蚂蚁集团
ANT GROUP

蚂蚁科技产品手册

定位

产品版本：V20210331
文档版本：V20210331
蚂蚁科技技术文档

蚂蚁科技集团有限公司版权所有 © 2020 ，并保留一切权利。

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

商标声明

蚂蚁集团
ANT GROUP

及其他蚂蚁科技服务相关的商标均为蚂蚁科技所有。
本文档涉及的第三方的注册商标，依法由权利人所有。

免责声明

由于产品版本升级、调整或其他原因，本文档内容有可能变更。蚂蚁科技保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在蚂蚁科技授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过蚂蚁科技授权渠道下载、获取最新版的用户文档。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

目录

1 定位简介.....1
2 接入 Android.....1
3 接入 iOS.....2
4 代码示例.....9

1 定位简介

mPaaS 提供了定位组件以方便使用定位相关服务。通过包装系统接口，该组件提供了如下功能：

- 获取当前设备所在位置的经纬度信息的简易方法。
- 获取经纬度信息的时间和精确度信息。
- 支持缓存和坐标系转换。
- Hook 所有系统 API 调用，统一定位流程。

说明：目前根据经纬度进行逆地理编码信息查询的功能暂不支持。

2 接入 Android

定位 SDK 是一套简单的 LBS（Location-based services）定位接口，您可以使用这套定位 API 获取定位结果。

定位支持 **原生 AAR 接入**、**mPaaS Inside 接入** 和 **组件化接入** 三种接入方式。

前置条件

- 若采用原生 AAR 方式接入，需先完成 将 mPaaS 添加到您的项目中 的前提条件和后续相关步骤。
- 若采用 mPaaS Inside 方式接入，需先完成 mPaaS Inside 接入流程 。
- 若采用组件化方式接入，需先完成 组件化接入流程 。

添加 SDK

原生 AAR 方式

参考 AAR 组件管理 ，通过 **组件管理（AAR）** 在工程中安装 **定位** 组件。

mPaaS Inside 方式

在工程中通过 **组件管理** 安装 **定位** 组件。
更多信息，参考 管理组件依赖 。

组件化方式

在 Portal 和 Bundle 工程中通过 **组件管理** 安装 **定位** 组件。
更多信息，参考 管理组件依赖 。

申请高德KEY

使用 LBS 之前，在 高德开放平台 申请账号并获取定位 Key。申请的 Key 示例如下：

配置 AndroidManifest

在 AndroidManifest.xml 文件中添加高德定位的 Key 和高德定位 Service。

```
<!--高德定位 Key-->
<meta-data
android:name="com.amap.api.v2.apikey"
android:value="填入您申请的高德 Key"/>
<!--高德定位 Service-->
<service android:name="com.amap.api.location.APSService"></service>
```

从 10.1.68.18 版本起，定位的自动签到功能默认关闭。如需开启自动签到，需要在 AndroidManifest 文件中手动添加如下配置：

```
<meta-data android:name="com.mpaas.lbs.autoCheckIn"android:value="true"/>
```

API 说明

- 调用定位能力

```
LBSLocationManagerProxy.getInstance().requestLocationUpdates(MainActivity.this, new
LBSLocationListener() {
@Override
public void onLocationUpdate(LBSLocation lbsLocation) {
Toast.makeText(MainActivity.this,"lbsLocation is" + lbsLocation.getAddress(), Toast.LENGTH_LONG).show();
}
@Override
public void onLocationFailed(int i) {
Toast.makeText(MainActivity.this,
"onLocationFailed"+ i, Toast.LENGTH_SHORT).show();
}
});
```

- 其他相关接口

```
// 注册定位
public void requestLocationUpdates(Context context, LBSLocationListener locationListener)
// remove 定位注册的回调
public void removeUpdates(Context context, LBSLocationListener listener)
// 获取最近一次定位成功的位置
public LBSLocation getLastKnownLocation(Context context)
```

3 接入 iOS

定位 SDK 是一套简单的 LBS (Location-based services) 定位接口，您可以使用这套定位 API 获取定位结果。

前置条件

您已经接入工程到 mPaaS。更多信息，请参见以下内容：

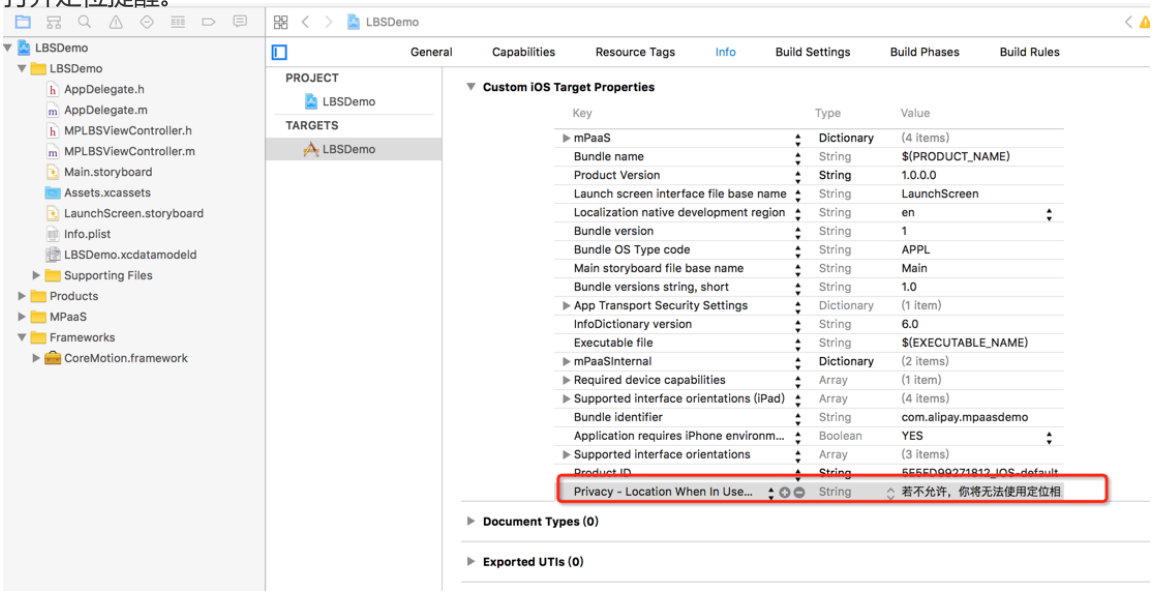
- 基于 mPaaS 框架接入
- 基于已有工程且使用 mPaaS 插件接入
- 基于已有工程且使用 CocoaPods 接入

添加 SDK

1. 根据您采用的接入方式，请选择相应的添加方式。
    - 使用 mPaaS Xcode Extension。
      此方式适用于采用了 **基于 mPaaS 框架接入** 或 **基于已有工程且使用 mPaaS 插件接入** 的接入方式。
        - 点击 Xcode 菜单项 **Editor** > **mPaaS** > **编辑工程**，打开编辑工程页面。
        - 选择 **移动定位**，保存后点击 **开始编辑**，即可完成添加。
    - 使用 cocoapods-mPaaS 插件。此方式适用于采用了 **基于已有工程且使用 CocoaPods 接入** 的接入方式。
        - 在 Podfile 文件中，使用 mPaaS_pod"mPaaS_LBS" 添加移动定位组件依赖。

- 在命令行中执行 pod install 即可完成接入。

2. 打开定位提醒。



使用 SDK

本文将结合 定位 官方 Demo 介绍如何在 10.1.32 及以上版本的基线中使用 定位 SDK。

目前，在 APMobileLBS 模块中，提供了获取当前位置的经纬度信息方法。

说明：定位服务目前暂不支持逆地理查询功能，您可调用高德接口进行逆地理查询。

API 说明

参见以下代码，了解定位服务相关接口，通过注释获取接口和相关参数说明。

使用 MPLBSConfiguration 配置参数

```
/**
```

```
定位服务的配置
*/
@interface MPLBSConfiguration : NSObject

/** 单次定位期望精度，单位米，建议结合业务场景传入一个可接受正数，如500，即500m以内的范围 */
@property (nonatomic, assign) CLLocationAccuracy desiredAccuracy;

/** 单次定位接受的缓存时间，从当前时间往前推，多长时间内的缓存是有效的，推荐设置30s以上的缓存时间 */
@property (nonatomic, assign) APCoreLocationCacheAvaliable cacheTimeInterval;

/** 单次定位或逆地理查询的超时时间，单位秒，默认和最小设置为2s */
@property (nonatomic, assign) NSTimeInterval timeOut;

/** 逆地理查询的信息级别，默认APCoreLocationReGeoLevelDistrict */
@property (nonatomic, assign) LBSLocationReGeoLevel reGeoLevel;

/** 逆地理查询的位置信息，在其中指定经纬度坐标 */
@property (nonatomic, strong) CLLocation *reGeoLocation;

/** 逆地理查询位置信息是否为高德坐标系，默认YES（使用 reGeoLocation 参数时生效） */
@property (nonatomic, assign) BOOL reGeoCoordinateConverted;

/** 是否打开签到功能，默认NO（按需设置打开） */
@property (nonatomic, assign) BOOL needCheckIn;

/**
* 是否需要高精度定位，iOS 14 以下不区分精度，iOS 14 及以上默认 NO (低精度)，需要业务指定是否需要高精度定位。
*/
@property (nonatomic,assign) BOOL highAccuracyRequired;

@end
```

使用 MPLBSLocationManager 发起定位请求

```
/**
定位结果的回调 block

@param success 是否成功
@param locationInfo 位置信息
@param error 定位失败的错误信息
*/
typedef void(^MPLBSLocationCompletionBlock)(BOOL success,
MPLBSLocationInfo *locationInfo,
NSError *error);

/**
定位服务
*/
@interface MPLBSLocationManager : NSObject

/**
初始化

@param configuration 参数配置
@return 实例
```

```
*/
- (instancetype)initWithConfiguration:(MPLBSConfiguration *)configuration;

/**
发起单次定位

@param needReGeocode 是否需要逆地理信息。由于定位服务目前暂不支持逆地理查询功能，此处需传入 NO。
@param block 定位结束的回调block
*/
- (void)requestLocationNeedReGeocode:(BOOL)needReGeocode
completionHandler:(MPLBSLocationCompletionBlock)block;
```

其中回调中 MPLBSLocationInfo 的说明

```
/**
逆地理信息
*/
@interface MPLBSReGeocodeInfo : NSObject

@property (nonatomic, strong) NSString* country; // 国家
@property (nonatomic, strong) NSString* countryCode; // 国家编码
@property (nonatomic, strong) NSString* provience; // 省
@property (nonatomic, strong) NSString* city; // 城市
@property (nonatomic, strong) NSString* district; // 区
@property (nonatomic, strong) NSString* street; // 街道
@property (nonatomic, strong) NSString* streetCode; // 街道编码
@property (nonatomic, strong) NSString* cityCode; // 城市编码
@property (nonatomic, strong) NSString* adCode; // 行政区划编码
@property (nonatomic, strong) NSArray* poiList; // poi 信息列表

@end

/**
定位结果的位置信息数据结构
*/
@interface MPLBSLocationInfo : NSObject

@property (nonatomic, strong) CLLocation* location; // 位置信息
@property (nonatomic, strong) MPLBSReGeocodeInfo* rgcInfo; // 逆地理信息

@end
```

代码示例

```
- (void)getLocation {
MPLBSConfiguration *configuration = [[MPLBSConfiguration alloc] init];
configuration.desiredAccuracy = kCLLocationAccuracyBest;

self.locationManager = [[MPLBSLocationManager alloc] initWithConfiguration:configuration];
[self.locationManager requestLocationNeedReGeocode:NO completionHandler:^(BOOL success,
MPLBSLocationInfo * _Nonnull locationInfo, NSError * _Nonnull error) {
NSString *message;
if (success) {
```

```
message = [NSString stringWithFormat:@"定位成功, 经度: %.5f, 维度: %.5f, 精确度: %.3f, 是否高精度 : %d",
locationInfo.location.coordinate.longitude, locationInfo.location.coordinate.latitude,
locationInfo.location.horizontalAccuracy, !locationInfo.location.ap_lbs_is_high_accuracy_close];
} else {
message = [NSString stringWithFormat:@"%@", error];
}
dispatch_async(dispatch_get_main_queue(), ^{
AUNoticeDialog *alert = [[AUNoticeDialog alloc] initWithTitle:@"定位结果"message:message delegate:nil
cancelButtonTitle:@"OK"otherButtonTitles:nil];
[alert show];
});
}];
}
```

iOS 14 适配

在 iOS 14 中，精确位置作为一个权限选项，在申请定位权限时供用户主动选择，并且在定位权限设置页面可供用户调整，如下图所示。

入参适配

在 MPLBSConfiguration 中，增加了 highAccuracyRequired 设置，如果入参 highAccuracyRequired = YES

，用户关闭高精度定位权限，则回调错误。

```
/**
定位服务的配置
*/
@interface MPLBSConfiguration : NSObject

/**
* 是否需要高精度定位，iOS14以下不区分精度，iOS14及以上默认NO(低精度)，需要业务指定是否需要高精度定位。
*/
@property (nonatomic,assign) BOOL highAccuracyRequired;

@end


//如果入参highAccuracyRequired = YES，且无高精度定位权限则回调错误
Errorcode：APCoreLocationErrorCodeHighAccuracyAuthorization
```

回调适配

如果入参 highAccuracyRequired = NO 或者 未设置，则回调的 CLLocation 对象里面会增加字段 ap_lbs_is_high_accuracy_close 以标识是否关闭高精度定位。

```
// 出参改造
@interface CLLocation (APMobileLBS)
/*
* 是否关闭精准定位，默认为NO
*/
@property(nonatomic,assign)BOOL ap_lbs_is_high_accuracy_close;
@end
```

代码示例

```
- (void)getLocationWithHighAccuracy {
MPLBSConfiguration *configuration = [[MPLBSConfiguration alloc] init];
configuration.desiredAccuracy = kCLLocationAccuracyBest;
configuration.highAccuracyRequired = YES;

self.locationManager = [[MPLBSLocationManager alloc] initWithConfiguration:configuration];
[self.locationManager requestLocationNeedReGeocode:NO completionHandler:^(BOOL success,
MPLBSLocationInfo * _Nonnull locationInfo, NSError * _Nonnull error) {
NSString *message;
if (success) {
message = [NSString stringWithFormat:@"定位成功, 经度: %.5f, 维度: %.5f, 精确度: %.3f, 是否高精度 : %d",
locationInfo.location.coordinate.longitude, locationInfo.location.coordinate.latitude,
locationInfo.location.horizontalAccuracy, !locationInfo.location.ap_lbs_is_high_accuracy_close];
} else {
message = [NSString stringWithFormat:@"%@", error];
}
dispatch_async(dispatch_get_main_queue(), ^{
AUNoticeDialog *alert = [[AUNoticeDialog alloc] initWithTitle:@"定位结果"message:message delegate:nil
```

```
cancelButtonTitle:@"OK"otherButtonTitles:nil];
[alert show];
});
}];
}
```

4 代码示例

Android 代码示例

参见 获取代码示例 以获取代码示例以及使用方法和注意事项。

iOS 代码示例

参见 获取代码示例 以获取代码示例以及使用方法和注意事项。

客户端集成定位功能的详细介绍请参见 接入 iOS 。

快速开始

1. 运行程序，输入定位接口的各参数值：
    - 场景标识：当前业务的类型，必传。建议如果是 native，传类名；如果是 h5，则传 URL。
    - 定位精确度：单位，米。建议结合业务场景传入一个可接受正数。例如，500 表示 500 米以内的范围。
    - 缓存时间：从当前时间往前推，多长时间内的缓存是有效的。推荐设置 30 秒以上的缓存时间。
    - 超时时间：定位超时的时间。单位，秒。默认值和最小值均为 2 秒。

2. 点击 **定位** 按钮，得到定位结果。

3. 点击右上角 **reset** ，重置参数。

蚂蚁集团
ANT GROUP