须
Authorization	String	填写根据token接口获取到的token	必须

请求参数

Body中放置请求参数，参数详情如下：

名称	类型	描述	是否必须
groupId	String	人脸库ID	必须

返回头域

除公共头域外，无其他特殊头域。

返回参数

参数名称	类型	描述
error_code	int	错误码
error_msg	String	错误提示信息

请求示例

```
 POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: authorization string
{
    "groupId":"face_group_id_1"
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功"
}
```

人脸库查询接口

接口描述

查询人脸库列表。

权限说明

请求发起人需要具有合法的根据token接口获取到的token才能发起请求。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/openapi/v1/face-manage/group/query`

Header如下：

头域	类型	说明	是否必须
Authorization	String	填写根据token接口获取到的token	必须

请求参数 Body中放置请求参数，参数详情如下：

名称	类型	描述	是否必须
pageNo	int	查询页数，大小不能小于1	必须
pageSize	int	每页显示数量，范围为1-100	必须

返回头域 除公共头域外，无其他特殊头域。

返回参数

参数名称	类型	描述
error_code	int	错误码
error_msg	String	提示信息
result	Object	操作结果展示Model

result的数据类型如下：

参数名称	类型	描述
pageSize	int	每页显示数量
curPage	int	当前页
totalPage	long	总页数
totalNumber	long	总条数
content	Array	列表内容

content的数据类型如下：

参数名称	类型	描述
groupId	String	人脸库ID
groupName	String	人脸库名称
createTime	long	创建时间
updateTime	long	更新时间
faceCount	long	人脸库包含的人脸数量

请求示例

```
 POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: authorization string
{
    "pageNo":1
    "pageSize":15
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功",
    "result": {
        "pageSize":15,
        "curPage":1,
        "totalPage":1,
        "totalNumber":1,
        "content":[
          {
              "groupId":"Default",
              "groupName":"Default",
              "createTime":1558340478928,
              "updateTime":1558340478928,
              "faceCount":1
          }
        ]
    }
}
```

人脸管理接口

人脸创建接口

接口描述

在指定的人脸库下创建人脸，如果指定的人脸库ID有多个，则会在指定的人脸库下都创建人脸。

权限说明

请求发起人需要具有合法的根据token接口获取到的token才能发起请求。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/openapi/v1/face-manage/face/create`

Header如下：

头域	类型	说明	是否必须
Authorization	String	填写根据token接口获取到的token	必须

请求参数 Body中放置请求参数，参数详情如下：

名称	类型	描述	是否必须
groupIds	List	人脸库ID，需提前创建，指定当前人脸要绑定的人脸库	必须
uid	String	不可修改，支持英文、数字、下划线，长度不超过32个字符。用户自定义。	必须
name	String	支持中文、英文、空格，长度不超过20个字符，并且必须要有一个非空字符	必须
title	String	称谓，可以为空，长度不超过20个字符	非必须
image	String	图片base64数据，需要注意的是，图片的base64编码是不包含图片头的，如data:image/jpg;base64。若一张图片中包含多张人脸，只选取其中人脸面积最大的人脸。支持PNG、JPG、JPEG、BMP格式。图片大小不超过2M。	必须
gender	String	性别，如果填写，则必须是male或female	非必须
remark	String	备注信息，长度不超过100字符	非必须

返回头域 除公共头域外，无其他特殊头域。

返回参数

参数名称	类型	描述
error_code	int	错误码
error_msg	String	错误提示信息
result	Object	操作结果展示Model

result的数据类型如下：

参数名称	类型	描述
uid	String	人脸id
name	String	姓名
title	String	称谓
groups	List	当前人脸关联的人脸库ID
image	String	图片base64数据
gender	String	性别
remark	String	备注
createTime	long	创建时间
updateTime	long	更新时间

请求示例

```
POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: authorization string
{
    "uid":"face_uid",
    "name":"user name",
    "title":"title from customer",
    "groupIds":[
        "customerGroupId1",
        "customerGroupId2"
    ],
    "image":"base64 encoded image",
    "gender":"female",
    "remark":"remark"
}
```

响应示例

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功",
    "result": {
        "uid":"face_uid",
        "name":"user name",
        "title":"title from customer",
        "groups":["customerGroupId1","customerGroupId2"],
        "image":"base64 encoded image",
        "gender":"gender from customer",
        "remark":"remark",
        "createTime":1547119839635,
        "updateTime":1547119839635
        }
}
```

人脸修改接口

接口描述

修改指定的人脸信息。

权限说明

请求发起人需要具有合法的根据token接口获取到的token才能发起请求。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/openapi/v1/face-manage/face/update`

Header如下：

头域	类型	说明	是否必须
Authorization	String	填写根据token接口获取到的token	必须

请求参数 Body中放置请求参数，参数详情如下：

名称	类型	描述	是否必须
uid	String	人脸ID	必须
name	String	支持中文、英文、空格，长度不超过20个字符，并且必须要有一个非空字符	非必须
title	String	称谓，可以为空，长度不超过20个字符	非必须
image	String	图片base64数据，需要注意的是，图片的base64编码是不包含图片头的，如data:image/jpg;base64。若一张图片中包含多张人脸，只选取其中人脸面积最大的人脸。支持PNG、JPG、JPEG、BMP格式。图片大小不超过2M。	非必须
gender	String	性别，如果填写，则必须是male或female	非必须
remark	String	备注	非必须

返回头域 除公共头域外，无其他特殊头域。

返回参数

参数名称	类型	描述
error_code	int	错误码
error_msg	String	错误提示信息

请求示例

```
 POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: authorization string
{
    "uid":"face_uid",
    "name":"user name",
    "title":"title from customer",
    "image":"base64 encoded image",
    "gender":"female",
    "remark":"remark"
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功"
}
```

人脸详情接口

接口描述

获取指定的人脸详情信息

权限说明

请求发起人需要具有合法的根据token接口获取到的token才能发起请求。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/openapi/v1/face-manage/face/detail`
Header如下：

头域	类型	说明	是否必须
Authorization	String	填写根据token接口获取到的token	必须

请求参数 Body中放置请求参数，参数详情如下：

名称	类型	描述	是否必须
uid	String	人脸ID	必须

返回头域 除公共头域外，无其他特殊头域。

返回参数

参数名称	类型	描述
error_code	int	错误码
error_msg	String	错误提示信息
result	Object	操作结果展示Model

result的数据类型如下：

参数名称	类型	描述
uid	String	人脸id
name	String	姓名
title	String	称谓
groups	List	当前人脸关联的人脸库ID
image	String	图片base64数据
gender	String	性别
remark	String	备注
createTime	long	创建时间
updateTime	long	更新时间

请求示例

```
 POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: authorization string
{
    "uid":"face_uid"
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功",
    "result": {
        "uid":"group1_uid",
        "name":"user name",
        "title":"title from customer",
        "groups":["customerGroupId1","customerGroupId2"],
        "image":"base64 encoded image",
        "gender":gender from customer,
        "remark":"remark",
        "createTime":1547119839635,
        "updateTime":1547119839635
        }
}
```

人脸删除接口

接口描述

删除一个指定的人脸，您可以选择在一个指定的人脸库中删除该人脸，或者在所有的人脸库删除该人脸。

权限说明

请求发起人需要具有合法的根据token接口获取到的token才能发起请求。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/openapi/v1/face-manage/face/delete`

Header如下：

头域	类型	说明	是否必须
Authorization	String	填写根据token接口获取到的token	必须

请求参数 Body中放置请求参数，参数详情如下：

名称	类型	描述	是否必须
uid	String	人脸ID	必须
groupIds	List	如果groupIds不为空，则在指定的人脸库中删除该人脸；如果为空，则在所有的人脸库删除该人脸。	非必须

返回头域 除公共头域外，无其他特殊头域。

返回参数

参数名称	类型	描述
error_code	int	错误码
error_msg	String	错误提示信息

请求示例

```
 POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: authorization string
{
    "uid":"face_uid",
    "groupIds":["customerGroupId1","customerGroupId2"]
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功"
}
```

人脸库间复制接口

接口描述

将用户人脸信息在人脸库间复制，以允许在另一个人脸库下也能识别出当前用户。

权限说明

请求发起人需要具有合法的根据token接口获取到的token才能发起请求。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/openapi/v1/face-manage/face/copy`

Header如下：

头域	类型	说明	是否必须
Authorization	String	填写根据token接口获取到的token	必须

请求参数 Body中放置请求参数，参数详情如下：

名称	类型	描述	是否必须
uid	String	人脸ID	必须
groupIds	List	需要复制人脸用户的人脸库ID	必须

返回头域 除公共头域外，无其他特殊头域。

返回参数

参数名称	类型	描述
error_code	int	错误码
error_msg	String	错误提示信息

请求示例

```
 POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: authorization string
{
    "uid":"face_uid",
    "groupIds":["customerGroupId1","customerGroupId2"]
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功"
}
```

人脸列表查询接口

接口描述

查询指定人脸库下的人脸列表。

权限说明

请求发起人需要具有合法的根据token接口获取到的token才能发起请求。

请求结构

HTTP方法：POST

请求URL： `https://robot.baidu.com/abc-robot/openapi/v1/face-manage/face/query`

Header如下：

头域	类型	说明	是否必须
Authorization	String	填写根据token接口获取到的token	必须

请求参数 Body中放置请求参数，参数详情如下：

名称	类型	描述	是否必须
groupId	String	人脸库ID	必须
pageNo	int	查询页数，大小不能小于1	必须
pageSize	int	每页显示数量，范围为1-100	必须

返回头域 除公共头域外，无其他特殊头域。

返回参数

参数名称	类型	描述
error_code	int	错误码
error_msg	String	提示信息
result	Object	操作结果展示Model

result的数据类型如下：

参数名称	类型	描述
pageSize	int	每页显示数量
curPage	int	当前页
totalPage	long	总页数
totalNumber	long	总条数
content	Array	列表内容

content的数据类型如下：

参数名称	类型	描述
uid	String	人脸ID
name	String	姓名
gender	String	性别
title	String	称谓
registerDate	String	更新时间
remark	String	备注

请求示例

```
 POST HTTP/1.1
Host: robot.baidu.com
Content-Type: application/json;charset=UTF-8
Authorization: authorization string
{
    "groupId":"Default"
    "pageNo":1
    "pageSize":15
}
```

响应示例

```
 HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
{
    "error_code":0,
    "error_msg":"操作成功",
    "result": {
        "pageSize":15,
        "curPage":1,
        "totalPage":1,
        "totalNumber":1,
        "content":[
          {
              "uid":"001",
              "name":"zhangsan",
              "gender":"male",
              "title":"title of user",
              "registerDate":"2019-05-30 18:59:06",
              "remark":"register remark"
          }
        ]
    }
}
```

错误码

错误码格式

当用户访问API出现错误时，会返回给用户相应的错误码和错误信息，便于定位问题，并做出适当的处理。请求发生错误时通过Response Body返回详细错误信息，遵循如下格式：

参数名	类型	说明
error_code	int	表示具体错误类型
error_msg	String	有关该错误的详细说明

例如：

```
 {
   "error_code": 0，
   "error_msg":"操作成功"
}
```

错误码	描述
0	请求成功时的提示信息
51200	token获取通用错误
51201	项目被禁用
51204	token认证失败
40001	请求参数错误。
50000	服务器内部错误
51300	API模块错误信息
50300	人脸模块内部通用错误