ICS 33.050
CCS M 30

团　体　标　准

T/TAF 283—2025

智能终端意图框架接口技术要求

Technical requirements for interface of intelligent terminal intent framework

2025-06-11 发布　　2025-06-11 实施

电信终端产业协会　发布

版权声明

邮箱：tafrb@taf.org.cn

电话：010-82052809

目　　次

前　言

本文件按照 GB/T 1.1—2020《标准化工作导则 第 1 部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由电信终端产业协会（TAF）提出并归口。

本文件起草单位：中国信息通信研究院、OPPO广东移动通信有限公司、维沃移动通信有限公司、荣耀终端股份有限公司、小米通讯技术有限公司、华为终端有限公司、蚂蚁科技集团股份有限公司、紫光展锐（上海）科技有限公司。

本文件主要起草人：解谦、樊列龙、戈志勇、曾晨曦、黄宇畅、曹宇琼、唐溯、高立发、潘斌斌、曾祥霆、曾勇波、郑钊、鲁岩、麦睿楷、史浩、林冠辰、林国涛、李琦、文军、张伟强、李丛蓉。

引　　言

借助人工智能技术发展，智能助手将能够更好的理解用户意图并自动完成复杂工作，带来与触摸屏的不同使用体验，这需要终端设备与移动互联网应用、用户间建立更合适的交互方式。

意图框架提供系统级的意图体系和标准的意图接入方式，通过多维系统感知、大模型等能力构建全局意图范式，为终端与应用及服务间提供标准化的接口和协议，使得不同设备和应用软件能够遵循统一的规范进行通信和交互。

主流终端厂商均开展意图框架产品的研发，为了保证意图框架同应用接口的统一，减少终端企业和应用企业开发适配成本，避免终端智能体生态产生碎片化，需要制定智能终端意图框架系列标准。

本文件“智能终端意图框架”系列标准之一，该系列标准的结构及名称预计如下：

——智能终端意图框架总体技术要求；

——智能终端意图框架接口技术要求。

智能终端意图框架接口技术要求

1 范围

本文件规定了智能终端意图框架接口技术要求，包括端侧技术方案和云侧技术方案两部分内容。其中端侧技术方案包括通信流程、意图注册、意图共享、意图调用、其它接口、接口安全、错误码、静态资源配置等接口相关要求。云侧技术方案包括意图注册与服务创建、接口鉴权、意图共享、意图调用等接口相关要求。

本文件适用于搭载安卓操作系统的智能终端意图框架开发、适配和测试。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

T/TAF 282—2025 智能终端意图框架总体技术要求

3 术语和定义

T/TAF 282—2025界定的术语和定义适用于本文件。

4 缩略语

下列缩略语适用于本文件。

UTF-8：8位元Unicode 转换格式（Unicode Transformation Format, 8-bit）

5 意图接口安卓端侧技术方案

5.1 安卓接口通信总览

在安卓技术方案中，应用和终端系统共同实现意图框架接口。意图框架接口在通信层使用 Content Provider 实现，应用层需要实现意图注册、共享和调用接口，接口通信总览如图 1 所示。

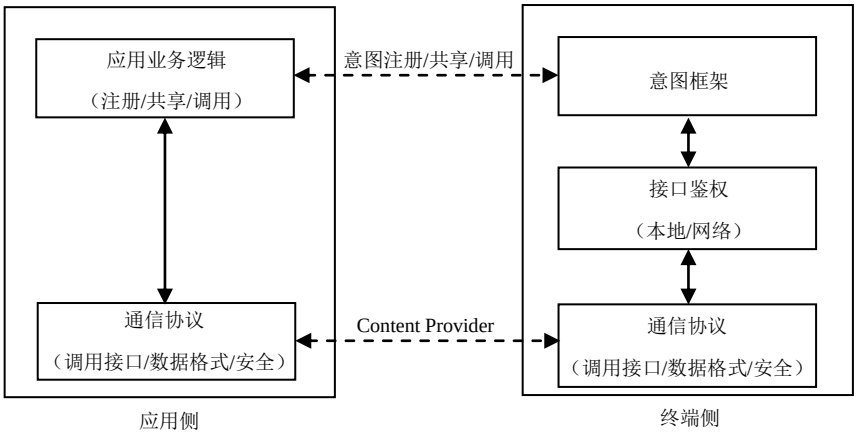

图 1　安卓接口通信总览

5.2　通信流程

5.2.1　通信流程概述

应用和终端系统之间通信支持同步和异步两种调用模式，同步的方式直接使用 Content Provider 完成通信，异步的方式使用 ResultReceiver 传递响应结果。

5.2.2　同步调用接口

安卓同步接口调用流程见图 2。

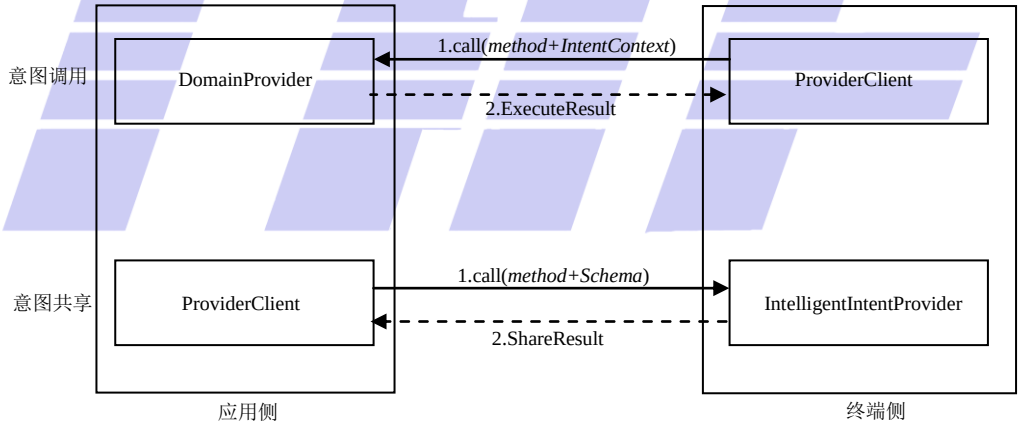

图 2　安卓同步接口调用流程

其中：

——**意图共享**：应用调用 Content Provider 的 call()方法，传递方法和要共享的数据，系统收到应用共享数据后将系统响应结果立即返回给应用；

——**意图调用**：应用实现业务 Content Provider，并在意图注册阶段声明意图调用的 content uri 路径，系统将在意图调用阶段调用应用声明的 Content Provider 的 call()方法，应用方在 call()方法中按照垂域要求返回对应数据。

5.2.3　异步接口调用

异步数据返回配合 ResultReceiver 使用，ProviderClient 调用后仅返回 CallResult 结果，完整

的结果需要等待 Provider 端调用 ResultReceiver.send()方法，Client 端在 onReceiveResult()方法中处理真实的意图数据，见图 3。

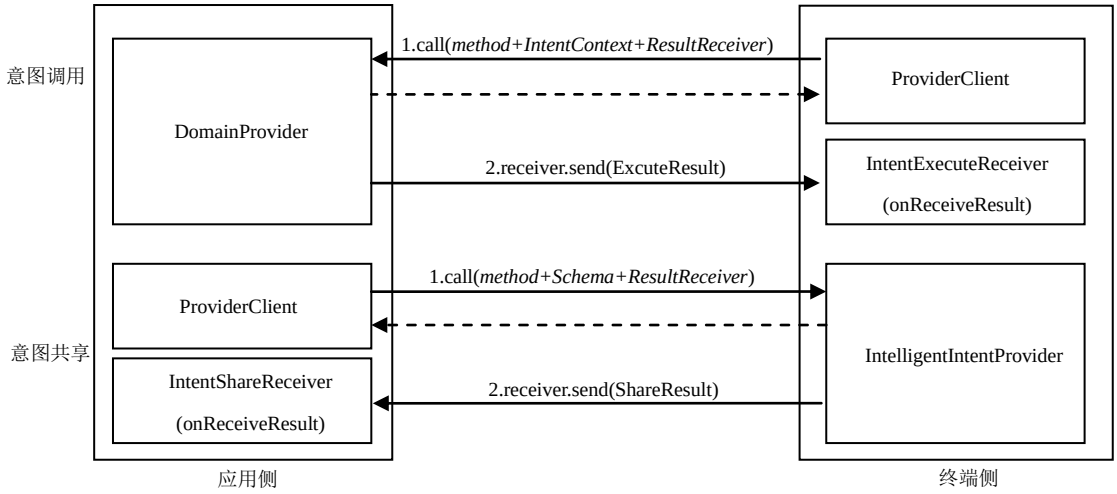

图 3　安卓异步接口调用流程

5.3　意图注册

5.3.1　意图命名规则

意图命名规则见 T/TAF 282—2025 中 5.2 要求。

5.3.2　AndroidManifest.xml 配置

```
<application>
  <!-- 注册文件放在 App 的 assets 目录中 -->
  <meta-data
    android:name="IntelligentIntentConfig"
    android:value="assets/intelligent_intent_config.json" />
</application>
```

5.3.3　意图注册文件

intelligent_intent_config.json：

```
{
  // 应用支持的意图列表
  "intelligentIntents":  [
    // 意图共享注册示例
    {
      // 意图名称
      // 名称应当遵循意图框架规范，当前仅支持预置垂域意图，不允许自定义
      // 应用内意图名称唯一，不允许出现相同的名称定义
      "intentName": "Navigation.ShareNavigationInfo",
```

```
      // 意图版本号
      "intentVersion": "1.0.0",

      // 用于系统在入口进行推荐
      "description": ["应用共享导航信息给意图框架"]
    },

    // 意图调用注册示例
    {
      // 意图名称
      // 名称应当遵循意图框架规范，当前仅支持预置垂域意图，不允许自定义
      // 应用内意图名称唯一，不允许出现相同的名称定义
      "intentName": "Navigation.StartNavigationFg",

      // 意图版本号
      "intentVersion": "1.0.0",

      // 意图调用类型, foreground / background
      "executeMode": "foreground",

      // 意图调用的响应入口：foreground 则配置 deep link 链接(如应用落地页)
      // background 则配置响应服务如：content://com.xxx.xxx/entry，调用传递的参数每个垂域不同
      "executeEntry": "navi://minimap/xxx",

      // 意图调用同步还是异步， 默认同步。executeMode 为 background 时生效
      "executeSync": true,

      // 用于系统在入口进行推荐
      "description": ["意图框架调用应用发起导航"]
    },
    {...}
  ]
}
```

意图共享注册结构如表 1 所示。

表 1　意图共享注册结构

参数名	类型	默认值	必填	描述
intentName	String	-	是	意图名称，名称应当遵循意图框架规范，当前仅支持预置垂域意图，不允许自定义

表 1　意图共享注册结构（续）

参数名	类型	默认值	必填	描述
intentVersion	String	-	是	意图体系定义的意图版本号，如“1.0.0”。开发者选择，不可随意填写。
description	Array	-	是	用于意图框架推荐描述

意图调用注册结构如表 2 所示。

表 2　意图调用注册结构

参数名	类型	默认值	必填	描述
intentName	String	-	是	意图名称，名称应当遵循意图框架规范，当前仅支持预置垂域意图，不允许自定义
intentVersion	String	-	是	意图体系定义的意图版本号，如“1.0.0”。开发者选择，不可随意填写。
executeMode	String	-	是	意图调用类型，可选值：foreground / background
executeEntry	JSON	-	是	意图调用的响应入口：1、foreground 则配置 deep link 链接(如快应用或原生应用的落地页) 2、background 则配置响应服务如：content://com.xxx.xxx/methodName
executeSync	Boolean	true	否	意图调用阶段是否同步调用
description	Array	-	是	用于意图框架推荐描述

5.4　意图共享

5.4.1　意图共享概述

意图共享接口支持一次共享多个意图和单个意图，支持同步和异步两类接口，支持删除共享的意图和实体数据。示意图如图 4 所示。

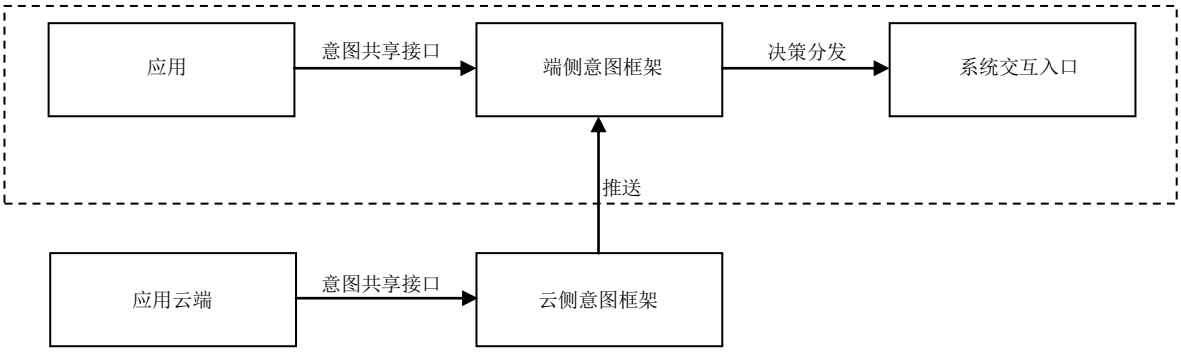

图 4　意图共享逻辑示意图

5.4.2　意图共享数据结构

IntelligentIntent：意图共享的基础数据单元，应用在调用意图共享接口的时候需要传递意图共享数据，见表 3，意图共享数据示例见附录 A.1。

表 3　意图共享基础数据单元

名称	类型	共性	必选	说明
intentName	String	是	是	意图名称
identifier	String	是	是	本次意图共享的唯一 id，可用 UUID 生成，开发者按照自身业务需要生成，作为单条共享记录的唯一标识符。
timestamp	Long	是	是	当前时间戳，单位为毫秒
serviceId	Array	否	是	服务在终端方开发者平台申请的服务 ID。当共建服务平台时终端方定义一致。
intentAction	JSON	是	是	用于描述意图的动作
├ actionType	String	是	是	数据或动作的类型：fact（事实）/ predict（预测）
├ actionTime	JSON	是	是	发生时间、结束时间{ "startTime": 1729485000989, "endTime": 1729485000989}
intentEntity	JSON	是	是	意图共享实体信息，用于描述意图的垂域实体内容
├ entityName	String	是	是	实体名称，如：Navigation
├ entityId	String	是	是	实体 ID，用于描述实体的唯一标识符，业务可传订单号之类的标识
├ isPublic	Boolean	是	是	是否与用户行为相关数据
├・・・（不同垂域特有字段）	・・・	・・・	・・・	不同垂域特有字段
extra	JSON	否	否	扩展字段

ShareResult：意图共享响应结果，见表 4。

表 4　意图共享响应结果

名称	类型	必选	说明
code	Int	是	调用状态码，0：成功，其他：失败
message	String	是	错误消息
data	JSON	否	响应结果其他附加内容
├ identifier	String	是	意图共享的时候，回传意图 id

5.4.3　系统侧 Provider 声明

权限声明：

```
<permission
    android:name="intelligent.permission.SHARE_INTENT"
    android:protectionLevel="normal"
    android:label="app can share intent to intelligent intent framework"
```

```
    android:description="Allow the app to call share intent and delete intent interface"
/>

<permission
    android:name="intelligent.permission.EXECUTE_INTENT"
    android:protectionLevel="normal"
    android:label="Applications can be called by the intelligent intent framework"
    android:description="Allows the application to be called by the intelligent intent fr
amework" />
```

Provider 声明：

```
<provider
    android:name="com.xxx.IntelligentIntentProvider"
    android:authorities="IntelligentIntent"
    android:permission="intelligent.permission.SHARE_INTENT"
    android:enabled="true"
    android:exported="true" />
```

意图框架 ContentProvider 入口 authorities 各系统厂家保持一致，默认值为 IntelligentIntent。该值可通过系统 settings 属性查询得到：

——adb shell 查询：

```
> settings get global intelligent_intent_authorities
> IntelligentIntent
```

——代码查询：

```
val authorities = Settings.Global.getString(context.getContentResolver(), "intelligent_in
tent_authorities") ?: "IntelligentIntent"
```

5.4.4 应用端声明

AndroidManifest.xml：

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
    <uses-permission android:name="intelligent.permission.SHARE_INTENT" />
    <uses-permission android:name="intelligent.permission.EXECUTE_INTENT" />
    <queries>
        <provider android:authorities="IntelligentIntent"/>
    </queries>
</manifest>
```

5.4.5 意图共享鉴权流程

意图共享鉴权流程如图 5 所示，包括：

a) 应用在注册阶段申请意图权限，审核通过后应用与其对应的权限存储在云侧意图框架；

b) 端侧意图框架从云侧意图框架预拉取意图授权信息，并缓存到本机；

c） 业务接口调用意图框架具体接口（如意图共享）；
d） 意图框架获取调用应用的包名+签名；
e） 根据包名+签名校验应用是否具备相应意图的处理权限；
f） 权限校验通过后执行具体的业务逻辑。

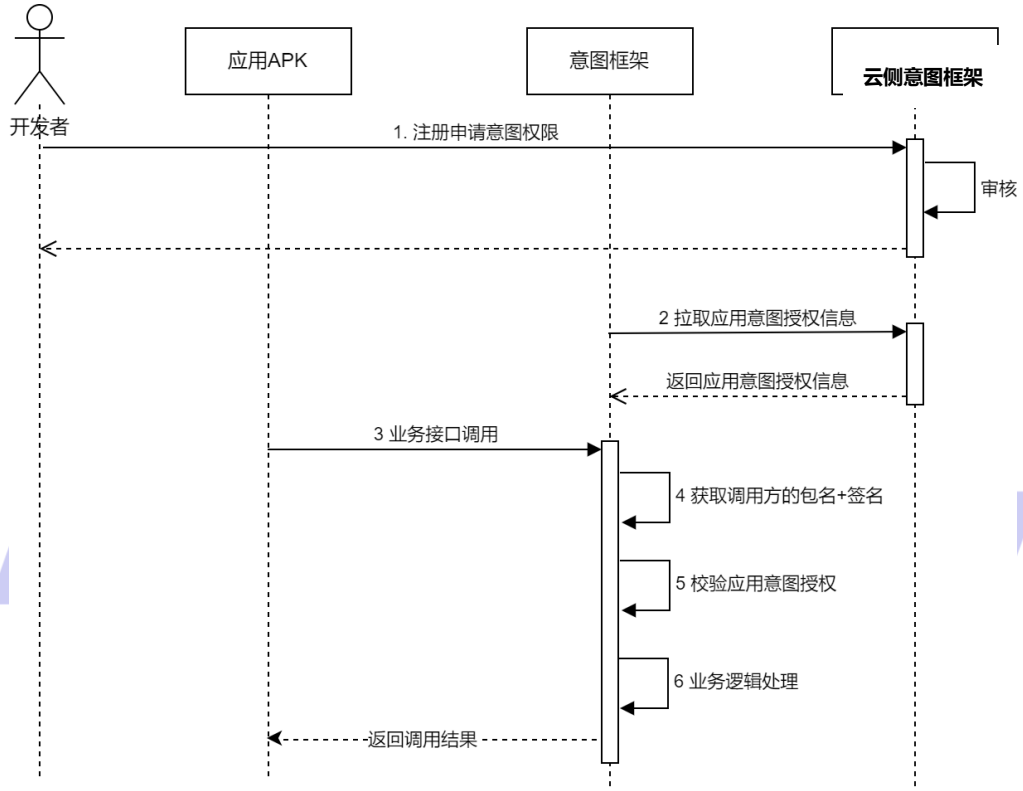

图 5　意图共享鉴权流程

5.4.6　共享单个意图

5.4.6.1　共享单个意图方法

通过 ContentProviderClient.call()直接调用系统的 IntelligentIntentProvider，call 方法声明如下：

```
fun call(authority: String, method: String, arg: String?, extras: Bundle?): Bundle
```

5.4.6.2　同步接口

同步接口如图 6 所示。

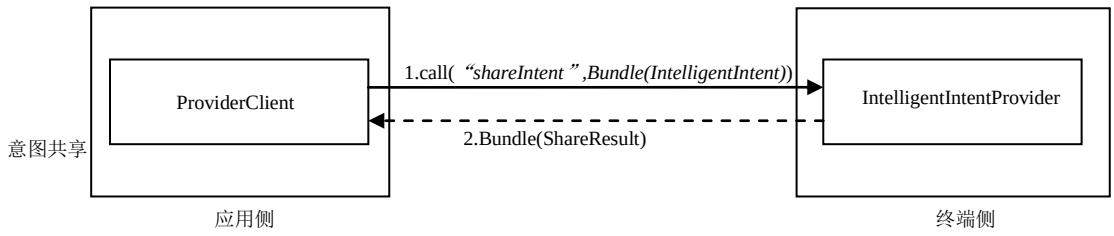

图 6　同步接口

对应 call 方法的参数见表 5。

表 5　同步接口 call 方法参数

参数	值
authorityIntelligentIntent	IntelligentIntent
method	shareIntent
arg	null
extras	该 Bundle 需要传递如下内容：- putString(“**intentData**”，“{…}”) // intentData 的内容为 JSON String,格式参考 IntelligentIntent 数据结构描述
code	10101001 - 应用无该意图共享权限 10101002 - 应用该意图共享权限已过期 10102001 - 意图共享参数错误 10103001 - 意图共享数据大小超过限制 10103002 - 意图共享频次超过限制 10103003 - 意图共享开关已关闭 10103004 - 意图共享实体数量超限 10103005 - 意图共享版本不支持

应用示例代码见附录 A.2。

5.4.6.3　异步接口

与同步调用方式一样，异步方式也是通过 ContentProviderClient.call() 调用 IntelligentIntentProvider 的 call()方法，不同的是，method 变更为 shareIntentAsync，传递 extra 参数除了包含 IntelligentIntent 之外还需要包含 ResultReceiver 对象，call()方法的响应值为 CallResult，而不是 ShareResult 结构。ShareResult 结构会在 IntentShareReceiver 的 onReceiveResult()方法中返回。异步接口示意见图 7。

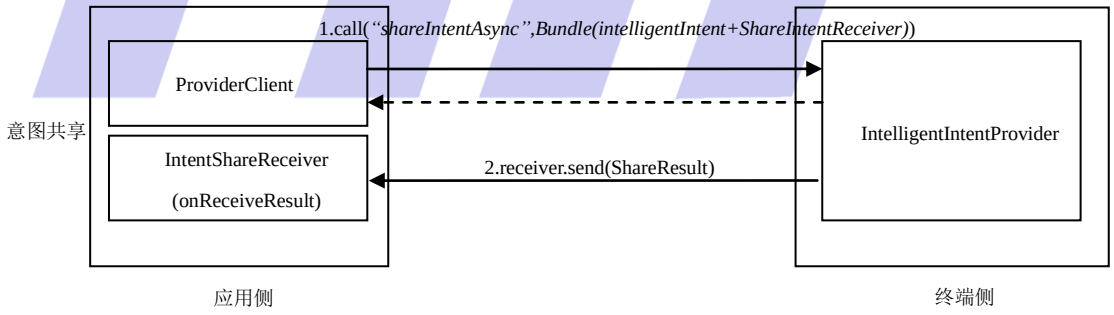

图 7　异步接口示意图

对应 call 方法的参数见表 6。

表 6　异步接口 call 方法参数

参数	值
authority	IntelligentIntent
method	shareIntentAsync
arg	null

表 6　异步接口 call 方法参数（续）

参数	值
extras	putString("intentData", "{...}") // intentData 的内容为 JSON String, 格式参考 IntelligentIntent 数据结构描述 putParcelable("shareIntentReceiver", shareIntentReceiver)// 传递意图共享结果回调响应 ResultReceiver 对象
code	10101001 - 应用无该意图共享权限； 10101002 - 应用该意图共享权限已过期； 10102001 - 意图共享参数错误； 10103001 - 意图共享数据大小超过限制； 10103002 - 意图共享频次超过限制； 10103003 - 意图共享开关已关闭； 10103004 - 意图共享实体数量超限； 10103005 - 意图共享版本不支持。

应用示例代码见附录 A.3。

5.4.7　删除意图

5.4.7.1 删除意图的接口

删除意图包含两个接口，deleteIntent 和 deleteEntity：

——deleteIntent：通过指定共享的意图 identifier 删除已共享的意图列表；

——deleteEntity：删除指定实体 id 的意图实体对象。

5.4.7.2　deleteIntent

通过 ContentProviderClient.call()直接调用系统的 IntelligentIntentProvider，call 方法参数见表 7。

表 7　deleteIntent 参数

参数	值
authority	IntelligentIntent
method	deleteIntent
arg	null
extras	该 Bundle 需要传递如下内容：putString(“intentName”, “navigation.StartNavigation”) // 意图名称 putStringArrayList(“identifiers”, arrayListOf(“id1”, “id2”)) // 意图 id 列表
code	10201001 - 应用无该意图删除权限； 10202001 - 删除意图参数错误； 10203001 - 删除意图失败。

应用调用示例代码见附录 A.4。

5.4.7.3　deleteEntity

通过 ContentProviderClient.call()直接调用系统的 IntelligentIntentProvider，call 方法参数见表 8。

表 8　deleteEntity 参数

参数	值
authority	IntelligentIntent
method	deleteEntity
arg	null
extras	该 Bundle 需要传递如下内容：putString(“intentName”，“Navigation.NavigationInfo”) // 意图名称 putStringArrayList(“entityIds”，arrayListOf(“abc123”)) // 实体 id 列表
code	10201001 - 应用无该意图删除权限； 10202001 - 删除意图参数错误； 10203001 - 删除意图失败。

应用调用示例代码见附录 A.5。

5.5　意图调用

5.5.1　意图调用接口

意图调用分为前台模式(foreground)、后台模式(background)，调用的模式在意图注册的时候已经声明。前台模式支持 deep link 的方式拉起应用；后台模式支持调用应用方提供的 ContentProvider.call()接口。在意图调用的过程中会传递意图参数（InetntParams）。意图调用示意图见图 8。

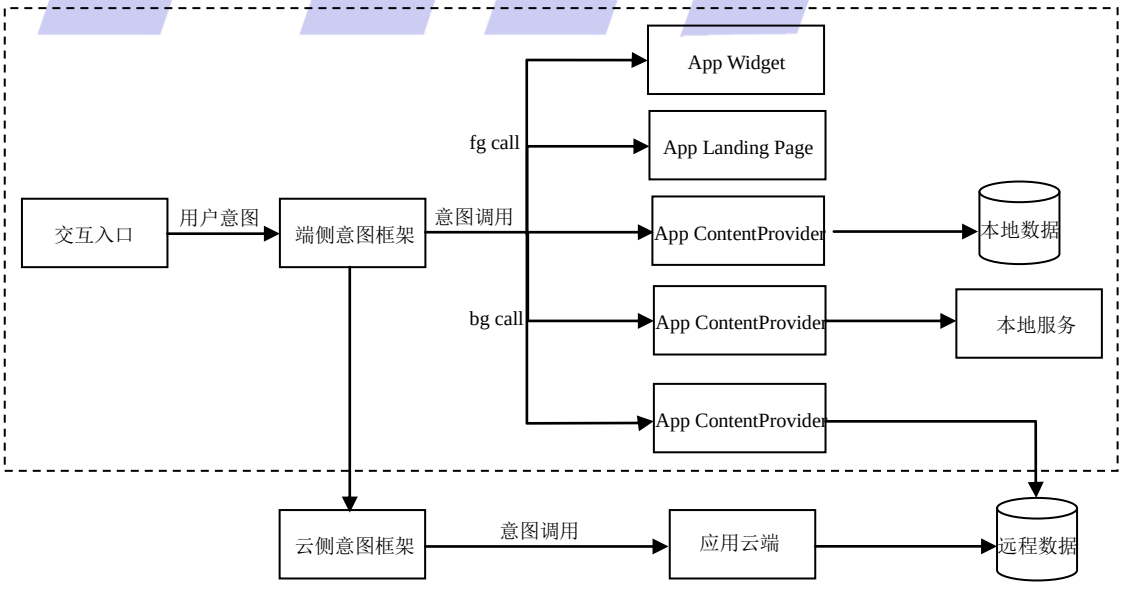

图 8　意图调用逻辑示意图

5.5.2 意图参数(IntentParams)数据结构

IntentParams 是意图调用阶段传递给应用的参数，用于携带意图调用过程中的上下文信息，如 intentName，parameters 等，见表 9。

表 9 意图参数数据结构

名称	类型	必选	说明
intentName	String	是	意图名称
requstId	String	是	本次意图调用的唯一 id，应用完成意图调用逻辑后通知系统返回需要回传该参数
executeSync	Boolean	是	本次意图调用是否是同步调用，默认是同步调用
device	String	是	触发意图调用的设备信息，可选值： phone/watch/tablet/tv/car
entry	String	是	触发意图调用的入口信息，如语音助手
entityId	String	否	当同一入口有多个实例的情况，用于区分是哪一个实例发起的调用。如有用户购买了多张车票，需要查询具体某一张车票信息
parameters	JSON	否	意图调用参数，应用解析，如地理位置坐标/用户习惯等，不同垂域应定义不同内容
extras	JSON	否	额外的意图参数

ExecuteResult：意图调用回执，由应用传递给系统，见表 10。

表 10 意图调用回执参数

名称	类型	必选	说明
code	Int	是	调用状态码，0：成功，其他：失败。 20301001 - 无该意图调用权限； 20301002 - 用户未授权该意图调用； 20302001 - 意图调用参数错误； 20303001 - 不支持该意图调用； 20303002 - 意图调用响应超时。
message	String	是	错误消息
requestId	String	是	意图参数中的 requestId
data	JSON	否	响应结果其他附加内容，不同垂域需不同定义

5.5.3 前台模式

对于在意图注册阶段声明的 deep link 链接，如 hap://com.xxx.xx/index， 在意图调用的时候会在 deep link 连接的后面拼接 intentParams 参数，该参数是 intentParams 的 JSON String 经过 Base64 编码后的字符串。因此在调用的阶段会变成：hap://com.xxx.xx/index?intentParams=base64string 。

应用拿到该部分 intentParams 参数内容后对其进行解码，然后转换成 JSON 对象完成自己的业务逻

辑。

5.5.4 后台模式

5.5.4.1 后台模式概述

后台模式支持 ContentProvider 方式将能力暴露给系统进行调用，系统会调用应用方 ContentProvider 的 call 方法，应用在 call 方法中将数据按照垂域约定字段回传给系统，回传过程支持同步和异步两种接口，同步调用直接构造返回的 bundle 结果即可。

5.5.4.2 意图调用 Provider 注册

应用在注册意图调用入口的时候，注册示例如下：

```
content://authority/methodName?key=value
```

call 方法的调用说明：

——authority：provider 的认证信息；

——methodName：意图调用方法名称，将会被当做 call 方法的 method 参数；

——key=value：将转换成 JSON 格式字符串，传递给 arg 参数；

——extras：将携带 IntentParams 参数信息；

——ExecteResult：意图调用响应结果应包含的数据格式。

后台意图调用参数规范见图 9。

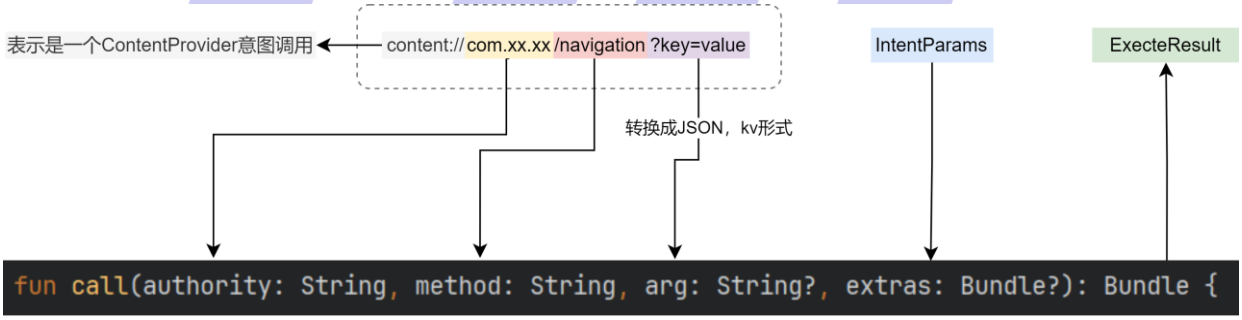

图 9　后台意图调用参数规范

5.5.4.3 同步意图调用

同步接口的调用应用在 ContentProvider 中收到 call()方法调用后，直接构造 ExecuteResult 格式的 Bundle 返回即可。决定是同步调用还是异步调用，取决于在意图注册阶段配置的 executeSync 字段值，如果该值为为 true 则为同步调用，如果该值为 false 则为异步调用。同步意图调用流程见图 10。

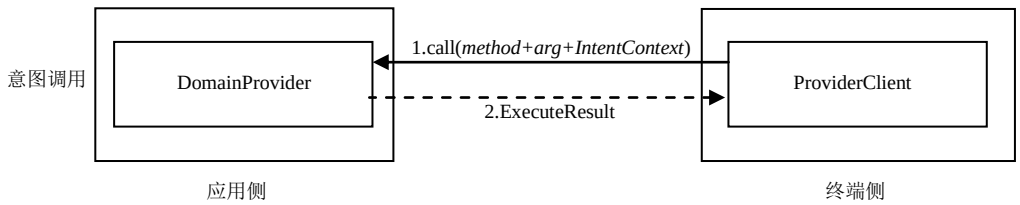

图 10　同步意图调用

应用开发示例见附录 A.6。

5.5.4.4 异步意图调用

异步接口系统会将 ExecteIntentReceiver 传递给应用，应用在规定 QoS 响应时间范围内将数据通过 ExecteIntentReceiver 的 send 方法传递给系统。异步意图调用流程见图 11。

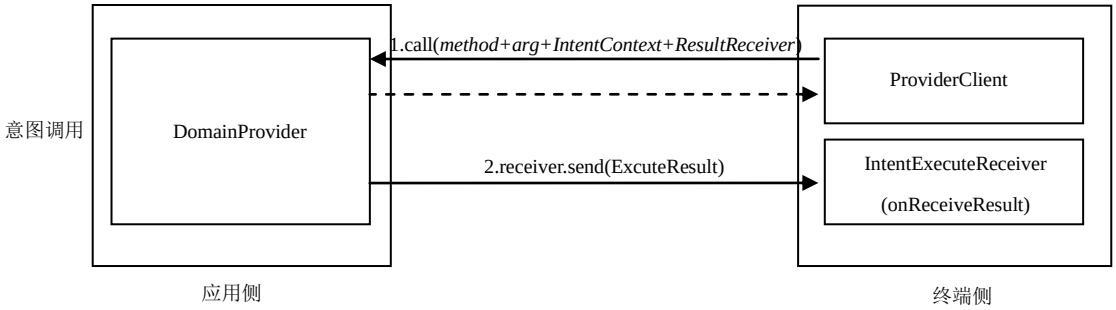

图 11　异步意图调用

应用开发示例见附录 A.7。

5.6　其他接口

5.6.1　概述

除了核心的意图共享、调用接口之外，还有部分工具类接口。

5.6.2　查询系统特性状态

查询系统特性状态流程见图 12。

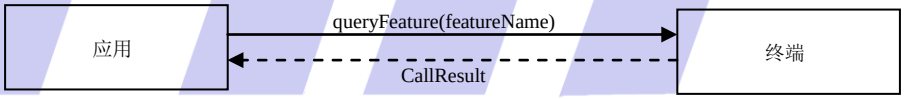

图 12　查询系统特性状态

CallResult：查询响应结果，见表 11。

表 11　查询响应结果

名称	类型	必选	说明
code	Int	是	调用状态码，0：成功，其他：失败。 10401001 - 应用无查询该特性权限； 10402001 - 特性查询请求参数错误； 10403001 - 未知特性。
message	String	是	错误消息
data	JSON	否	其他响应数据

系统特性状态查询包括：

——enableIntelligentIntent：查询当前系统是否开启意图框架能力，结果：

```
{
  "code": 0,
   "message": "success",
   "data": {
        "enableIntelligentIntent": true,
    }
}
```

——userAcceptShareIntent：查询用户是否同意某个应用共享意图，结果：

```
{
  "code": 0,
   "message": "success",
   "data": {
        "userAcceptShareIntent": true,
    }
}
```

——querySupportIntent：查询当前系统支持的意图列表，参数：["intentName1", "intentName2"]，结果：

```
{
  "code": 0,
   "message": "success",
   "data": {
        "querySupportIntent": [
          {
              "intentName": "intentName1",
              "support": true,
              "version": ["1.0.0", "1.0.1"]
          },
          {
              "intentName": "intentName2",
              "support": false,
              "version": null
          }
        ]
    }
}
```

应用调用示例见附录 A.8。

5.6.3 getSid()

getSid()流程见图 13。

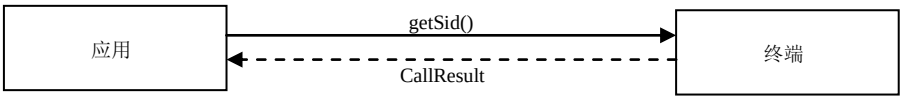

图 13　getSid()

getSid 接口用于云端意图共享阶段，意图共享需要绑定用户 ID 的场景。例如用户在端侧点外卖产生订单，后续的订单状态推进由应用云端意图共享发送到终端方云侧意图框架，终端方云侧意图框架需要知道该共享数据需要推送的具体设备，该 SID（Share Id）即终端方云侧意图框架意图共享数据推送需要关联到具体设备的标识。

CallResult：查询响应结果，见表 12。

表 12　getSid 查询响应结果

名称	类型	必选	说明
code	Int	是	调用状态码，0：成功，其他：失败。 10501001 - 无权限调用 getSid 接口； 10504001 - getSid 网络连接异常； 10503001 - getSid 调用次数超限； 10503002 - getSid 未知异常。
message	String	是	错误消息
data	JSON	否	**原生应用：**{ “sid”：“2fe3a970-efbb-29a0-0add-e5dbbf751ac0”，“type”：“oaid”，— 如：oaid/duid/phoneNumber “expire_in”：86400 }； **快应用：** 增加 getSid 接口。

应用调用示例见附录 A.9。

5.6.4　查询系统是否支持标准意图

通过 settings 数据库可以查询当前系统是否已支持标准意图，应用可以根据查询的结果来决定是否使能意图共享和意图调用能力。support_intelligent_intent > 0 代表当前系统已支持标准意图。

```
val support = Settings.Global.getInt(context.getContentResolver(), "support_intelligent_intent") > 0
```

5.7　错误码

错误码见表 13。

表 13　错误码规范

分类	来源类型（1 位）	功能类型（2 位）	错误类型（2 位）	错误编码（3 位）
枚举	1 - 系统（端侧）； 2 - 应用（端侧）； 3 - 终端方云侧； 4 - 应用方云侧。	01 - 意图共享； 02 - 意图删除； 03 - 意图调用； 04 - 特性查询； 05 - 其他类型。	01 - 权限类； 02 - 参数类； 03 - 业务类； 04 - 网络类； 05 - 其他类型。	如：001 - 无访问权限…

5.8 静态资源配置

在意图共享和意图调用阶段，存在图片或文件等资源类型的静态资源传输，应用需要告知意图框架如何获取对应的资源。

图片支持三种数据格式：

——**网络图片**：使用 https 地址即可，意图框架会去拉取对应的图片资源，需要注意的是：

- 网络图片可能会导致交互的延迟；
- 部分场景可能会导致网络图片加载失败影响效果（如开机启动、弱网、无网环境）。

——**base64 图片**：将图片数据编码成 base64 格式字符串，base64 格式数据由两部分组成，分别为头部数据和编译数据。只有两部分完整的数据，才能解析出正确的数据。如：
...
使用 base64 图片可以减少网络请求，提高加载效率，但是空间利用率低，使用该格式编码应保证图片的大小控制在 1K 内。

——**应用包内图片**：对应用包内图片要求如下：

- 应用包内图片统一使用 app://app_package/开头后接具体的路径，应保证相关资源在应用包内可被意图框架访问；
- 对于 android 应用支持从 assets 目录下获取静态资源，写法如下：
  android://com.xx.xx/assets/icon.png
- 对于 android 应用支持从 res 目录下获取静态资源，写法如下：
  android://com.xx.xx/res/icon.png
- 对于 android 应用支持从 FileProvider（FileProvider 是 ContentProvider 的子类，它让应用间共享文件变得更加容易。应用需要实现 FileProvider 共享文件，并授予 URI 访问权限）获取静态资源，写法如下：
  content://URI

6 意图接口云端技术方案

6.1 概述

云端意图接入包含意图注册、意图共享和意图调用三部分。

6.2 意图注册与服务创建

在意图注册之前需要创建终端方意图框架云服务，并生成应用的 client_id 和 client_secret。意图注册在终端方云端开发平台完成，需要填写的内容见表 14。

表 14 云端意图注册内容

参数名	类型	默认值	必填	描述
intentName	String	-	是	意图名称，名称应当遵循意图框架规范，当前仅支持预置垂域意图，不允许自定义
intentVersion	String	-	是	意图版本号，如“1.0”
executeMode	String	background	否	意图调用类型，云端只支持 background 模式

表 14　云端意图注册内容（续）

参数名	类型	默认值	必填	描述
executeEntry	String	-	否	意图调用的响应入口（GET）：https://aa.bb.com/ServiceEndpoint/， 参数支持
description	Array	-	否	用于意图框架推荐描述
providerAuthType	String	null	是	支持 OAuth2
providerOAuthRequest	String	null	否	curl 格式请求命令， 意图调用应用方有鉴权必填
providerOAuthTokenKey	String	null	否	providerOAuthRequest 的返回值若带授权码，则此处填写授权码的 key
providerOAuthExpireTimeKey	String	null	否	providerOAuthRequest 的返回值若带过期时间，则此处填写获取过期时间的 key
providerOAuthExpireTime	Int	null	否	providerOAuthRequest 返回结果若不带过期时间，则填写授权码的过期时间
providerOAuthTokenKey	String	null	否	访问 API 服务时，token 映射的字段名称
providerOAuthTokenArea	String	HEADER	否	访问 API 服务时 token 映射的字段存放位置(HEADER、BODY、QUERY)
clientAppVersion	String	-	否	关联应用及版本号， 格式 ： com.app.name:1234

6.3　接口鉴权

6.3.1　接口鉴权方式

为了保证云端数据传输安全，云端的接口调用需要做认证。意图共享和意图调用采用 OAuth 方式鉴权。

6.3.2　意图共享鉴权

6.3.2.1 意图共享鉴权流程

意图共享是应用云端与终端方云侧意图框架直接通信，调用方向为 应用云端 --> 终端方云侧意图框架，为保证调用安全，在调用过程中需要携带有效 access_token，流程见图 14，包括：

a)　应用向终端方申请意图成功后，会生成 client_id 和 client_secret；
b)　应用云端调用终端方云侧意图框架 token 获取接口，获取 access_token；
c)　后续业务需要携带 access_token，且内容需要按照约定的规则签名。

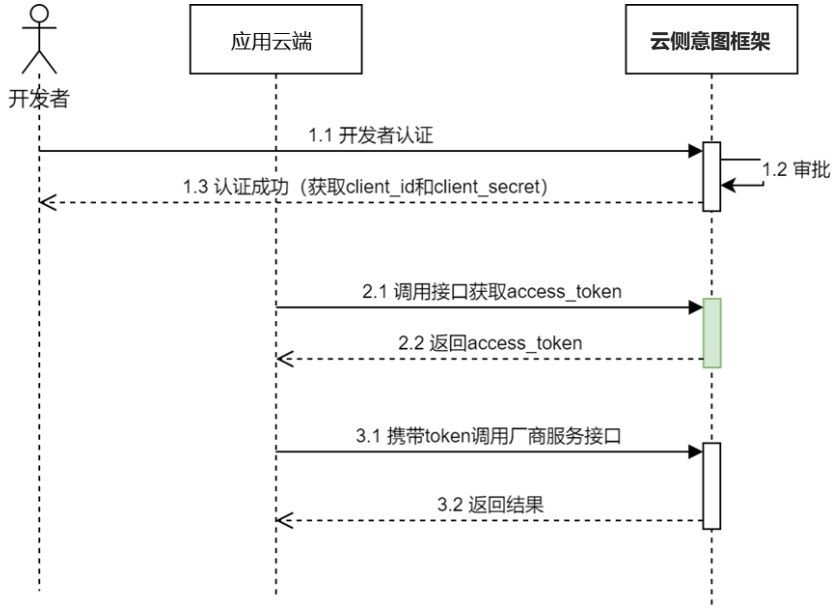

图 14　意图共享鉴权

6.3.2.2　access_token 获取

调用业务接口前需先获取授权 token（access_token），用于后续参数签名，应用开发者需要进行妥善保存。access_token 的有效期通过字段 expire_in 来标识，默认为 2 小时（7200 秒）有效。

重复调用获取 token 的接口，会将上一次获取的 token 的剩余有效期置为 5 分钟（300 秒）。

接口说明：

——请求路径：/intent/oauth2/v1/token；

——请求方法：GET；

——请求头：Content-type：x-www-form-urlencoded。

请求参数见表 15。

表 15　access_token 获取请求参数

参数名称	参数类型	是否必须	参数描述
client_id	string	是	终端方开放平台申请分配的 client_id，18 位字符串
client_secret	string	是	终端方开放平台申请分配的 client_secret，与 client_id 配对，64 位字符串
grant_type	string	是	固定值“client_credentials”

响应体见表 16。

表 16　access_token 获取响应体

参数名称	参数类型	参数描述	备注
code	int	0 表示成功，非 0 表示失败	-
message	string	错误信息描述	-

表 16　access_token 获取响应体（续）

参数名称	参数类型	参数描述	备注
data	access_token	string	访问 token
	expire_in	int	有效时间（秒），7200 内的值

错误码见表 17。

表 17　access_token 获取错误码

code	错误描述
0	请求成功
30502001	参数错误
30502002	签名错误
30503001	系统异常
30501001	IP 被禁止

响应体示例见附录 A.10。

6.3.2.3　数据签名

应用云端在请求时需要添加如表 18 所示请求头。

表 18　数据签名请求头

Header	类型	默认值	是否必选	说明
Authorization	String	-	是	用户获取的 access_token
X-Client-Send-Utc-Ms	String	-	是	发起请求时的时间戳，毫秒级，误差允许 15 分钟
X-Nonce	String	-	是	0-20000 范围内的随机数，接口需要签名时必须设置
X-Api-Sign	String	-	是	请求签名，接口需要签名时必须设置
X-Api-Sign-Type	string	sort	否	请求签名类型，不传默认 sort，即字段排序签名

签名逻辑的输出结果用来填充 X-Api-Sign 字段，发起请求前需要对参数进行签名，云侧意图框架会对应用云端请求数据进行校验，签名规则如下：

a)　所有请求参数按照 ASCII 升序排序；
b)　请求参数使用&拼接字符串，值为 null 的参数不参与签名，拼接成 k1=v1&k2=v2；
c)　将 access_token 、时间戳、随机数与第 2 步中的字符串按指定格式拼接；
d)　对第 3 步得到的字符串进行 HmacSHA256 计算，计算时使用的密钥 key 为获取使用的 client_secret；
e)　将 hash 计算结果转换为小写 16 进制字符串，得到签名 sign。

当请求 Body 为 JSON 时，签名时取第一层字段排序签名，值为数组或 Map 的第一层字段，值转换为

JSON 字符串。

预签名字符串格式为：

access_token=${access_token}&timestamp=${timestamp}&nonce=${nonce} + body 参数按 key 排序后拼接。

签名代码逻辑示例见附录 A.11。

6.3.3 意图调用鉴权

意图调用接口鉴权支持 OAuth 接口方式鉴权，在意图注册的时候需要填写应用方服务器鉴权接口信息，见表 19。

表 19 意图调用鉴权接口信息

参数名	类型	默认值	必填	描述
providerAuthType	String	null	是	支持 OAuth2 和 None
providerOAuthRequest	String	null	否	curl 格式请求命令，意图调用应用侧有鉴权必填
providerOAuthTokenKey	String	null	否	providerOAuthRequest 的返回值若带授权码，则此处填写授权码的 key
providerOAuthExpireTimeKey	String	null	否	providerOAuthRequest 的返回值若带过期时间，则此处填写获取过期时间的 key
providerOAuthExpireTime	Int	null	否	providerOAuthRequest 返回结果若不带过期时间，则填写授权码的过期时间
providerOAuthTokenKey	String	null	否	访问 API 服务时，token 映射的字段名称
providerOAuthTokenArea	String	HEADER	否	访问 API 服务时 token 映射的字段存放位置（HEADER、BODY、QUERY）

意图调用鉴权是应用云端对云侧意图框架的调用进行鉴权，流程见图 15，描述如下：

a) 应用开发者在应用云端申请账号，并获取账号对应的 client_id 和 client_secret；
b) 应用开发者在终端系统根据意图注册的注册规范填写 client_id 和 client_secret 等信息；
c) 云侧意图框架在发起意图调用前向应用云端发起调用请求 access_token；
d) 云侧意图框架根据 access_token 的有效时间对 access_token 进行缓存；
e) 云侧意图框架按照应用云端授权接口要求和垂域字段要求，构造意图调用请求；
f) 云侧意图框架向应用云端发起意图调用请求，应用云端根据垂域规范返回结构化数据。

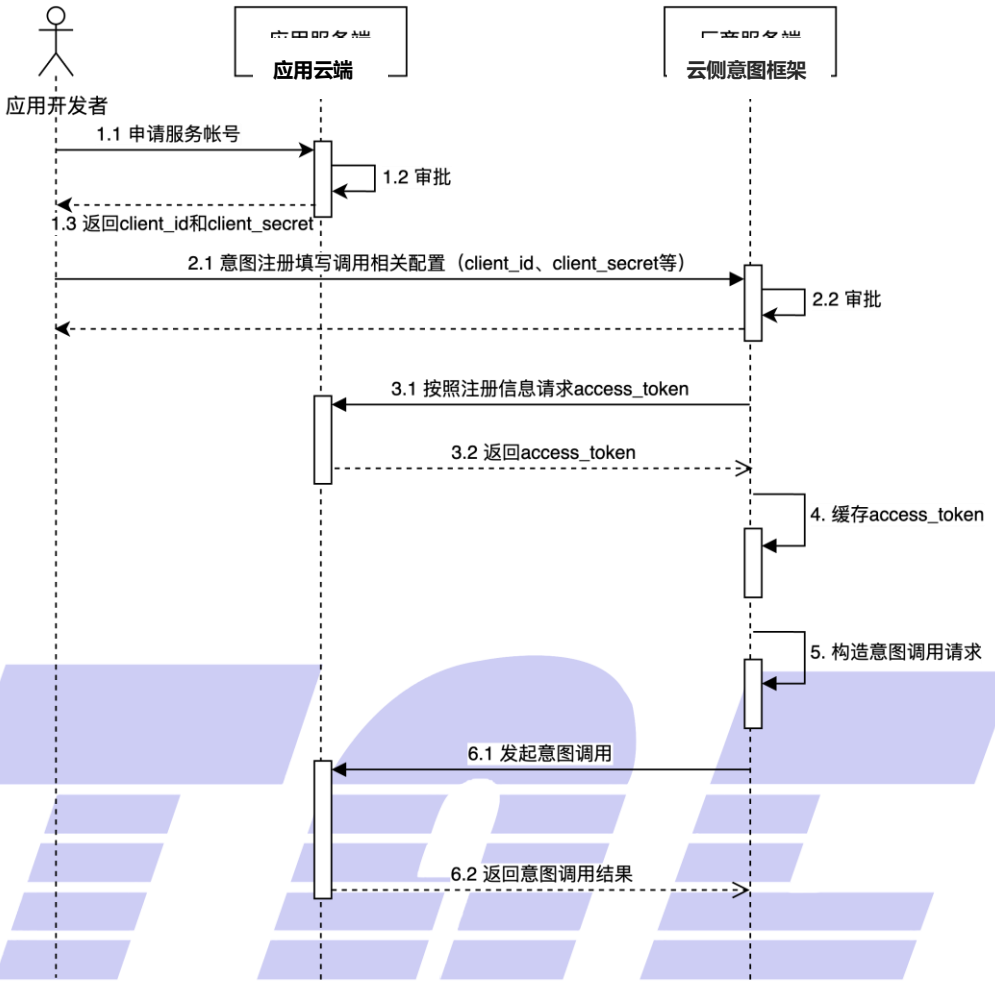

图 15　意图调用鉴权

6.4　意图共享

6.4.1　共享单个意图

6.4.1.1　共享单个意图概述

意图根据面向的人群不同，可以分为面向特定人群的事件共享和面向指定用户的用户共享：

——事件共享：该类共享面向特定人群，比如在天气异常的时候提醒某个特定区域内的所有用户

——用户共享：该类共享面向某个特性用户，如外卖点餐进度提醒，需要应用方服务账号与终端方账号进行绑定。

这两类共享均使用同一接口：

——请求路径：/intent/v1/shareIntent；

——请求方法：POST；

——请求头：

- Content-type: application/json；
- Authorization: 88F5CEZsArO...wBhS；
- X-Client-Send-Utc-Ms: 1729654789000；

- X-Nonce: 15；
- X-Api-Sign: 88f5...a012；
- X-Api-Sign-Type: sort。

——请求体：

```
{
    "intelligentIntent": {
        "intentName": "意图名称",
        ...
    },
    "requestId": "4f93a967effb29a10bdae5c0bf701ac4",
    "target": {
        // 用户类型
        "targetType": "user",
        // 分发目标 ID，需要绑定终端方账号
        "targetIds": ["2fe3a970-efbb-29a0-0add-e5dbbf751ac0"],
        // 设备标识类型
        "idType": "oaid"
    }
}
```

intelligentIntent：填写意图共享内容，参考 IntelligentIntent 数据结构。

target 类型分为两种 group 和 user 分别对应事件共享和用户共享两种类型：

——事件共享：

```
{
    // 用户类型
    "targetType": "group",
    // 用户标签
    "tags": ["young"]
}
```

——用户共享：

```
{
   // 用户类型
    "targetType": "user",
    // 设备标识，需要绑定设备账号
    "targetIds": ["2fe3a970-efbb-29a0-0add-e5dbbf751ac0"],
    // 设备标识类型
    "idType": "oaid"
}
```

指定用户共享需要携带用户标识 targetId，该标识需要通过账号绑定获取流程。

6.4.1.2 账号绑定流程 1（应用方记录映射关系）

意图框架端侧提供获取 SID（ShareId）接口，在云端用户意图共享场景，需要知道意图数据推送

给某台具体设备。流程见图 16，包括：

a) 应用端侧调用 getSid 接口获取 SID；
b) 应用端侧业务触发（如有订单产生）将 SID 传递给应用云侧，并在应用云侧完成事件 SID 与应用用户账号的绑定；
c) 应用方云侧意图共享数据的时候携带 SID；
d) 云侧意图框架收到应用云侧共享的数据后适时将数据下发到用户端侧意图框架；
e) 端侧意图框架决定在合适的入口推出服务卡片。

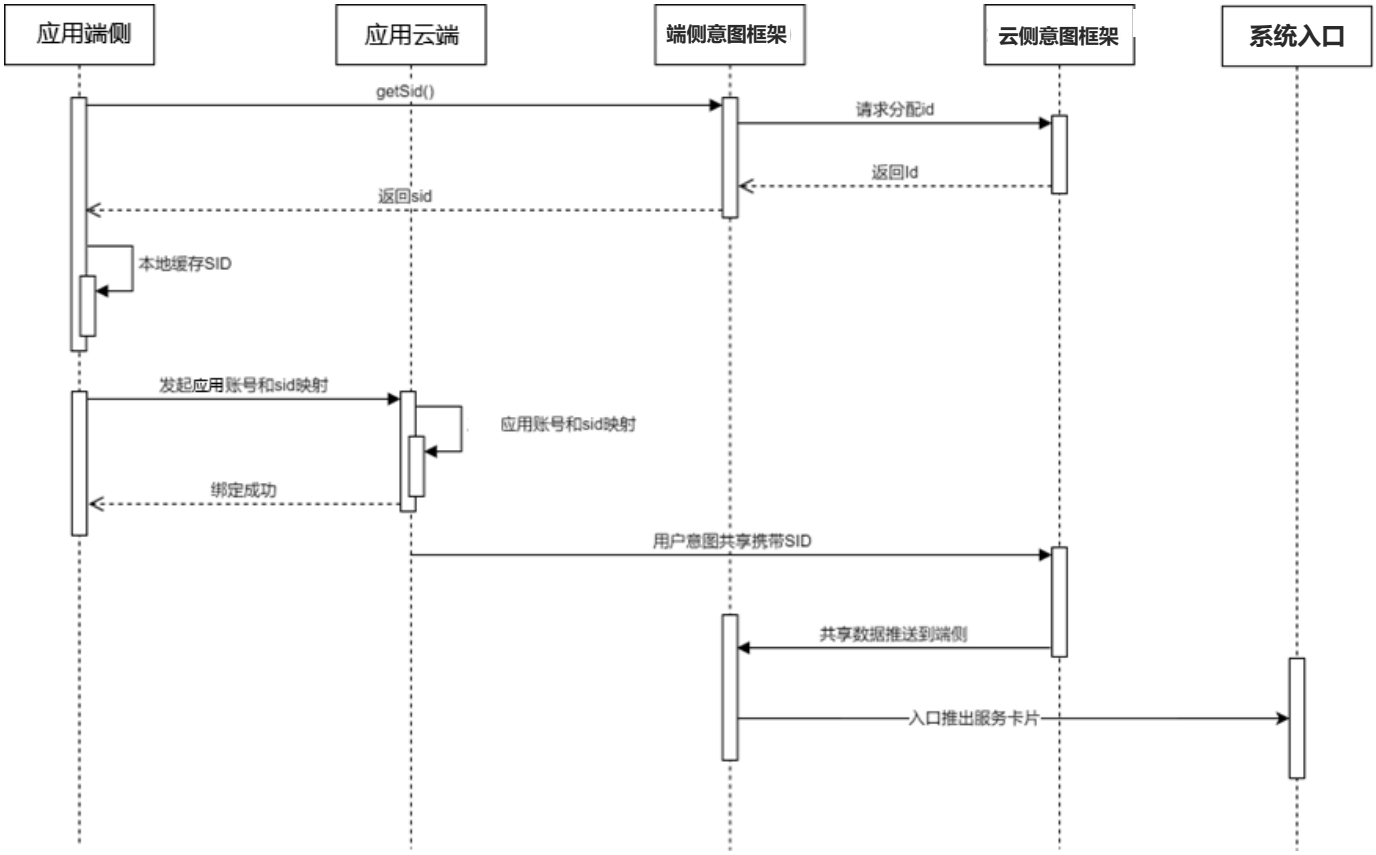

图 16　账号绑定流程 1

响应体字段见表 20，示例见附录 A.12。

表 20　账号绑定流程 1 响应体

字段名称	类型	字段说明
code	Int	0 表示成功，非 0 表示失败
message	String	错误信息描述
data	JSON	-
├identifier	string	请求 ID

6.4.1.3　账号绑定流程 2（终端方记录映射关系）

由终端方绑定应用账号，需要用户授权，绑定关系记录在终端方，基本流程见图 17。

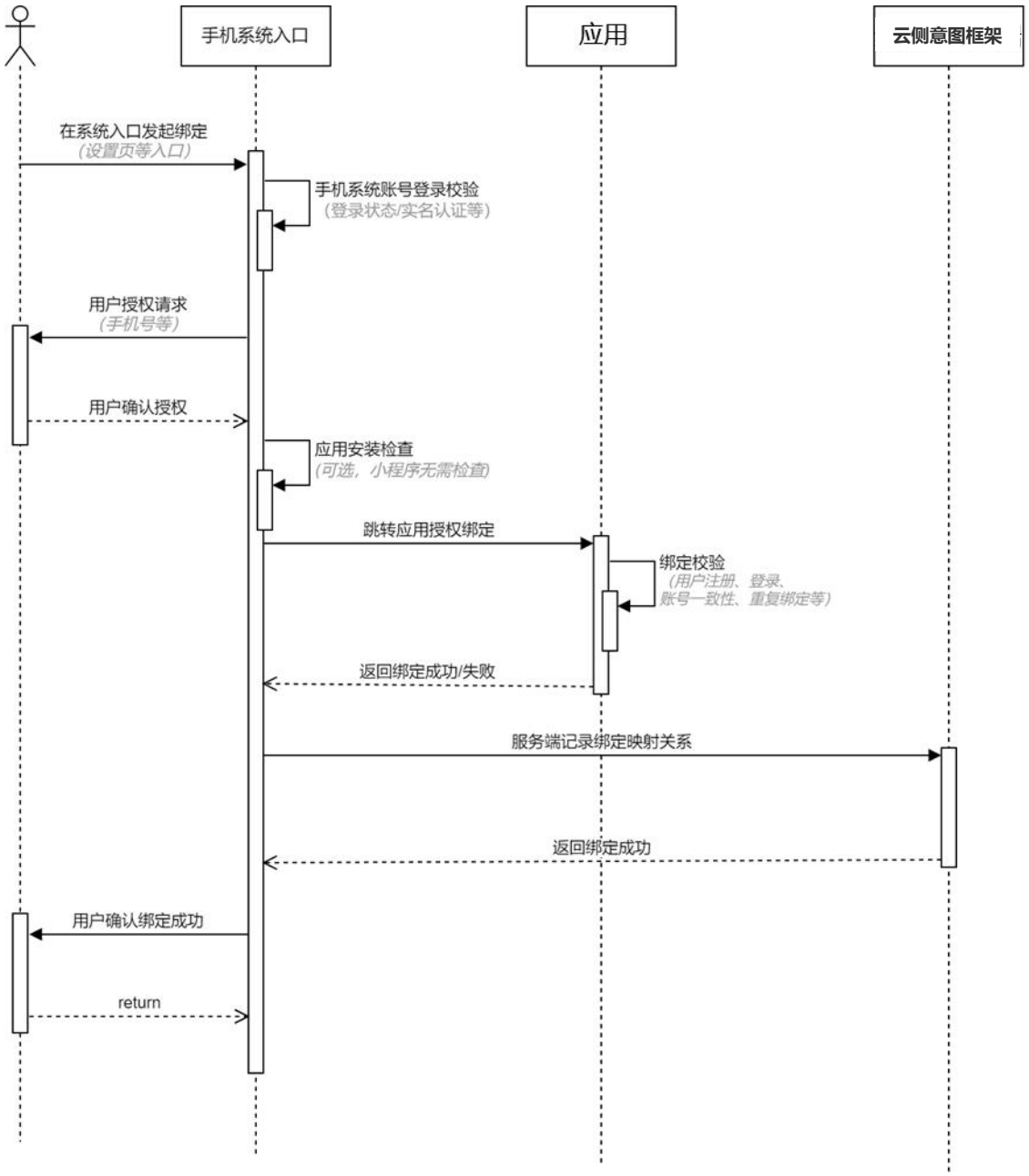

图 17　账号绑定流程 2

此场景在意图共享阶段，需要传递的应用账号为绑定阶段返回给终端系统的账号信息，方法如下：

```
{
    // 用户类型
    "targetType": "user",
    // 应用方应用账号
    "identifiers": ["应用账号"],
    // 设备标识类型
```

```
    "idType": "appAccount"
}
```

6.4.2 意图删除接口

6.4.2.1 deleteIntent

请求路径： /intent/v1/deleteIntent
请求方法： POST
请求头：

Content-type: application/json
Authorization: 88F5CEZsAr0…wBhS
X-Client-Send-Utc-Ms: 1729654789000
X-Nonce: 15
X-Api-Sign: 88f5…a012
X-Api-Sign-Type: sort

请求体：

```
{
    "data": {
        "intentName": "意图名称",
        "identifiers": ["xxxxxx"]
    },
    "requestId": "4f93a967effb29a10bdae5c0bf701ac4",
    "target": {
        // 用户类型
        "targetType": "user",
        // 分发目标 ID，需要绑定终端方账号
        "targetIds": ["2fe3a970-efbb-29a0-0add-e5dbbf751ac0"],
        // 设备标识类型
        "idType": "oaid"
    }
}
```

参数说明见表 21。

表 21 deleteIntent 参数

参数名称	参数类型	是否必须	参数描述
intentName	String	是	意图名称
identifiers	Array	是	意图 id 列表

响应体见表 22。

表 22　deleteIntent 响应体

字段名称	类型	字段说明
code	int	0 表示成功，非 0 表示失败
message	string	错误信息描述
data	JSON	未定义

错误码见表 23。

表 23　deleteIntent 错误码

code	错误描述
0	请求成功,非 0 表示失败

6.4.2.2　deleteEntity

请求路径：/intent/v1/deleteEntity
请求方法：POST
请求头：

Content-type: application/json
Authorization: 88F5CEZsAr0…wBhS
X-Client-Send-Utc-Ms: 1729654789000
X-Nonce: 15
X-Api-Sign: 88f5…a012
X-Api-Sign-Type: sort

请求体：

```
{
    "data": {
        "intentName": "意图名称",
        "entityIds": ["xxxxxxxxxxxxxx"]
    },
    "requestId": "4f93a967effb29a10bdae5c0bf701ac4",
    "target": {
        // 用户类型
        "targetType": "user",
        // 分发目标 ID，需要绑定终端方账号
        "targetIds": ["2fe3a970-efbb-29a0-0add-e5dbbf751ac0"],
        // 设备标识类型
        "idType": "oaid"
    }
}
```

参数说明见表 24。

表 24　deleteEntity 参数

参数名称	参数类型	是否必须	参数描述
intentName	String	是	意图名称
entityIds	Array	是	实体 id 列表

响应体见表 25。

表 25　deleteEntity 响应体

字段名称	类型	字段说明
code	int	0 表示成功，非 0 表示失败
message	string	错误信息描述
data	JSON	未定义

错误码见表 26。

表 26　deleteEntity 错误码

code	错误描述
0	请求成功

6.5　意图调用接口

云侧意图调用传递由终端方云侧意图框架调用应用云端，调用接口由应用开发者在意图注册的时候声明接口 https 入口，接口调用规则如下：

——调用的接口 method 为 GET；

——url 路径允许拼接意图上下文参数，拼接形式为

intentParams=base64(JSONString(IntentParams))。

意图上下文定义 IntentParams 见表 27。

表 27　意图上下文参数

名称	类型	必选	说明
intentName	String	是	意图名称
requestId	String	是	本次意图调用的唯一 id，应用完成意图调用逻辑后通知系统返回需要回传该参数
device	String	是	触发意图调用的设备信息，可选值： phone/watch/tablet/tv/car
entry	String	是	触发意图调用的入口用户信息，如语音助手
entityId	String	否	当同一入口有多个实例的情况，用于区分是哪一个实例发起的调用。如有用户购买了多张车票，需要查询具体某一张车票信息
parameters	JSON	否	意图调用参数，应用方应用解析，如地理位置坐标/用户习惯等，不同垂域应定义不同内容

表 27　意图上下文参数（续）

名称	类型	必选	说明
extras	JSON	否	额外的意图参数

返回数据格式见表 28。

表 28　意图调用返回数据格式

名称	类型	必选	说明
code	Int	是	调用状态码，0：成功，其他：失败
message	String	是	错误消息
data	JSON	否	响应结果其他附加内容，不同垂域需不同定义

附 录 A
（资料性）
数据和代码示例

A.1 意图共享数据示例

图 A.1 打车示意图

意图注册：

```
{
  "intelligentIntents": [
    {
      "intentName": "Navigation.StartNavigation",
      "intentVersion": "1.0.0",
      "desc": ["打车订单"]
    }
  ]
}
```

意图共享：

```
{
  "intentName": "Ridehailing.RecommendRide",
  "identifier": "e82d498d4c0dcab8e82d498d4c0dcab8",
  "timestamp": 1740402512123,
  "serviceId": ["4293970000"],
  "intentAction": {
    "actionType": "fact",
    "actionTime": {
      "startTime": 1729485000989,
      "endTime": 1729485000989,
    }
  },
  "intentEntity": {
    "entityName": "",
    "entityId": "202408211736e82d498d4c0dcab8",
    "isPublic": false,

    "title": "5 分钟接驾",
    "shortTitle": "5 分钟接驾",
    "content": "京 XXXXXX 白·奥迪 A6L",
    "shortContent": "京 XXXXXX 白",
    "brandName": "XX 打车",
    "brandLogo": "https://www.xxx.com/XXXX.png",
    "capsule": "5 分钟接驾",
    "shortCapsule": "接驾中",
    "capsuleContent": "5 分钟接驾",
    "tts": "京 XXXXXX 白色奥迪 A6L，预计 5 分钟接驾",

    "orderStatus": 103,
    "image": "https://www.xxx.com/car.png",
    "buttonDesc": "联系司机",
    "buttonDeep link": "onetravel://activity?id=TWpreE9ESXlO",
    "progress": 20,
    "progressCount": 2
  }
}
```

A.2　同步接口call方法代码示例

```
// 获取authorities
val authorities = Settings.Global.getString(context.getContentResolver(),
"intelligent_intent_authorities") ?: "IntelligentIntent"
```

```
// 获取ContentProviderClient
val client = context
          .contentResolver
          .acquireContentProviderClient(authorities)

// 调用call方法共享数据
val resultBundle = client.call("shareIntent", null,
                               Bundle().apply {
    // intentData的内容为JSON String
    // 格式参考IntelligentIntent数据结构描述
    putString("intentData", "{...}")
})
// 从resultBundle中获取ShareResult的响应结构，格式为JSON String
val shareResult = resultBundle.getString("result")
val code = JSONObject(shareResult).getInt("code")
if (code == 0) {
  // 共享数据成功
}
```

A.3 异步接口call方法代码示例

继承 ResultReceiver 类，重写 onReceiveResult 方法：

```
class MyShareIntentReceiver(handler: Handler?)
                            : ResultReceiver(handler) {
    // 响应结果将在此函数中实现
    override fun onReceiveResult(resultCode: Int,
                                 resultData: Bundle?) {
        // resultData 数据格式为：
        // - method: 回传 call 方法调用时的方法名，如 shareIntentAsync
        // - result: 格式为 ShareResult 的 JSON String
        // 开发者可以解析该数据格式获取响应结果
        if (resultCode != 0) { // 失败
            return
        }

        val method = resultData.getString("method")
        if (method == "shareIntentAsync") {
            val result = JSONObject(resultData.getString("result"))
            val code = result.getInt("code")
            if (code == 0) {
                // 共享成功
            }
        }
```

```
    }
}
```

调用异步共享：

```
// 共享数据结果处理器
val receiver = MyShareIntentReceiver(Handler(Looper.getMainLooper()))

// 获取 ContentProviderClient
val client = context
          .contentResolver
          .acquireContentProviderClient("IntelligentIntent")

// 调用 call 方法共享数据
val callResultBundle = client.call("shareIntentAsync", null,
                                   Bundle().apply {
    // intentData 的内容为 JSON String
    // 格式参考 IntelligentIntent 数据结构描述
    putString("intentData", "{...}")

    // 传递意图共享结果接收器
    putParcelable("shareIntentReceiver", receiverForSending(receiver))
})

private fun receiverForSending(actualReceiver: ResultReceiver): ResultReceiver {
    val parcel = Parcel.obtain()
    actualReceiver.writeToParcel(parcel, 0)
    parcel.setDataPosition(0)
    val receiverForSending = ResultReceiver.CREATOR.createFromParcel(parcel)
    parcel.recycle()
    return receiverForSending
}
```

A.4　deleteIntent方法示例

```
// 获取 ContentProviderClient
val client = context
          .contentResolver
          .acquireContentProviderClient("IntelligentIntent")

// 调用 call 方法共享数据
val resultBundle = client.call("deleteIntent", null,
                               Bundle().apply {
    // 意图名称
```

```
    putString("intentName", "Navigation.StartNavigation")

    // 意图 id 列表
    putStringArrayList("identifiers", arrayListOf("id1", "id2"))
})
// 从 resultBundle 中获取 ShareResult 的响应结构，格式为 JSON String
val shareResults = resultBundle.getString("result")
// 获取意图删除的响应结果
val code = JSONObject(shareResults).getInt("code")
if (code == 0) {
  // 删除意图数据成功
}
```

A.5 deleteEntity方法示例

```
// 获取 ContentProviderClient
val client = context
        .contentResolver
        .acquireContentProviderClient("IntelligentIntent")

// 调用 call 方法共享数据
val resultBundle = client.call("deleteIntent", null,
                               Bundle().apply {
    // 意图名称
    putString("intentName", "Navigation.NavigationInfo")

    // 实体 id 列表， entityIds
    putStringArrayList("entityIds", arrayListOf("abc123"))
})
// 从 resultBundle 中获取 ShareResult 的响应结构，格式为 JSON String
val shareResults = resultBundle.getString("result")
// 获取意图删除的响应结果
val code = JSONObject(shareResults).getInt("code")
if (code == 0) {
  // 删除意图数据成功
}
```

A.6 同步意图调用代码示例

AndroidManifest.xml：

```
<provider
    android:name="com.xxxmap.MyNavigationIntentProvider"
    android:authorities="com.xxxmap.intelligent.provider"
    android:enabled="true"
```

```
    android:exported="true" >
    <intent-filter>
        <action android:name="intelligent.action.ACTION_EXECUTE_INTENT" />
    </intent-filter>
</provider>
```

编写意图调用的 ContentProvider，如：MyNavigationIntentProvider.kt：

```
class MyNavigationIntentProvider : ContentProvider() {
    ...

    override fun call(authority: String, method: String,
                      arg: String?, extras: Bundle?): Bundle {
        super.call(authority, method, arg, extras)
        // 1. 基本安全检查，应用方也可包含其他检查条件
        // 1.1 从意图上下文取出调用 id
        val requestId = intentParams.getString("requestId")
        if (Binder.getCallingUid() != Process.SYSTEM_UID) {
            // 校验失败返回
            return Bundle().apply {
                putString("result", JSONObject().apply {
                    putInt("code", 1)
                    putString("message", "has no permission")
                    putString("requestId", requestId)
                }.toString())
            }
        }

        // 2. 解析参数
        // 2.1 获取意图上下文内容
        val intentParams = JSONObject(extras!!.getString("intentParams"))

        // 3 处理意图请求
        // 3.1 处理搜索周边意图
        if (method == "Navigation.SearchArround" && isSync) {
          val result = queryNavigation(intentParams)
          return Bundle().apply {
                putString("result", JSONObject().apply {
                        putInt("code", 0)
                        putString("message", "success")
                        putString("requestId", requestId)
                        putString("data", JSONObject(result).toString())
                    }.toString())
                }
```

```
            }
        else {
            // 3.2 不支持场景
            return Bundle().apply {
                putString("result", JSONObject().apply {
                    putInt("code", 2)
                    putString("message", "not support")
                    putString("requestId", requestId)
                }.toString())
            }
        }
    }
}
```

A.7 异步意图调用代码示例

AndroidManifest.xml：

```
<provider
    android:name="com.xxxmap.intelligent.MyNavigationIntentProvider"
    android:authorities="com.xxxmap.intelligent.provider"
    android:enabled="true"
    android:exported="true" >
    <intent-filter>
        <action android:name="intelligent.action.ACTION_EXECUTE_INTENT" />
    </intent-filter>
</provider>
```

编写意图调用的 ContentProvider，如：MyNavigationIntentProvider.kt：

```
class MyNavigationIntentProvider : ContentProvider() {
    ...

    override fun call(authority: String, method: String,
                      arg: String?, extras: Bundle?): Bundle {
        super.call(authority, method, arg, extras)
        // 1. 基本安全检查，应用方也可包含其他检查条件
        // 1.1 从意图上下文取出调用 id
        val requestId = intentParams.getString("requestId")
        // 1.2 判断是否异步调用
        val isSync = intentParams.getBoolean("executeSync") ?: true
        // 1.3 获取 ExecuteIntentReceiver 对象
        val receiver = extras.getParcelable("executeReceiver",
                                    ResultReceiver::class.java)
        if (Binder.getCallingUid() != Process.SYSTEM_UID || isSync) {
            // 1.1 校验失败返回
```

```
            receiver.send(1, Bundle().apply {
                putString("result", JSONObject().apply {
                    putInt("code", 1)
                    putString("message", "has no permission")
                    putString("requestId", requestId)
                }.toString())
            })
            return Bundle();
        }
        // 2. 解析参数
        // 2.1 获取意图上下文内容
        val intentParams = JSONObject(extras!!.getString("intentParams"))

        // 2.3 处理查询导航路线意图
        if (method == "Navigation.QueryNavigation" && !isSync ) {
            // 2.3.2 异步执行业务逻辑
            Thread({
                queryNavigationAsync(intentParams, callId, receiver)
            }).start()

            // 2.3.3 返回
            return Bundle()
        } else if (method == "Navigation.SearchArround" && isSync) {
            // 2.4 处理搜索周边意图
            ...
        } else {
            // 2.5 不支持场景
            receiver.send(2, Bundle().apply {
                putString("result", JSONObject().apply {
                    putInt("code", 2)
                    putString("message", "not support")
                    putString("requestId", requestId)
                }.toString())
            })
        }
      return Bundle()
    }

    /**
     * 2.6 异步处理业务逻辑并返回
     */
    private fun queryNavigationAsync(
        intentParams: JSONObject,
```

```
        callId: String,
        receiver: ExecuteIntentReceiver
    ) {
        val result = queryNavigation(intentParams)
        receiver.send(0, Bundle().apply {
            putString("result", JSONObject().apply {
                putInt("code", 0)
                putString("message", "success")
                putString("requestId", requestId)
                put("data", JSONObject(result))
            }.toString())
        })
    }
}
```

A.8 查询系统特性状态代码示例

```
// 获取 ContentProviderClient
val client = context
        .contentResolver
        .acquireContentProviderClient("IntelligentIntent")

// 调用 call 方法查询特性
val featureName1 = "featureName1"
val featureName2 = "featureName2"
val featureNames = listOf(featureName1, featureName2).joinToString(",")
// 支持传递多个 featureName， 用','分割
val callResultBundle = client.call("queryFeature", featureNames, Bundle().apply {
    putString("featureArgs", """
                            {
                                "featureName1": ["arg1", "arg2"],
                                "featureName2": ["arg1", "arg2"],
                            }
                            """)
})
val callResult = JSONObject(callResultBundle.getString("result"))
// 从 callResult 中获取响应结果
val code = callResult.getInt("code")
if (code == 0) {
    // 获取结果
    val enable= JSONObject(callResult.getString("data"))
                .getBoolean(featureName1)
}
```

A.9 getSid（）方法示例

```
// 获取 ContentProviderClient
val client = context
          .contentResolver
          .acquireContentProviderClient("IntelligentIntent")

// 调用 call 方法获取 SID
val callResultBundle = client.call("getSid", null, null)
val callResult = JSONObject(callResultBundle.getString("result"))
// 从 callResult 中获取响应结果
val code = callResult.getInt("code")
if (code == 0) {
    // 获取结果
    val sid = JSONObject(callResult.getString("data"))
                .getString("sid")
}
```

A.10 access_token响应体示例

```
{
    "code": 0,
    "message": "Success",
    "data": {
        "access_token":
"88F5CEZsArOaNOSNKc2a8LiGMjEeHTSg2ewSgpW8W8q8X8n18msa2Oem522w8B2W5212M5HfpY558828fb5F
FfK2UqnZR2W8wBhS",
        "expire_in": 7200
    }
}
```

A.11 签名代码逻辑示例

```
/**
 * 对请求参数进行签名
 * @param accessToken  /oauth2/v1/token 接口获取到的 token 值
 * @param timestamp    当前时间戳
 * @param nonce         随机数
 * @param secret        开放平台申请的 client_secret
 * @param paramsMap    业务数据
 * @return String       签名字符串
 * @throws IOException
 */
String sign(String accessToken,
                String timestamp,
```

```
                String nonce,
                String secret,
                Map<String, Object> paramsMap){
    // 对 key 排序
    List<String> keysList = new ArrayList<>(paramsMap.keySet());
    Collections.sort(keysList);

    // 拼接待签名字符串
    List<String> paramList = new ArrayList<>();
    for (String key : keysList) {
        Object object = paramsMap.get(key);
        if (object == null){
            continue;
        }
        String value = key + "=" + object;
        paramList.add(value);
    }
    String s = "access_token="+accessToken+"&timestamp="+timestamp+"&nonce="+nonce;
    if(!paramList.isEmpty()){
        String signStr = String.join("&", paramList);
        s = s + "&" +signStr;
    }
    // hmacSHA256 签名
    return hmacSHA256(s, secret);
}

/**
 * HMAC_SHA256 计算签名
 * @param data 需要加密的参数
 * @param key 签名密钥
 * @return String 返回加密后字符串
 */
public static String hmacSHA256(String data, String key) {
    try {
        byte[] secretByte = key.getBytes(Charset.forName("UTF-8"));
        SecretKeySpec signingKey = new SecretKeySpec(secretByte, "HmacSHA256");
        Mac mac = Mac.getInstance("HmacSHA256");
        mac.init(signingKey);
        byte[] dataByte = data.getBytes(Charset.forName("UTF-8"));
        byte[] by = mac.doFinal(dataByte);
        return byteArr2HexStr(by);
    } catch (Exception e) {
        e.printStackTrace();
```

```
    }
    return null;
}
```

A.12　账号绑定流程1响应体

```
{
    "code": 0,
    "message": "Success",
    "data": {
        "identifier": "xxxxxxxxxxxxxxx"
    }
}
```

电信终端产业协会团体标准

智能终端意图框架接口技术要求

T/TAF 283—2025

*

电信终端产业协会发布
地址：北京市西城区新街口外大街 28 号
电话：010-82052809
电子版发行网址：www.taf.org.cn