**喜马拉雅SDK接入文档**

目录

[为了便于喜马拉雅对播放数据进行统计，接入SDK时必须接入播放器逻辑，否则喜马拉雅有权直接封停接口。 6](#_Toc439244724)

[1概述 6](#_Toc439244725)

[2接入流程 6](#_Toc439244726)

[2.1申请PACKID、APPKEY和APPSECRET 6](#_Toc439244727)

[2.2 SDK依赖的第三方jar包 6](#_Toc439244728)

[2.2配置应用工程的AndroidManifest.xml 7](#_Toc439244729)

[2.2.1添加权限 7](#_Toc439244730)

[2.2.2在AndroidManifest.xml中添加meta-data 7](#_Toc439244731)

[3编码接入详细介绍 7](#_Toc439244732)

[3.1初始化接口（必接） 7](#_Toc439244733)

[3.1.1设置APP秘钥 7](#_Toc439244734)

[3.1.2 设置网络配置 8](#_Toc439244735)

[3.2获取点播内容 9](#_Toc439244736)

[3.2.1获取喜马拉雅内容分类 9](#_Toc439244737)

[3.2.2获取专辑标签或者声音标签 10](#_Toc439244738)

[3.2.3根据分类和标签获取某个分类某个标签下的热门专辑列表 11](#_Toc439244739)

[3.2.4根据专辑ID获取专辑下的声音列表，即专辑浏览 12](#_Toc439244740)

[3.2.5批量获取专辑列表 13](#_Toc439244741)

[3.2.6根据分类和标签获取某个分类某个标签下的热门声音列表 14](#_Toc439244742)

[3.2.7批量获取声音 15](#_Toc439244743)

[3.2.8搜索专辑 16](#_Toc439244744)

[3.2.9搜索声音 17](#_Toc439244745)

[3.2.10获取运营人员在首页推荐的专辑分类 19](#_Toc439244746)

[3.2.11获取全量专辑数据 19](#_Toc439244747)

[3.2.12根据专辑ID列表获取批量专辑更新提醒信息列表 20](#_Toc439244748)

[3.2.13获取某个专辑的相关推荐 21](#_Toc439244749)

[3.2.14获取下载听模块的推荐下载专辑 22](#_Toc439244750)

[3.2.15根据上一次所听声音的id，获取此声音所在那一页的声音 23](#_Toc439244751)

[3.2.16获取最新热搜词 23](#_Toc439244752)

[3.2.17获取某个关键词的联想词 24](#_Toc439244753)

[3.2.18获取某个声音的相关推荐专辑 25](#_Toc439244754)

[3.2.19获取猜你喜欢的专辑 26](#_Toc439244755)

[3.2.20获取主播的专辑列表 26](#_Toc439244756)

[3.2.21根据分类和标签获取某个分类某个标签下的最新专辑列表 27](#_Toc439244757)

[3.3获取直播内容 28](#_Toc439244758)

[3.3.1获取直播省市列表 28](#_Toc439244759)

[3.3.2获取直播电台 29](#_Toc439244760)

[3.3.3获取直播电台某一天的节目排期表 30](#_Toc439244761)

[3.3.4获取直播节目详情 31](#_Toc439244762)

[3.3.5搜索直播 32](#_Toc439244763)

[3.3.6批量获取电台接口 33](#_Toc439244764)

[3.3.7获取某个城市下的电台列表 34](#_Toc439244765)

[3.4获取榜单内容 34](#_Toc439244766)

[3.4.1根据榜单类型获取榜单首页的榜单列表 34](#_Toc439244767)

[3.4.2根据rank\_key获取某个榜单下的专辑列表 35](#_Toc439244768)

[3.4.3根据rank\_key获取某个榜单下的声音列表 36](#_Toc439244769)

[3.4.4获取直播电台排行榜 37](#_Toc439244770)

[3.5获取听单内容 38](#_Toc439244771)

[3.5.1获取精品听单内容 38](#_Toc439244772)

[3.5.1获取精品听单内容 38](#_Toc439244773)

[3.6获取焦点图内容 39](#_Toc439244774)

[3.6.1获取榜单的焦点图列表 39](#_Toc439244775)

[3.6.2获取发现页推荐的焦点图列表 40](#_Toc439244776)

[3.6.3获取分类推荐的焦点图列表 41](#_Toc439244777)

[3.7冷启动接口 42](#_Toc439244778)

[3.7.1获取冷启动一级分类 42](#_Toc439244779)

[3.7.2获取冷启动二级分类 43](#_Toc439244780)

[3.7.3根据冷启动一级分类和冷启动二级分类获取相应类别下的冷启动标签列表 43](#_Toc439244781)

[3.7.4提交用户感兴趣的冷启动标签 44](#_Toc439244782)

[3.7.5获取用户提交的冷启动标签详情 45](#_Toc439244783)

[3.8主播内容 46](#_Toc439244784)

[3.8.1 获取喜马拉雅主播分类 46](#_Toc439244785)

[3.8.2获取某个分类下的主播列表 46](#_Toc439244786)

[3.9错误处理 47](#_Toc439244787)

[3.9.1 HTTP 响应码 47](#_Toc439244788)

[3.9.2错误码 48](#_Toc439244789)

[4播放器说明 49](#_Toc439244790)

[4.1播放器概述 49](#_Toc439244791)

[4.2 播放器环境搭建 49](#_Toc439244792)

[4.3播放器状态介绍 49](#_Toc439244793)

[4.4播放器初始化和释放 50](#_Toc439244794)

[4.5通过播放器播放声音 51](#_Toc439244795)

[4.6播放器控制 51](#_Toc439244796)

[4.7播放器回调 53](#_Toc439244797)

[4.8播放器通知栏使用 54](#_Toc439244798)

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 修改时间 | 内容 | 修改人 |
| 0.1.0.0 | 2015-4-24 | 获取点播和直播内容接口 | Jack.qin |
| 0.1.1.0 | 2015-5-22 | 根据服务端API改动修改SDK接口 | Jack.qin |
| 0.1.2.0 | 2015-6-23 | SDK接口修改 | Jack.qin |
| 0.1.3.0 | 2015-6-30 | SDK播放器模块 | ChadWii |
| 0.1.4.0 | 2015-07-06 | 文档格式修正 | Jack.qin |
| 1.0.0.1 | 2015-08-31 | SDK架构优化 | Jack.qin |
| 1.0.0.2 | 2015-09-02 | SDK接口修改 | ChadWii |
| 1.0.0.3 | 2015-09-08 | SDK Bug修复 | ChadWii |
| 1.0.0.4 | 2015-09-15 | SDK增加联网配置 | ChadWii |
| 1.0.0.5 | 2015-10-21 | SDK 增加批量获取电台列表接口 | Jack.qin |
| 1.0.0.6 | 2015-11-06 | SDK 增加猜你喜欢接口 | Jack.qin |
| 1.0.0.7 | 2015-11-18 | SDK增加主播的专辑列表 | Jack.qin |
| 1.0.1.0 | 2015-11-27 | otp替换为access\_token | Jack.qin |
| 1.0.1.1 | 2015-12-09 | 增加冷启动接口；修改3.2.20的返回值 | Jack.qin |
| 1.0.1.2 | 2015-12-17 | 增加最新专辑列表接口；增加主播分类接口；增加主播分类下主播列表接口 | Jack.qin |
| 1.0.1.3 | 2015-12-28 | 增加获取某个城市下的电台列表 | Jack.qin |
| 1.0.1.4 | 2016-01-19 | 优化播放数据统计；在 3.2.4的返回数据中增加album\_intro字段 | Jack.qin |
| 1.0.1.5 | 2016-01-21 | 增加控制音量接口 | Jack.qin |
| 1.0.1.6 | 2016-01-22 | 修改冷启动中参数错误 | Jack.qin |

# 为了便于喜马拉雅对播放数据进行统计，接入SDK时必须接入播放器逻辑，否则喜马拉雅有权直接封停接口。

# 1概述

本文档面向Android开发者。

本文档用于指导开发者快速接入喜马拉雅SDK，本SDK为Android应用提供获取点播、直播内容等功能。

# 2接入流程

## 2.1申请PACKID、APPKEY和APPSECRET

PACKID客户端包名，对Android客户端是包名

APPKEY开放平台应用唯一Key (应用私钥，请务必加密存储，切勿泄露)

APPSECRET开放平台应用密钥，请妥善保存

## 2.2 SDK依赖的第三方jar包

SDK在实现的过程中，使用了某些第三方的jar包，当第三方App集成SDK时，需要同时将以下jar包拷贝到项目的libs目录下，否则SDK不能正常运行。

1. gson.jar//SDK在解析请求返回的数据是用到
2. litpal.jar//SDK在数据持久化时用到。修复了原生jar不能持久化父类属性的bug，因此不建议使用原生jar
3. okhttp.jar// SDK联网框架使用okhttp
4. okhttp-urlconnection.jar// SDK联网框架使用okhttp
5. okio.jar// SDK联网框架使用okhttp

## 2.2配置应用工程的AndroidManifest.xml

### 2.2.1添加权限

<uses-permission android:name=*"android.permission.INTERNET"* />

<uses-permissionandroid:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="android.permission.CHANGE\_NETWORK\_STATE" />

### 2.2.2在AndroidManifest.xml中添加meta-data

注意：必须放在<application>元素区块内：

<meta-data

android:name=*"app\_key"*

android:value=*"b617866c20482d133d5de66fceb37da3"*/>

<meta-data

android:name=*"pack\_id"*

android:value=*"com.app.test.android"*/>

# 3编码接入详细介绍

## 3.1初始化接口（必接）

### 3.1.1设置APP秘钥

**接口签名：**

**publicvoid**init(Context context, String appsecret)

**接口说明：**

用于初始化XiMaLaYa SDK，在应用Activity的onCreate()函数中必须调用一次该方法，否则XiMaLaYa SDK处于未初始化状态，无法使用其他接口，建议位于application中调用。

**参数说明：**

context 上下文对象

appsecret开放平台应用密钥

**示例代码：**

CommonRequest.*getInstanse*().init(context, mAppSecret);

### 3.1.2 设置网络配置

**接口签名：**

**public void** setHttpConfig(Config config)

**接口说明：**

用于设置XiMaLaYa SDK联网配置。当不需要代理是，直接设置null。

**参数说明：**

ConfigSDK联网配置，Config的所有成员变量如下：

|  |  |  |
| --- | --- | --- |
| 变量名 | 变量类型 | 变量说明 |
| useProxy | Boolean | 使用代理开关 |
| proxyHost | String | 代理服务器ip地址 |
| proxyPort | Int | 代理服务器端口 |
| authorization | String | 代理服务器授权信息 |
| useCache | Boolean | 使用缓存开关 |
| connectionTimeOut | Int | 连接超时 |
| readTimeOut | Int | 读取超时 |
| property | Map<String, String> | 请求头 |

**示例代码：**

Config config = **new**Config();

config.useProxy = **true**; //若想使用代理，必须配置此项为true，否则代理配置被忽略

config.proxyHost = "192.168.3.1";

config.proxyPort = 1080;

config.connectionTimeOut = 3000;

config.readTimeOut = 3000;

CommonRequest.*getInstanse*().setHttpConfig(config);

## 3.2获取点播内容

### 3.2.1获取喜马拉雅内容分类

**接口签名：**

**publicstaticvoid**getCategories(**final** Map<String, String>specificParams,**final**IDataCallBack<CategoryList> callback)

**接口说明：**

此接口用于获取点播资源的内容分类，如：“有声小说”、“相声评书”等。

**参数说明：**

1. specificParams:预留参数，目前不需要传。
2. callback:回调接口，成功则返回CategoryList对象，失败则返回错误原因。

**示例代码：**

Map<String, String> map = **new**HashMap<String, String>();

CommonRequest.*getCategories*(map, **new**IDataCallBack<CategoryList>()

{

@Override

**publicvoid**onSuccess(CategoryList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.2.2获取专辑标签或者声音标签

**接口签名：**

**publicstaticvoid**getTags(Map<String, String>specificParams,

**final**IDataCallBack<TagList> callback)

**接口说明：**

此接口用于返回某个内容分类中的标签，如“相声评书”下的“郭德纲”、“赵本山”等。

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| category\_id | Int | 是 | 分类ID |
| type | Int | 否 | 指定查询的是专辑还是声音，0-专辑，1-声音，默认是0-专辑 |

参考代码

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.*CATEGORY\_ID*, categoryId);

map.put(DTransferConstants.*TYPE*, 0);

1. callback:回调接口，成功则返回TagList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getTags*(map, **new**IDataCallBack<TagList>()

{

@Override

**publicvoid**onSuccess(TagList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.2.3根据分类和标签获取某个分类某个标签下的热门专辑列表

**接口签名：**

**publicstaticvoid**getAlbums(Map<String, String>specificParams,

**final**IDataCallBack<AlbumList> callback)

**接口说明：**

接口用于返回热门专辑列表

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| category\_id | Int | 是 | 分类ID，指定分类 |
| tag\_name | String | 否 | 分类下对应**专辑标签**，不填则为热门分类 |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |

参考代码

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.*CATEGORY\_ID*, mCategoryId);

map.put(DTransferConstants.*TAG\_NAME*, mTagName);

map.put(DTransferConstants.*PAGE*, mPageNum);

注意：对于返回的每页的条数，可以通过如下接口设置

**publicvoid**setDefaultPagesize(**int** size)

1. callback:回调接口，成功则返回AlbumList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getAlbums*(map, **new**IDataCallBack<AlbumList>()

{

@Override

**publicvoid**onSuccess(AlbumList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.2.4根据专辑ID获取专辑下的声音列表，即专辑浏览

**接口签名：**

**publicstaticvoid**getTracks(Map<String, String>specificParams,

**final**IDataCallBack<TrackList> callback)

**接口说明：**

此接口用于返回声音列表

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| album\_id | Int | 是 | 专辑ID |
| sort | String | 否 | asc-正序或desc-倒序，默认为asc-正序 |
| page | Int | 否 | 当前第几页，不填默认为1 |

参考代码

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.*ALBUM\_ID*, mAlbumId);

map.put(DTransferConstants.*SORT*, "asc");

map.put(DTransferConstants.*PAGE*, mPage);

注意：对于返回的每页的条数，可以通过如下接口设置

**publicvoid**setDefaultPagesize(**int** size)

1. callback:回调接口，成功则返回TrackList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getTracks*(map, **new**IDataCallBack<TrackList>()

{

@Override

**publicvoid**onSuccess(TrackList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}});

### 3.2.5批量获取专辑列表

**接口签名：**

**publicstaticvoid**getBatch(Map<String, String>specificParams,

**final**IDataCallBack<BatchAlbumList> callback)

**接口说明：**

此接口用于批量返回专辑列表

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| ids | String | 是 | 专辑ID列表 |

参考代码

Stringalbum\_ids = "albumid1,albumid2,albumid3…";

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.*ALBUM\_IDS*, album\_ids);

1. callback:回调接口，成功则返回BatchAlbumList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getBatch*(map, **new**IDataCallBack<BatchAlbumList>()

{

@Override

**publicvoid**onSuccess(BatchAlbumList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.2.6根据分类和标签获取某个分类某个标签下的热门声音列表

**接口签名：**

**publicvoid**getHotTracks(**final** Map<String, Object>specificParams,

**final**IDataCallBack<TrackHotList> callback)

**接口说明：**

此接口用于返回热门声音列表

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| category\_id | Int | 是 | 分类ID，指定分类 |
| tag\_name | String | 否 | 分类下对应**声音标签**，不填则为热门分类 |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |

参考代码

Map<String, String>map = **new**HashMap<String, String>();

map.put(DTransferConstants.*CATEGORY\_ID*, mCategoryId);

map.put(DTransferConstants.*TAG\_NAME*, mTagName);

map.put(DTransferConstants.*PAGE*, mPageNum);

注意：对于返回的每页的条数，可以通过如下接口设置

**publicvoid**setDefaultPagesize(**int** size)

1. callback:回调接口，成功则返回TrackHotList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getHotTracks*(map, **new**IDataCallBack<TrackHotList>()

{

@Override

**publicvoid**onSuccess(TrackHotList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.2.7批量获取声音

**接口签名：**

**publicstaticvoid**getBatchTracks(Map<String,String>specificParams,**final**IDataCallBack<BatchTrackList> callback)

**接口说明：**

此接口用于批量获取声音

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| ids | string | 是 | 声音ID列表 |

参考代码

Stringtrack\_ids = "替换成用英文逗号分隔的声音ID";

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.*TRACK\_IDS*, track\_ids);

1. callback:回调接口，成功则返回BatchTrackList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getBatchTracks*(map, **new**IDataCallBack<BatchTrackList>()

{

@Override

**publicvoid**onSuccess(BatchTrackList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.2.8搜索专辑

**接口签名：**

**publicstaticvoid**getSearchedAlbums(Map<String, String>specificParams,**final**IDataCallBack<SearchAlbumList> callback)

**接口说明：**

用于搜索专辑

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| q | String | 是 | 搜索关键词 |
| category\_id | Int | 否 | 分类ID，不填或者为0检索全库 |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |

参考代码

String q = "因为爱情来之不易";

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.*SEARCH\_KEY*, q);

map.put(DTransferConstants.*CATEGORY\_ID*, mCategoryId);

map.put(DTransferConstants.*PAGE*, mPageNum);

注意：对于返回的每页的条数，可以通过如下接口设置

**publicvoid**setDefaultPagesize(**int** size)

1. callback:回调接口，成功则返回SearchAlbumList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getSearchedAlbums*(map, **new**IDataCallBack<SearchAlbumList>()

{

@Override

**publicvoid**onSuccess(SearchAlbumList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.2.9搜索声音

**接口签名：**

**publicstaticvoid**getSearchedTracks(Map<String, String>specificParams,**final**IDataCallBack<SearchTrackList> callback)

**接口说明：**

此接口用于搜索声音

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| q | String | 是 | 搜索关键词 |
| category\_id | Int | 否 | 分类ID，不填或者为0检索全库 |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |

参考代码

String q = "因为爱情来之不易";

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.*SEARCH\_KEY*, q);

map.put(DTransferConstants.*CATEGORY\_ID*, mCategoryId);

map.put(DTransferConstants.*PAGE*, mPageNum);

注意：对于返回的每页的条数，可以通过如下接口设置

**publicvoid**setDefaultPagesize(**int** size)

1. callback:回调接口，成功则返回SearchAlbumList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getSearchedTracks*(map, **new**IDataCallBack<SearchTrackList>()

{

@Override

**publicvoid**onSuccess(SearchTrackList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.2.10获取运营人员在首页推荐的专辑分类

**接口签名：**

**publicstaticvoid**getRecommendCategory(Map<String, String> specificParams,**final** IDataCallBack<HumanRecommendCategoryList> callback)

**接口说明：**

此接口用于获取运营人员在首页推荐的专辑分类

**参数说明：**

1. specificParams: 字段介绍如下

无参数

1. callback:回调接口，成功则返回HumanRecommendCategoryList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getRecommendCategory*(map, **new** IDataCallBack<HumanRecommendCategoryList>()

{

@Override

**publicvoid** onSuccess(HumanRecommendCategoryList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

### 3.2.11获取全量专辑数据

**接口签名：**

**publicstaticvoid** getAllAlbums(Map<String, String> specificParams,**final** IDataCallBack<AlbumList> callback)

**接口说明：**

此接口用于获取全量专辑数据

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| category\_id | Int | 是 | 分类ID，指定分类，为0时表示热门分类 |
| tag\_name | String | 否 | 标签名 |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |
| count | Int | 否 | 每页多少条，默认20，最多不超过100 |

1. callback:回调接口，成功则返回AlbumList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getAlbums*(map, **new** IDataCallBack<AlbumList>()

{

@Override

**publicvoid** onSuccess(AlbumList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

### 3.2.12根据专辑ID列表获取批量专辑更新提醒信息列表

**接口签名：**

**publicstaticvoid** getUpdateBatch(Map<String, String> specificParams,**final** IDataCallBack<UpdateBatchList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| ids | string | 是 | 专辑ID列表 |

参考代码

Stringalbum\_ids = "albumid1,albumid2,albumid3…";

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.*ALBUM\_IDS*, album\_ids);

1. callback:回调接口，成功则返回UpdateBatchList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getUpdateBatch*(map, **new** IDataCallBack<UpdateBatchList>()

{

@Override

**publicvoid** onSuccess(UpdateBatchList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

### 3.2.13获取某个专辑的相关推荐

**接口签名：**

**publicstaticvoid**getRelativeAlbums(Map<String, String>specificParams,**final** IDataCallBack<RelativeAlbums> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| albumId | Long | 是 | 要搜索获得相关推荐的专辑id |

1. callback:回调接口，成功则返回RelativeAlbums对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getRelativeAlbums*(map, **new** IDataCallBack<RelativeAlbums>()

{

@Override

**publicvoid** onSuccess(RelativeAlbums object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

### 3.2.14获取下载听模块的推荐下载专辑

**接口签名：**

**publicstaticvoid** getRecommendDownloadList(Map<String, String> specificParams,**final** IDataCallBack<RecommendDownload> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| calc\_dimension | Int | 是 | 计算维度，现支持经典（0），最火（1），最新（2），播放最多（3） |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |

1. callback:回调接口，成功则返回RelativeAlbums对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getRecommendDownloadList*(map, **new** IDataCallBack<RecommendDownload>()

{

@Override

**publicvoid** onSuccess(RecommendDownload object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

### 3.2.15根据上一次所听声音的id，获取此声音所在那一页的声音

**接口签名：**

**publicstaticvoid** getLastPlayTracks(Map<String, String>specificParams,**final** IDataCallBack<LastPlayTrackList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| album\_id | Long | 是 | 专辑id |
| track\_id | Long | 是 | 声音id |

1. callback:回调接口，成功则返回LastPlayTrackList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getLastPlayTracks*(map, **new**IDataCallBack<LastPlayTrackList>()

{

@Override

**publicvoid** onSuccess(LastPlayTrackList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

### 3.2.16获取最新热搜词

**接口签名：**

**publicstaticvoid** getHotWords(Map<String, String> specificParams,**final** IDataCallBack<HotWordList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| top | int | 是 | 获取前top长度的热搜词。（1<=top<=20：目前top只支持最多20个） |

1. callback:回调接口，成功则返回HotWordList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getHotWords*(map, **new**IDataCallBack<HotWordList>()

{

@Override

**publicvoid** onSuccess(HotWordList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

### 3.2.17获取某个关键词的联想词

**接口签名：**

**publicstaticvoid** getSuggestWord(Map<String, String> specificParams,**final** IDataCallBack<SuggestWords> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| q | String | 是 | 搜索查询词参数 |

1. callback:回调接口，成功则返回SuggestWords对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getSuggestWord*(map, **new**IDataCallBack<SuggestWords>()

{

@Override

**publicvoid** onSuccess(SuggestWords object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

### 3.2.18获取某个声音的相关推荐专辑

**接口签名：**

**publicstaticvoid**getRelativeAlbumsUseTrackId(Map<String, String> specificParams,**final** IDataCallBack<RelativeAlbums> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| trackId | Long | 是 | 要获得相关推荐的声音id |

1. callback:回调接口，成功则返回RelativeAlbums对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getRelativeAlbumsUseTrackId*(map, **new**IDataCallBack<RelativeAlbums>()

{

@Override

**publicvoid** onSuccess(RelativeAlbums object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

### 3.2.19获取猜你喜欢的专辑

**接口签名：**

**public static void** getGuessLikeAlbum(Map<String, String> specificParams,

**final** IDataCallBack<GussLikeAlbumList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| device\_id | String | 是 | 设备唯一标识 |
| like\_count | Int | 否 | 返回几条，不填则默认为3，取值区间为[1, 50] |

1. callback:回调接口，成功则返回GussLikeAlbumList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getGuessLikeAlbum*(specificParams, **new** IDataCallBack<GussLikeAlbumList>()

{

@Override

**publicvoid** onSuccess(GussLikeAlbumList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

### 3.2.20获取主播的专辑列表

**接口签名：**

**publicstaticvoid** getAlbumsByAnnouncer(Map<String, String> specificParams,

**final** IDataCallBack<BatchAlbumList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| aid | Int | 是 | 主播用户ID |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |
| count | Int | 否 | 每页多少条，默认20，最多不超过100 |

1. callback:回调接口，成功则返回BatchAlbumList对象，失败则返回错误原因

### 3.2.21根据分类和标签获取某个分类某个标签下的最新专辑列表

**接口签名：**

**public** **static** **void** getUpToDateAlbums(Map<String, String> specificParams,

**final** IDataCallBack<AlbumList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| category\_id | Int | 是 | 分类ID，指定分类，为0时表示热门分类 |
| tag\_name | String | 否 | 分类下对应的专辑标签，不填则为热门分类 |
| calc\_dimension | Int | 是 | 计算维度，现支持最火（1），最新（2），经典或播放最多（3） |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |
| count | Int | 否 | 每页多少条，默认20，最多不超过200 |

1. callback:回调接口，成功则返回AlbumList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getUpToDateAlbums*(map, **new** IDataCallBack<AlbumList>() {

@Override

**public** **void** onSuccess(AlbumList object) {

}

@Override

**public** **void** onError(**int** code, String message) {

}

});

### 3.2.22获取喜马拉雅提供的由合作方自定义图标的内容分类

**接口签名：**

**public** **static** **void** getCustomizedCategories(

**final** Map<String, String> specificParams,

**final** IDataCallBack<CategoryList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| icon\_set\_id | Int | 是 | 定制化的图标集ID |

1. callback:回调接口，成功则返回CategoryList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getCustomizedCategories*(map, **new** IDataCallBack<CategoryList>() {

@Override

**public** **void** onSuccess(CategoryList object) {

}

@Override

**public** **void** onError(**int** code, String message) {

}

});

### 3.2.23获取合作方定制化的声音列表

**接口签名：**

**public** **static** **void** getCustomizeTrackList(

Map<String, String> specificParams,

**final** IDataCallBack<TrackList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| customized\_tracklist\_id | Int | 是 | 定制化的声音集ID |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |
| count | Int | 否 | 每页多少条，默认20，最多不超过200 |

1. callback:回调接口，成功则返回TrackList对象，失败则返回错误原因

**示例代码：**

## 3.3获取直播内容

### 3.3.1获取直播省市列表

**接口签名：**

**publicstaticvoid**getProvinces(Map<String, String>specificParams,

**final**IDataCallBack<ProvinceList> callback)

**接口说明：**

此接口用于获取直播省市列表

**参数说明：**

1. specificParams: 字段介绍如下

无参数

1. callback:回调接口，成功则返回ProvinceList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getProvinces*(map, **new**IDataCallBack<ProvinceList>()

{

@Override

**publicvoid**onSuccess(ProvinceList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.3.2获取直播电台

**接口签名：**

**publicstaticvoid**getRadios(Map<String, String>specificParams,

**final**IDataCallBack<RadioList> callback)

**接口说明：**

此接口用于返回直播电台列表

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| radio\_type | Int | 是 | 电台类型：1-国家台，2-省市台，3-网络台 |
| province\_code | Int | 否 | 省份代码，radio\_type为2时不能为空 |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |

参考代码

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.*RADIOTYPE*, 2);

map.put(DTransferConstants.*PROVINCECODE*, 310000);

map.put(DTransferConstants.*PAGE*, mPageNum);

注意：对于返回的每页的条数，可以通过如下接口设置

**publicvoid**setDefaultPagesize(**int** size)

1. callback:回调接口，成功则返回RadioList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getRadios*(map, **new**IDataCallBack<RadioList>()

{

@Override

**publicvoid**onSuccess(RadioList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.3.3获取直播电台某一天的节目排期表

**接口签名：**

**publicstaticvoid**getSchedules(Map<String, String>specificParams,

**final**IDataCallBack<ScheduleList> callback)

**接口说明：**

此接口用于返回直播电台某一天的节目排期表

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| radio\_id | Int | 是 | 直播电台ID |
| weekday | Int | 否 | 表示星期几，不填则取今天的星期。0-星期天，1-星期一，2-星期二，3-星期三，4-星期四，5-星期五，6-星期六 |

参考代码

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.*RADIOID*, 61);

map.put(DTransferConstants.*WEEKDAY*, Constants.WeekDay.*MONDAY*);

1. callback:回调接口，成功则返回ScheduleList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getSchedules*(map, **new**IDataCallBack<ScheduleList>()

{

@Override

**publicvoid**onSuccess(ScheduleList object)

{

}

**@Override**

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.3.4获取直播节目详情

**接口签名：**

**publicstaticvoid**getProgram(Map<String, String>specificParams,

**final**IDataCallBack<ProgramList> callback)

**接口说明：**

此接口用于返回所选直播节目的详情

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| radio\_id | Int | 是 | 直播电台ID |

参考代码

Map<String, Object> map = **new**HashMap<String, Object>();

map.put(DTransferConstants.*RADIOID*, 61);

1. callback:回调接口，成功则返回ProgramList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getProgram*(map, **new**IDataCallBack<ProgramList>()

{

@Override

**publicvoid**onSuccess(ProgramList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.3.5搜索直播

**接口签名：**

**publicstaticvoid**getSearchedRadios(Map<String, String>specificParams,**final**IDataCallBack<RadioList> callback)

**接口说明：**

按照关键词搜索直播

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| page | Int | 否 | 分页请求参数，表示请求第几页，默认为1即第一页 |
| q | String | 否 | 搜索查询词参数 |

参考代码

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.SEARCH\_KEY, “xx”);

1. callback:回调接口，成功则返回RadioList对象，失败则返回错误原因。

**示例代码：**

CommonRequest.*getSearchedRadios*(map, **new**IDataCallBack<RadioList>()

{

@Override

**publicvoid**onSuccess(RadioList object)

{

}

@Override

**publicvoid**onError(**int** code, String message)

{

}

});

### 3.3.6批量获取电台接口

**接口签名：**

**publicstaticvoid** getRadiosByIds(Map<String, String> specificParams,

**final** IDataCallBack<RadioListById> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| ids | Int[] | 是 | 电台ID列表 |

参考代码

Stringradio\_ids = "radioid1,radioid2,radioid3…";

Map<String, String> map = **new**HashMap<String, String>();

map.put(DTransferConstants.*RADIO\_IDS*, radio\_ids);

1. callback:回调接口，成功则返回RadioListById对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getRadiosByIds*(map, **new** IDataCallBack<RadioListById >()

{

@Override

**publicvoid** onSuccess(RadioListById object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

### 3.3.7获取某个城市下的电台列表

**接口签名：**

**public** **static** **void** getRadiosByCity(Map<String, String> specificParams,

**final** IDataCallBack<RadioList> callback) {

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| city\_code | Int | 是 | 城市code (国家行政规划的城市代码) |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |
| count | Int | 否 | 每页多少条，默认20，最多不超过100 |

1. callback:回调接口，成功则返回RadioList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getRadiosByCity*(map, **new** IDataCallBack<RadioList>() {

@Override

**public** **void** onSuccess(RadioList object) {

}

@Override

**public** **void** onError(**int** code, String message) {

}

});

## 3.4获取榜单内容

## 3.4.1根据榜单类型获取榜单首页的榜单列表

**接口签名：**

**publicstaticvoid**getRankList(Map<String, String> specificParams,

**final** IDataCallBack<RankList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| rank\_type | Int | 是 | 榜单类型，1-节目榜单 |

1. callback:回调接口，成功则返回RankList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getRankList*(map, **new** IDataCallBack<RankList>()

{

@Override

**publicvoid** onSuccess(RankList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

## 3.4.2根据rank\_key获取某个榜单下的专辑列表

**接口签名：**

**publicstaticvoid** getRankAlbumList(Map<String, String> specificParams,**final** IDataCallBack<RankAlbumList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| rank\_key | String | 是 | 用于获取具体榜单内容的key，可以先通过/ranks/index\_list获得 |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |

rank\_key为通过3.4.1接口获取

1. callback:回调接口，成功则返回RankAlbumList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getRankAlbumList*(map, **new**IDataCallBack<RankAlbumList>()

{

@Override

**publicvoid** onSuccess(RankAlbumList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

## 3.4.3根据rank\_key获取某个榜单下的声音列表

**接口签名：**

**publicstaticvoid** getRankTrackList(Map<String, String> specificParams,**final** IDataCallBack<RankTrackList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| rank\_key | String | 是 | 用于获取具体榜单内容的key，可以先通过/ranks/index\_list获得 |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |

rank\_key为通过3.4.1接口获取

1. callback:回调接口，成功则返回RankTrackList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getRankTrackList*(map, **new** IDataCallBack<RankTrackList>()

{

@Override

**publicvoid** onSuccess(RankTrackList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

## 3.4.4获取直播电台排行榜

**接口签名：**

**publicstaticvoid** getRankRadios(Map<String, String>specificParams,**final** IDataCallBack<RadioList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| radio\_count | Int | 是 | 需要获取排行榜中的电台数目。 |

1. callback:回调接口，成功则返回RadioList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getRankRadios*(map, **new** IDataCallBack<RadioList>()

{

@Override

**publicvoid** onSuccess(RadioList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

## 3.5获取听单内容

## 3.5.1获取精品听单内容

**接口签名：**

**publicstaticvoid** getColumnList(Map<String, String> specificParams,**final** IDataCallBack<ColumnList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |

1. callback:回调接口，成功则返回ColumnList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getColumnList*(map, **new**IDataCallBack<ColumnList>()

{

@Override

**publicvoid** onSuccess(ColumnList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

## 3.5.1获取精品听单内容

**接口签名：**

**publicstaticvoid** getComlumnDetail(Map<String, String> specificParams,**final** IDataCallBack<ColumnDetail> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| id | Int | 是 | 听单ID |

1. callback:回调接口，成功则返回ColumnDetail对象，失败则返回错误原因

注意：听单会返回声音列表或者专辑列表，需要通过ColumnDetail中的”column\_content\_type”字段判断声音列表或者专辑列表，如下

**示例代码：**

CommonRequest.*getComlumnDetail*(map, **new** IDataCallBack<ColumnDetail>()

{

@Override

**publicvoid** onSuccess(ColumnDetail object)

{

/\*\* 专辑\*/

**if**(object.getColumnContentType() == 1)

{

ColumnDetailAlbum albumList = (ColumnDetailAlbum) object;

}

/\*\* 声音\*/

**elseif**(object.getColumnContentType() == 2)

{

ColumnDetailTrack trackList = (ColumnDetailTrack) object;

}

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

## 3.6获取焦点图内容

## 3.6.1获取榜单的焦点图列表

**接口签名：**

**publicstaticvoid** getRankBannerList(Map<String, String> specificParams,**final** IDataCallBack<RankBannerList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| channel | String | 是 | app的渠道号（对应渠道焦点图配置） |
| app\_version | String | 是 | app版本号 |
| image\_scale | Int | 是 | 控制焦点图图片大小参数，scale=2为iphone适配类型，scale=3为iphone6适配机型，对于Android一般设为2 |

1. callback:回调接口，成功则返回RankBannerList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getRankBannerList*(map, **new**IDataCallBack<RankBannerList>()

{

@Override

**publicvoid** onSuccess(RankBannerList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

## 3.6.2获取发现页推荐的焦点图列表

**接口签名：**

**publicstaticvoid** getDiscoveryBannerList(Map<String, String>

specificParams,**final** IDataCallBack<DiscoveryBannerList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| channel | String | 是 | app的渠道号（对应渠道焦点图配置） |
| app\_version | String | 是 | app版本号 |
| image\_scale | Int | 是 | 控制焦点图图片大小参数，scale=2为iphone适配类型，scale=3为iphone6适配机型，对于Android一般设为2 |

1. callback:回调接口，成功则返回DiscoveryBannerList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getDiscoveryBannerList*(map, **new** IDataCallBack<DiscoveryBannerList>()

{

@Override

**publicvoid** onSuccess(DiscoveryBannerList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

## 3.6.3获取分类推荐的焦点图列表

**接口签名：**

**publicstaticvoid** getCategoryBannerList(Map<String, String> specificParams,**final** IDataCallBack<CategoryBannerList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| channel | String | 是 | app的渠道号（对应渠道焦点图配置） |
| app\_version | String | 是 | app版本号 |
| image\_scale | Int | 是 | 控制焦点图图片大小参数，scale=2为iphone适配类型，scale=3为iphone6适配机型，对于Android一般设为2 |
| category\_id | Int | 是 | 分类ID |
| content\_type | String | 是 | 内容类型：暂时仅专辑(album) |

1. callback:回调接口，成功则返回CategoryBannerList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getCategoryBannerList*(map, **new**IDataCallBack<CategoryBannerList>()

{

@Override

**publicvoid** onSuccess(CategoryBannerList object)

{

}

@Override

**publicvoid** onError(**int** code, String message)

{

}

});

## 3.7冷启动接口

## 3.7.1获取冷启动一级分类

**接口签名：**

**public** **static** **void** getColdbootGenres(

**final** Map<String, String> specificParams,

**final** IDataCallBack<GenreList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

无

1. callback:回调接口，成功则返回GenreList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getColdbootGenres*(map, **new** IDataCallBack<GenreList>() {

@Override

**public** **void** onSuccess(GenreList object) {

}

@Override

**public** **void** onError(**int** code, String message) {

}

});

## 3.7.2获取冷启动二级分类

**接口签名：**

**public** **static** **void** getColdbootSubGenres(

**final** Map<String, String> specificParams,

**final** IDataCallBack<SubGenreList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

无

1. callback:回调接口，成功则返回SubGenreList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getColdbootSubGenres*(map, **new** IDataCallBack<SubGenreList>() {

@Override

**public** **void** onSuccess(SubGenreList object) {

}

@Override

**public** **void** onError(**int** code, String message) {

}

});

## 3.7.3根据冷启动一级分类和冷启动二级分类获取相应类别下的冷启动标签列表

**接口签名：**

**public** **static** **void** getColdBootTag(**final** Map<String, String> specificParams,

**final** IDataCallBack<ColdBootTag> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| coldboot\_genre | String | 是 | 冷启动分类 |
| coldboot\_sub\_genre | String | 是 | 冷启动二级分类 |

1. callback:回调接口，成功则返回ColdBootTag对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getColdBootTag*(map, **new** IDataCallBack<ColdBootTag>() {

@Override

**public** **void** onSuccess(ColdBootTag object) {

}

@Override

**public** **void** onError(**int** code, String message) {

}

});

## 3.7.4提交用户感兴趣的冷启动标签

**接口签名：**

**public** **static** **void** postColdBootTag(Map<String, String> specificParams,

**final** IDataCallBack<PostResponse> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| coldboot\_genre | String | 是 | 冷启动分类 |
| coldboot\_sub\_genre | String | 是 | 冷启动二级分类 |
| device\_type | Int | 是 | 1-ios，2-android，3-pc |
| device\_id | String | 是 | 设备唯一标识 |
| coldboot\_tags | JSON Array | 是 | 用户勾选的感兴趣的冷启动标签列表 |

1. callback:回调接口，当PostResponse中的response字段为”ok”时，则上传成功；如果为”error”或者在回调onError中返回时则为失败。

**示例代码：**

CommonRequest.*postColdBootTag*(map, **new** IDataCallBack<PostResponse>() {

@Override

**public** **void** onSuccess(PostResponse object) {

}

@Override

**public** **void** onError(**int** code, String message) {

}

});

## 3.7.5获取用户提交的冷启动标签详情

**接口签名：**

**public** **static** **void** getColdBootDetail(

**final** Map<String, String> specificParams,

**final** IDataCallBack<ColdBootDetail> callback)

**参数说明：**

1. specificParams: 字段介绍如下

无

1. callback:回调接口，成功则返回ColdBootDetail对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getColdBootDetail*(map, **new** IDataCallBack<ColdBootDetail>() {

@Override

**public** **void** onSuccess(ColdBootDetail object) {

}

@Override

**public** **void** onError(**int** code, String message) {

}

});

## 3.8主播内容

## 3.8.1 获取喜马拉雅主播分类

**接口签名：**

**public** **static** **void** getAnnouncerCategoryList(

Map<String, String> specificParams,

**final** IDataCallBack<AnnouncerCategoryList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

无

1. callback:回调接口，成功则返回AnnouncerCategoryList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getAnnouncerCategoryList*(map, **new** IDataCallBack<AnnouncerCategoryList>() {

@Override

**public** **void** onSuccess(AnnouncerCategoryList object) {

}

@Override

**public** **void** onError(**int** code, String message) {

}

});

## 3.8.2获取某个分类下的主播列表

**接口签名：**

**public** **static** **void** getAnnouncerList(Map<String, String> specificParams,

**final** IDataCallBack<AnnouncerList> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| vcategory\_id | Int | 是 | 主播分类ID |
| page | Int | 否 | 返回第几页，必须大于等于1，不填默认为1 |
| calc\_dimension | Int | 是 | 返回的主播列表排序规则，取值和含义如下：  1-最火2-最新3-粉丝最多 |
| count | Int | 否 | 每页多少条，默认20，最多不超过200 |

1. callback:回调接口，成功则返回AnnouncerList对象，失败则返回错误原因

**示例代码：**

CommonRequest.*getAnnouncerList*(map, **new** IDataCallBack<AnnouncerList>() {

@Override

**public** **void** onSuccess(AnnouncerList object) {

}

@Override

**public** **void** onError(**int** code, String message) {

}

});

## 3.9播放数据上传

## 3.9.1上传单条直播播放数据

**接口签名：**

**public** **static** **void** postLiveSingleRecord(

**final** Map<String, String> specificParams,

**final** IDataCallBack<PostResponse> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| radio\_id | Int | 是 | 电台ID |
| program\_schedule\_id | Int | 否 | 节目排期ID |
| program\_id | Int | 否 | 节目ID |
| duration | Double | 是 | 播放时长，单位秒。  即播放一个音频过程中，真正处于播放中状态的总时间。 |
| played\_sec | Double | 是 | 播放第几秒或最后播放到的位置，是相对于这个音频开始位置的一个值。  如果没有拖动播放条、快进、快退、暂停、单曲循环等操作，played\_secs的值一般和duration一致。 |
| started\_at | Long | 否 | 播放开始时刻，Unix毫秒数时间戳 |

1. callback: 回调接口，当PostResponse中的response字段为”ok”时，则上传成功；如果为”error”或者在回调onError中返回时则为失败，其中的code以及message字段分别表示http status code 以及message

## 3.9.1上传单条声音播放数据

**接口签名：**

**public** **static** **void** postTrackSingleRecord(

**final** Map<String, String> specificParams,

**final** IDataCallBack<PostResponse> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 必填 | 描述 |
| track\_id | Long | 是 | 声音ID |
| duration | double | 是 | 播放时长，单位秒。  即播放一个音频过程中，真正处于播放中状态的总时间。 |
| played\_secs | double | 是 | 播放第几秒或最后播放到的位置，是相对于这个音频开始位置的一个值。  如果没有拖动播放条、快进、快退、暂停、单曲循环等操作，played\_secs的值一般和duration一致。 |
| started\_at | Long | 否 | 播放开始时刻，Unix毫秒数时间戳 |
| play\_type | Int | 是 | 0：联网播放，1：断网播放 |

1. callback: 回调接口，当PostResponse中的response字段为”ok”时，则上传成功；如果为”error”或者在回调onError中返回时则为失败，其中的code以及message字段分别表示http status code 以及message

## 3.9.1批量上传直播播放数据

**（注意：每次上传的批量条数必须小于等于200条！）**

**接口签名：**

**public** **static** **void** postLiveBatchsRecords(

**final** Map<String, String> specificParams,

**final** IDataCallBack<PostResponse> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| live\_records | Array | 直播播放数据列表，每条播放数据包含下列字段：   * radio\_id：听单ID * program\_schedule\_id: 节目排期ID * program\_id：节目ID * duration：播放时长，单位秒。 * played\_sec：播放第几秒或最后播放到的位置，是相对于这个音频开始位置的一个值。 * started\_at：播放开始时刻，Unix毫秒数时间戳 |

1. callback: 回调接口，当PostResponse中的response字段为”ok”时，则上传成功；如果为”error”或者在回调onError中返回时则为失败，其中的code以及message字段分别表示http status code 以及message

## 3.9.1批量上传声音播放数据

**（注意：每次上传的批量条数必须小于等于200条！）**

**接口签名：**

**public** **static** **void** postTrackBatchsRecords(

**final** Map<String, String> specificParams,

**final** IDataCallBack<PostResponse> callback)

**参数说明：**

1. specificParams: 字段介绍如下

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| track\_records | Array | 直播播放数据列表，每条播放数据包含下列字段：   * track\_id：声音ID * duration: 播放时长，单位秒。 * played\_sec：播放第几秒或最后播放到的位置，是相对于这个音频开始位置的一个值。 * started\_at：播放开始时刻，Unix毫秒数时间戳 * play\_type：0-联网播放，1-断网播放 |

1. callback: 回调接口，当PostResponse中的response字段为”ok”时，则上传成功；如果为”error”或者在回调onError中返回时则为失败，其中的code以及message字段分别表示http status code 以及message

## 3.10错误处理

## 3.10.1 HTTP 响应码

|  |  |
| --- | --- |
| 响应码 | 描述 |
| 200 | 请求成功 |
| 201 | 创建成功 |
| 202 | 更新成功 |
| 301 | 请求永久重定向 |
| 302 | 请求临时重定向 |
| 304 | 未更改 |
| 400 | 请求地址不存在或者包含不支持的参数 |
| 401 | 未授权，因为未成功进行身份认证 |
| 403 | 被禁止访问，因为没有访问特定资源的权限 |
| 404 | 请求的资源不存在 |
| 422 | 服务器理解请求，但其中包含非法参数（如包含XSS攻击风险参数） |
| 429 | 请求频率超配 |
| 500 | 服务器内部错误 |

## 3.10.2错误码

|  |  |  |
| --- | --- | --- |
| error\_no | error\_code | 说明 |
| 100 | ximalaya.common.request\_param\_check\_failed | 参数校验失败错误，包括参数缺失、参数格式错误等情况 |
| 101 | ximalaya.common.signature\_check\_failed | 签名验证失败 |
| 102 | ximalaya.common.permission\_validate\_failed | 权限验证失败 |
| 103 | ximalaya.common.app\_validate\_failed | app\_key无效或者应用处于不可用状态或者应用的client\_os\_type、pack\_id等参数不匹配 |
| 300 | ximalaya.platformcoorp.otp\_invalid\_or\_expired | 动态口令参数otp不正确或者已过期 |
| 500 | ximalaya.system.internal\_service\_error | 调用开放平台内部服务发生错误 |
| 501 | ximalaya.system.thirdparty\_service\_error | 调用第三方服务发生错误 |
| 502 | ximalaya.system.unknown\_server\_error | 未知服务器错误 |

# 4播放器说明

本部分介绍喜马拉雅SDK中附带播放器的使用说明。

## 4.1播放器概述

由于Android系统自带的媒体播放器不同版本差异较大，且后期我们会对喜马拉雅平台的音频内容加密，所以如果希望正常收听喜马拉雅平台的音频内容，需要定制版播放器再能正常播放。目前播放器支持点播、电台直播、电台回听。

## 4.2 播放器环境搭建

为了能正常使用播放器，首先需要开发者将SDK包中的两个so包拷贝到您的项目libs/armeabi目录下，并将xmopensdk.jar拷贝到libs目录下(如果你的项目中用到armeabi-v7等，请同样拷贝一份到armeabi-v7目录下)。

同时，播放器还依赖以下第三方jar：

android\_support\_v4.jar //使用了某些兼容性组件，必须加入项目的libs目录下

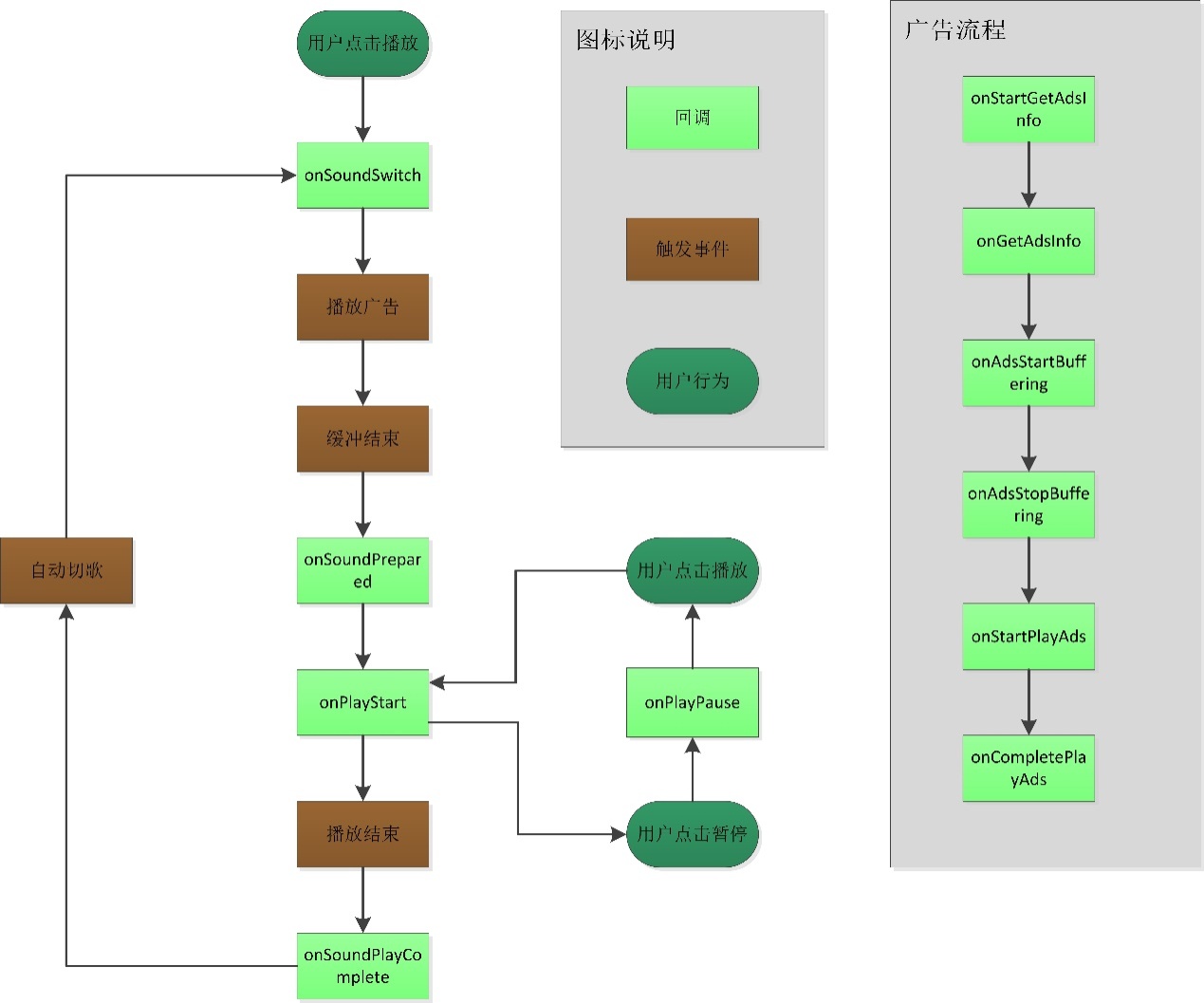
其次，需要在项目Manifest文件中声明以下权限（监听电话状态，当有来电或去电时，自动暂停，当电话结束后，自动恢复播放）：

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

<uses-permission android:name="android.permission.PROCESS\_OUTGOING\_CALLS" />

## 4.3播放器状态介绍

为了大家方便接入喜马拉雅播放器，喜马拉雅播放器的各状态尽量保持和Android系统播放器状态一致。状态图如下：



## 4.4播放器初始化和释放

喜马拉雅播放器服务采用AIDL机制，独立运行在一个进程中，我们在此基础上又做了一层封装，第三方App只需要和XmPlayerManager交互，XmPlayerManager再负责和播放器服务交互。

首先需要在App的AndroidManifest.xml文件中注册播放器服务

<serviceandroid:name=*"com.ximalaya.ting.android.opensdk.player.service.XmPlayerService"*

android:process=*":player"*/>

建议将播放Service配置成RemoteService，这样可以减少app的内存峰值，避免OOM。

其次，需要在App初始化的时候初始化XmPlayerManager，XmPlayerManager会去启动播放器服务，并初始化Service。XmPlayerManager有两个初始化方法：

|  |  |
| --- | --- |
| init() | 初始化方法 |
| init(int id, Notification notification) | 此方法初始化在播放器服务启动后会在通知栏显示指定的Notification。参数同NotificationManager.notify(int id, Notification notification)。 |

注：init()方法只需要调用一次，多次调用仅第一次有效。

当程序不在使用播放器时，应及时释放播放器，让系统回收资源。释放播放器通过调用XmPlayerManager.release()来实现。release()和init()应成对出现。

## 4.5通过播放器播放声音

|  |  |
| --- | --- |
| playList(List<Track> list, intstartIndex) | 此方法用来播放普通声音列表  list :声音列表  startIndex：播放index指定处的声音 |
| playList(CommonTrackList list, intstartIndex) | 此方法播放由XiMaLaYa.getTracks()、getHotTracks()、getBatchTracks()、getSearchedTracks()直接返回的TrackList。使用此方法播放的列表播放器可自动加载下一页 |
| playRadio(Radio radio) | 此方法用来播放电台 |
| playSchedule(List<Schedule> schedules, intstartIndex) | 此方法用来播放电台排期表,当startindex为-1时播放列表中的直播（非回听）；当startindex大于0且小于当前直播index时，则播放对应位置的回听；需要注意如果获取的不是当天的Schedule，需要在start\_time和end\_time中加上时间天，如  19:09:01，表示19日9点1分。 |

## 4.6播放器控制

XmPlayerManager提供了一系列方法用于控制播放器的状态，主要有：开始播放，暂停播放，上一首，下一首，拖动等。

|  |  |
| --- | --- |
| play() | 尝试恢复播放器的播放状态，如果恢复失败，则播放列表的第一条声音 |
| play(int index) | 播放指定索引处的声音 |
| pause() | 暂停播放器 |
| stop() | 停止播放器 |
| playPre() | 上一首 |
| playNext() | 下一首 |
| setPlayMode(int mode) | 设置播放器模式，mode取值为PlayerConstants中的下列之一：  PLAY\_MODEL\_SINGLE单曲播放  PLAY\_MODEL\_SINGLE\_LOOP 单曲循环播放  PLAY\_MODEL\_LIST列表播放  PLAY\_MODEL\_LIST\_LOOP列表循环  PLAY\_MODEL\_RANDOM 随机播放 |
| getPlayerStatus() | 获取播放器状态，返回值是PlayerConstants中的下列之一：  STATE\_IDLE播放器IDLE状态  STATE\_INITIALIZED 播放器已经初始化了  STATE\_PREPARING播放器准备中  STATE\_PREPARED播放器准备完成  STATE\_STARTED 播放器开始播放  STATE\_STOPPED播放器停止  STATE\_PAUSED 播放器暂停  STATE\_COMPLETED 播放器单曲播放完成  STATE\_ERROR播放器错误  STATE\_END播放器被释放 |
| isPlaying() | 播放器是否正在播放（注意，播放器在缓冲时，虽然没有声音输出，此方法仍然认为播放器正在播放） |
| hasPreSound() | 是否有上一首 |
| hasNextSound() | 是否有下一首 |
| getDuration() | 获取音频时长，同Mediaplayer.getDuration() |
| seekToByPercent(float percent) | 按音频时长的比例来拖动 |
| seekTo(intpos) | 拖动，同Mediaplayer.seekTo(); |
| setVolume(float leftVolume, float rightVolume) | 调节音量 |

## 4.7播放器回调

第三方app如果想监听播放器的状态变化，需要实例化IXmPlayerStatusListener，并通过XmPlayerManager.addPlayerStatusListener()注册一个监听器。

各回调方法解释如下：

|  |  |
| --- | --- |
| onPlayStart() | 开始播放 |
| onPlayPause() | 暂停播放 |
| onPlayStop() | 停止播放 |
| onSoundPlayComplete() | 播放完成 |
| onSoundPrepared() | 播放器准备完毕 |
| void onSoundSwitch(PlayableModel lastModel, PlayableModel curModel);) | 切歌  lastModel：上一首model 可能为空  curModel：下一首model  请通过model中的kind字段来判断是track、radio和schedule；  上一首的播放时间请通过lastPlayedMills字段来获取；和上一版不一样 |
| onBufferingStart() | 开始缓冲 |
| onBufferingStop() | 结束缓冲 |
| onBufferProgress(int percent) | 缓冲进度回调 |
| onPlayProgress(intcurrPos, int duration) | 播放进度回调 |
| onError(XmPlayerException exception) | 播放器错误 |

如果app还想监听广告播放状态，则需要实例化IXmAdsStatusListener，并通过XmPlayerManager.addAdsStatusListener()注册监听器，各方法说明如下：

|  |  |
| --- | --- |
| onStartGetAdsInfo() | 开始获取广告物料 |
| onGetAdsInfo(AdvertisList ads) | 获取广告物料成功 |
| onAdsStartBuffering() | 广告开始缓冲 |
| onAdsStopBuffering() | 广告结束缓冲 |
| onStartPlayAds(Advertis ad, int position) | 开始播放广告  ad : 当前播放广告  position : 当前播放的广告在广告列表中的索引 |
| onCompletePlayAds() | 广告播放完毕 |
| onError(int what, int extra) | 播放广告错误  what : 错误类型  extra : 错误的额外信息 |

## 4.8播放器通知栏使用

1. 特殊要求

提供文件名为”view\_notify\_play”和”view\_notify\_play\_big”的布局文件，如以下样例文件，每个view的名字和资源名不要修改。

XmPlayerManager.*getInstance*(**this**).init(notificationid,XmNotificationCreater.*getInstanse*(**this**).createNotification(**this**.getApplicationContext(), MainActivity.**class**));

1. 样例布局文件,分别代表”view\_notify\_play”和”view\_notify\_play\_big”

<?xmlversion=*"1.0"*encoding=*"utf-8"*?>

<RelativeLayoutxmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:background=*"#80000000"*>

<ImageView

android:id=*"@+id/img\_notifyIcon"*

android:layout\_width=*"@dimen/size\_110px"*

android:layout\_height=*"@dimen/size\_110px"*

android:layout\_alignParentLeft=*"true"*

android:layout\_centerVertical=*"true"*

android:layout\_marginLeft=*"@dimen/size\_space\_default"*

android:src=*"@drawable/ting"*/>

<ImageView

android:id=*"@+id/img\_notifyNext"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentRight=*"true"*

android:layout\_centerVertical=*"true"*

android:layout\_marginRight=*"@dimen/size\_space\_default"*

android:padding=*"@dimen/size\_space\_default"*

android:contentDescription=*"@string/next\_sound"*

android:src=*"@drawable/ic\_notify\_small\_next"*/>

<ImageView

android:id=*"@+id/img\_notifyPlayOrPause"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_centerVertical=*"true"*

android:layout\_marginRight=*"@dimen/size\_space\_default"*

android:padding=*"@dimen/size\_space\_default"*

android:layout\_toLeftOf=*"@id/img\_notifyNext"*

android:contentDescription=*"@string/start\_play"*

android:src=*"@drawable/ic\_notify\_small\_play"*/>

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:layout\_marginLeft=*"@dimen/size\_10px"*

android:layout\_toLeftOf=*"@id/img\_notifyPlayOrPause"*

android:layout\_toRightOf=*"@id/img\_notifyIcon"*

android:gravity=*"center"*

android:orientation=*"vertical"*>

<TextView

android:id=*"@+id/txt\_notifyMusicName"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:ellipsize=*"end"*

android:gravity=*"left"*

android:singleLine=*"true"*

android:text=*"@string/ximalaya\_slogan"*

android:textColor=*"#ffffff"*

android:textSize=*"@dimen/text\_size\_16"*/>

<TextView

android:id=*"@+id/txt\_notifyNickName"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_marginTop=*"@dimen/size\_10px"*

android:gravity=*"left"*

android:text=*"@string/ximalaya"*

android:textColor=*"#50ffffff"*

android:textSize=*"@dimen/text\_size\_15"*/>

</LinearLayout>

</RelativeLayout>

<?xmlversion=*"1.0"*encoding=*"utf-8"*?>

<RelativeLayoutxmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"110dp"*

android:background=*"#80000000"*>

<ImageView

android:id=*"@+id/img\_notifyIcon"*

android:layout\_width=*"110dp"*

android:layout\_height=*"110dp"*

android:layout\_alignParentLeft=*"true"*

android:layout\_centerVertical=*"true"*

android:scaleType=*"centerCrop"*

android:src=*"@drawable/ting"*/>

<ImageView

android:id=*"@+id/img\_notifyClose"*

android:layout\_width=*"35dp"*

android:layout\_height=*"25dp"*

android:layout\_alignParentRight=*"true"*

android:layout\_alignParentTop=*"true"*

android:layout\_marginRight=*"0dp"*

android:layout\_marginTop=*"4dp"*

android:contentDescription=*"关闭"*

android:paddingRight=*"4dp"*

android:scaleType=*"centerInside"*

android:src=*"@drawable/notify\_close"*/>

<TextView

android:id=*"@+id/txt\_notifyMusicName"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignParentTop=*"true"*

android:layout\_marginLeft=*"10dp"*

android:layout\_marginTop=*"15dp"*

android:layout\_toLeftOf=*"@+id/img\_notifyClose"*

android:layout\_toRightOf=*"@+id/img\_notifyIcon"*

android:ellipsize=*"end"*

android:gravity=*"left"*

android:singleLine=*"true"*

android:text=*"喜马拉雅听我想听"*

android:textColor=*"#ffffff"*

android:textSize=*"16sp"*/>

<TextView

android:id=*"@+id/txt\_notifyNickName"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/txt\_notifyMusicName"*

android:layout\_below=*"@+id/txt\_notifyMusicName"*

android:layout\_marginTop=*"3dp"*

android:gravity=*"left"*

android:text=*"喜马拉雅"*

android:textColor=*"#50ffffff"*

android:textSize=*"13sp"*/>

<RelativeLayout

android:id=*"@+id/control\_bar"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"50dp"*

android:layout\_alignLeft=*"@+id/txt\_notifyMusicName"*

android:layout\_alignParentBottom=*"true"*

android:layout\_below=*"@+id/txt\_notifyNickName"*

android:gravity=*"center\_vertical"*

android:paddingBottom=*"8dp"*

android:paddingTop=*"3dp"*>

<ImageView

android:id=*"@+id/img\_notifyPre"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_centerVertical=*"true"*

android:layout\_marginRight=*"25dp"*

android:layout\_toLeftOf=*"@id/img\_notifyPlayOrPause"*

android:contentDescription=*"上一首"*

android:scaleType=*"fitCenter"*

android:src=*"@drawable/ic\_notify\_pre"*/>

<ImageView

android:id=*"@+id/img\_notifyPlayOrPause"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_centerInParent=*"true"*

android:contentDescription=*"开始播放"*

android:scaleType=*"fitCenter"*

android:src=*"@drawable/ic\_notify\_play"*/>

<ImageView

android:id=*"@+id/img\_notifyNext"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_centerVertical=*"true"*

android:layout\_marginLeft=*"25dp"*

android:layout\_toRightOf=*"@id/img\_notifyPlayOrPause"*

android:contentDescription=*"下一首"*

android:scaleType=*"fitCenter"*

android:src=*"@drawable/ic\_notify\_next"*/>

</RelativeLayout>

</RelativeLayout>