

Ant Technology

Social Sharing User Guide

Document Version: 20230209



**蚂蚁集团
ANT GROUP**

Legal disclaimer

Ant Group all rights reserved©2022.

No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Ant Group.

Trademark statement



and other trademarks related to Ant Group are owned by Ant Group. The third-party registered trademarks involved in this document are owned by the right holder according to law.

Disclaimer

The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Ant Group reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through channels authorized by Ant Group. You must pay attention to the version changes of this document as they occur and download and obtain the latest version of this document from Ant Group's authorized channels. Ant Group does not assume any responsibility for direct or indirect losses caused by improper use of documents.

Document conventions









Style	Description	Example
 Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
 Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
 Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: If the weight is set to 0, the server no longer receives new requests.
 Note	A note indicates supplemental instructions, best practices, tips, and other content.	 Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click Settings> Network> Set network type .
Bold	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click OK .
Courier font	Courier font is used for commands	Run the <code>cd /d C:/window</code> command to enter the Windows system folder.
<i>Italic</i>	Italic formatting is used for parameters and variables.	<code>bae log list --instanceid</code> <i>Instance_ID</i>
[] or [a b]	This format is used for an optional value, where only one item can be selected.	<code>ipconfig [-all -t]</code>
{ } or {a b}	This format is used for a required value, where only one item can be selected.	<code>switch {active stand}</code>

Table of Contents

1. Social Sharing	05
1.1. Overview	05
1.2. Integrate Android SDK	05
1.2.1. Quick start	05
1.2.2. Migrate to baseline 10.1.60	07
1.2.3. API_reference	07
1.2.3.1. ShareService interface	08
1.2.3.2. ShareType interface	08
1.2.3.3. ShareContent	09
1.2.3.4. ShareException interface	10
1.3. Integrate iOS SDK	11
1.3.1. Quick start	11
1.3.2. Use SDK	11

1.Social Sharing

1.1. Overview

MPSHareKit provides the function of sharing information through Weibo, WeChat, Alipay Friends, QQ, SMS and other channels, and offers you a unified interface, so you don't have to deal with the interface difference among various SDKs.

1.2. Integrate Android SDK

1.2.1. Quick start

About this task

The Social sharing component offers the feature of sharing to Weibo, WeChat, Alipay, QQ, DingTalk, text messages, and other channels. This component provides a unified API to the developers so that they do not need to cope with the differences among various SDK APIs. To add a share component to an Android client, you need to configure a project to determine the basic framework and add the SDK of the `share` component.

Before you begin

Before you add a sharing channel, make sure that you have registered an account on the official website of this channel. Some sample official sites of sharing channels are listed as follows:

- Weibo: <http://open.weibo.com/>
- WeChat: <https://open.weixin.qq.com/>
- QQ: <http://open.qq.com/>
- Alipay: <http://open.alipay.com/index.htm>
- DingTalk: <https://www.dingtalk.com/>

Social sharing is supported in the **native AAR mode**, the **mPaaS Inside mode**, and the **component-based mode**.

- If you want to connect the component to the mPaaS based on the native AAR mode, you need to first complete the prerequisites and the subsequent steps. For more information, see [Add mPaaS to your project](#)
- If you want to connect the component to the mPaaS based on the mPaaS Inside mode, you need to first complete the [mPaaS Inside access procedure](#).
- If you want to connect the component to the mPaaS based on components, you need to first complete the [Component-based access procedure](#).

Add the SDK

Native AAR mode

You can use the AAR Component Management (AAR) function to install the share component in your project. For more information, see [AAR component management](#).

mPaaS Inside mode

Use the Component Management function to install the share component in your project.

For more information, see [Manage component dependencies](#).

Component-based mode

In your Portal and Bundle projects, use the Component Management function to install the share component.

For more information, see [Manage component dependencies](#).

Initialize mPaaS

In the native AAR or mPaaS Inside mode, you must initialize the mPaaS.

Add the following code to the Application class:

```
public class MyApplication extends Application {

    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        // Initialize callback settings for mPaaS.
        QuinoxlessFramework.setup(this, new IInitCallback() {
            @Override
            public void onPostInit() {
                // This callback indicates that mPaaS has been initialized, and mPaaS-related calls can be made in this callback.
            }
        });
    }

    @Override
    public void onCreate() {
        super.onCreate();
        // Initialize mPaaS.
        QuinoxlessFramework.init();
    }
}
```

Use the social sharing SDK on different platforms

This topic describes how to use the social sharing SDK in the baseline version 10.1.32 and later based on the official demo of [Social sharing](#).

Share to WeChat

You need to manually create an `Activity` with a specific path and name to receive callback events for sharing to WeChat. This `Activity` inherits from `DefaultWXEntryActivity`. The path is `package_name.wxapi.WXEntryActivity`, where `package_name` is the package name of the application.

Note

The path and the 'Activity' name must be correctly specified. Otherwise, the application will fail to receive any callback.

In the following example, the package name is `com.mpaas.demo`.

```
package com.mpaas.demo.wxapi;
import com.alipay.android.shareassist.DefaultWXEntryActivity;
public class WXEntryActivity extends DefaultWXEntryActivity {
}
```

Register for the `Activity` in `AndroidManifest.xml`:

```
<application>
...
<activity android:name="com.mpaas.demo.wxapi.WXEntryActivity"
    android:exported="true"
    android:launchMode="singleTop">
</activity>
...
</application>
```

Note

When you set the WeChat share icon, make sure that the icon size does not exceed 32 KB. Otherwise, the WeChat sharing may fail. The SDK for Android needs to be validated. If the WeChat share icon exceeds 32 KB, then the default Alipay icon will be used instead.

Share to QQ and QZone

You need to register for the `Activity` required for sharing to QQ in `AndroidManifest.xml`. Otherwise, you will fail to use the features of sharing to QQ and QZone and the callback feature.

Note

- If the QQ share ID that you write in the 'AndroidManifest.xml' file is different from the QQ share ID registered in the code, a callback error will occur during the share. And the 'onException' callback is called even the share is successful. Therefore, you need to check the QQ share ID carefully.
- Enter the corresponding QQ share ID in `data android:scheme`. The ID format is 'tencent' immediately followed by a QQ ID, which is `tencent+QQID`. Note that `+` is not included in the QQ share ID. You need to register for this ID on [Tencent Open Platform](#). In the following example, the QQ ID is '1104122330'.

```
<application>
...
<activity
    android:name="com.tencent.connect.common.AssistActivity"
    android:configChanges="orientation|keyboardHidden|screenSize"
    android:screenOrientation="portrait"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"/>
<activity
    android:name="com.tencent.tauth.AuthActivity"
    android:launchMode="singleTask"
    android:exported="true"
    android:noHistory="true">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="tencent1104122330"/>
    </intent-filter>
</activity>
...
</application>
```

Share to Weibo

Make sure that the application signature, the package name, and the share ID are the same as those registered on [Weibo Open Platform](#). Otherwise, a sharing failure will occur. If a sharing failure occurs due to this inconsistency issue, the `share` component triggers the sharing success callback `onComplete` rather than the sharing exception callback `onException`. This defect comes with the Weibo SDK. And the same issue occurs in the official demo of the Weibo SDK.

Related topics

- For how to obtain the code sample and the corresponding usage and notes, see [Obtain the code sample](#).
- Related API documentation:

- [ShareService API](#)
- [ShareType API](#)
- [ShareContent API](#)
- [ShareException API](#)

1.2.2. Migrate to baseline 10.1.60

This topic describes how to migrate the Social sharing component from a baseline earlier than version 10.1.60 to the baseline 10.1.60.

Procedure

1. Uninstall the Social sharing component.

If you have installed an earlier version of the Social sharing component according to [Installation guidelines](#), you need to uninstall it by deleting the following contents in the `build.gradle` file.

```
provided "com.alipay.android.phone.mobilecommon:share-build:1.3.0.xxxx:api@jar"
bundle "com.alipay.android.phone.mobilecommon:share-build:1.3.0.xxxx:jar"
manifest "com.alipay.android.phone.mobilecommon:share-build:1.3.0.xxxx:AndroidManifest@xml"
```

2. Upgrade your baseline to version 10.1.60. Skip this step if you have upgraded the baseline.

In the Android Studio menu, choose **mPaaS > Baseline Upgrade**. After you select the baseline 10.1.60, click OK.



3. Install the Social sharing component of the baseline 10.1.60.

In the Android Studio menu, choose **mPaaS > Component Management** to add the mPaaS Social sharing component.



Note The API for Social sharing components of the baseline 10.1.60 and 10.1.32 has not changed.

1.2.3. API_reference

1.2.3.1. ShareService interface

ShareService interface:

```
public abstract class ShareService extends ExternalService {

    /**
     * Silent share, only one share type can be used, share type selection will not appear.
     * @param content: The content to share
     * @param shareType: Share type
     * @param biz: biz
     */
    public abstract void silentShare(ShareContent content, final int shareType, final String biz);

    /**
     * Set the monitored target in sharing
     * @param listener: The monitored target
     */
    public abstract void setShareActionListener(ShareActionListener listener);

    /**
     * Obtain the monitored target in sharing
     * @return: The monitored target
     */
    public abstract ShareActionListener getShareActionListener();

    /**
     * Set the name of application
     * @param name: Application name
     */
    public abstract void setAppName(String name);

    /**
     * Initialize WeChat sharing
     * @param appId: WeChat appId, registered and obtained from WeChat channel
     * @param appSecret: WeChat appSecret, registered and obtained from WeChat channel
     */
    public abstract void initWeixin(String appId, String appSecret);

    /**
     * Initialize Weibo sharing
     * @param appId: Weibo appId, registered and obtained from Weibo channel
     * @param appSecret: Weibo appSecret, registered and obtained from Weibo channel
     * @param redirectUrl: The redirect URL for Weibo sharing
     */
    public abstract void initWeibo(String appId, String appSecret, String redirectUrl);

    /**
     * Initialize Qzone sharing
     * @param appId: QZone appId, registered and obtained from QQ channel
     */
    public abstract void initQZone(String appId);

    /**
     * Initialize QQ sharing
     * @param appId: QQ appId, registered and obtained from QQ channel
     */
    public abstract void initQQ(String appId);

    /**
     * Initialize Alipay sharing
     * @param appId: Alipay appId, registered and obtained from Alipay channel
     */
    public abstract void initAlipayContact(String appId);

    /**
     * Initialize DingTalk sharing
     * @param appId: DingTalk appId, registered and obtained from DingTalk channel
     */
    public abstract void initDingDing(String appId);
}
```

1.2.3.2. ShareType interface

ShareType interface:


```
public class ShareType {
    /**
     * SMS
     */
    public static final int SHARE_TYPE_SMS = 2;

    /**
     * Weibo
     */
    public static final int SHARE_TYPE_WEIBO = 4;

    /**
     * WeChat friends
     */
    public static final int SHARE_TYPE_WEIXIN = 8;

    /**
     * WeChat Moments
     */
    public static final int SHARE_TYPE_WEIXIN_TIMELINE = 16;

    /**
     * Copy link
     */
    public static final int SHARE_TYPE_LINKCOPY = 32;

    /**
     * Qzone
     */
    public static final int SHARE_TYPE_QZONE = 256;

    /**
     * QQ friends
     */
    public static final int SHARE_TYPE_QQ = 512;

    /**
     * Contacts
     */
    public static final int SHARE_TYPE_CONTACT = 1024;

    /**
     * My timeline
     */
    public static final int SHARE_TYPE_CONTACT_TIMELINE = 2048;

    /**
     * DingTalk
     */
    public static final int SHARE_TYPE_DINGDING = 4096;

    /**
     * Group
     */
    public static final int SHARE_TYPE_GROUP = 8192;

    /**
     * All
     */
    public static final int SHARE_TYPE_ALL = 65535;
}
```

1.2.3.3. ShareContent

This topic describes how to use the `ShareContent` operation, as shown in the following code:

② Note

- Call `get` and `set` method to access `ShareContent` variable.
- Since there is no uniform standard for sharing, `imgUrl` is used to internally “smooth” the differences. Therefore `imgUrl` is the first choice for sharing, use `image` only if `imgUrl` is unavailable.

```
public class ShareContent implements Serializable {

    /**
     * Content to share
     */
    private String content;

    /**
     * Share image
     */
    private byte[] image;

    /**
     * Share URL
     */
    private String url;

    /**
     * Share title
     */
    private String title;

    /**
     * Share image URL
     */
    private String imgUrl;

    /**
     * Extended parameter: Used for passing the business parameters of generating short link and SMS sending success
     */
    private String extData;

    /**
     * Sharing type: "url" is used to share the link, "image" is used to share the picture
     */
    private String contentType;

    /**
     * Share to contact: Share the URL of the small icon in the preview box
     */
    private String iconUrl;

    /**
     * Local image URL
     */
    private String localImageUrl;
}
```

1.2.3.4. ShareException interface

ShareException interface:

```
public class ShareException extends RuntimeException {

    /**
     * Status code: User cancelled
     */
    public static final int USER_CANCEL = 1001;

    /**
     * Status code: Authentication failed
     */
    public static final int AUTH_ERROR = 1002;

    /**
     * Status code: Unknown error
     */
    public static final int UNKNOWN_ERROR = 1003;

    /**
     * Status code: Application not installed
     */
    public static final int APP_UNINSTALL = 40501;

    /**
     * Obtain status code
     * @return
     */
    public int getStatusCode() {
        return this.statusCode;
    }
}
```

1.3. Integrate iOS SDK

1.3.1. Quick start

The Social sharing component MPShareKit offers the feature of sharing to Weibo, WeChat, Alipay, QQ, DingTalk, text messages and other channels. It provides a unified interface for developers so that they do not need to cope with the differences among various SDK interfaces.

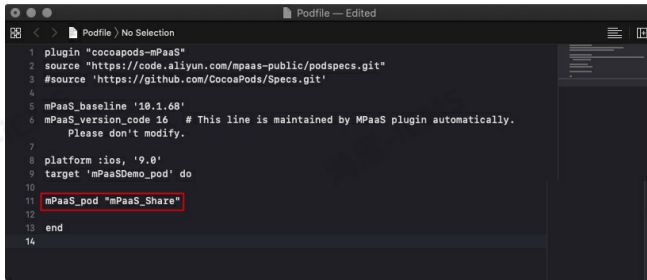
Prerequisites

You have connected your project to mPaaS. For more information, see [Access based on native framework and using Cocoapods](#).

Add the SDK

Use CocoaPods plugin to add the SDK. Complete the following steps:

1. In the Podfile file, add `mPaaS_pod "mPaaS_Share"` to add the dependencies of the sharing component.



2. Run `pod install` to connect the component to the mPaaS.

What to do next

[Use the SDK \(Version 10.1.60 and later\)](#)

1.3.2. Use SDK

After you add the SDK for the Social sharing component, you need to configure the project. You can start to use this component after it has been initialized. [TI](#)

Configure the project

Configure the allowlist of third-party applications

In iOS 9 and later, configure the Scheme list to support application operations such as social sharing and authorization. The system will automatically check w

Key	Type	Value	
▼ Information Property List	Dictionary	(22 items)	
Localization native development re...	String	en	
Executable file	String	\$(EXECUTABLE_NAME)	
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)	
InfoDictionary version	String	6.0	
Bundle name	String	\$(PRODUCT_NAME)	
Bundle OS Type code	String	APPL	
Bundle versions string, short	String	1.0	
Bundle creator OS Type code	String	????	
► URL types	Array	(1 item)	
Bundle version	String	1	
▼ LSApplicationQueriesSchemes	Array	(15 items)	
Item 0	String	weixinULAPI	Wechat
Item 1	String	wechat	
Item 2	String	weixin	
Item 3	String	sinaweibohd	Weibo
Item 4	String	sinaweibo	
Item 5	String	weibosdk	
Item 6	String	weibosdk2.5	Alipay
Item 7	String	alipay	
Item 8	String	alipayShare	QQ
Item 9	String	mqq	
Item 10	String	mqqapi	
Item 11	String	mqqwpa	
Item 12	String	mqqOpensdkSSoLogin	dingTalk
Item 13	String	dingtalk	
Item 14	String	dingtalk-open	
► App Transport Security Settings	Dictionary	(1 item)	

Configure URL Scheme

To ensure that you can jump back to the current application from other channel applications, you need to add the urlScheme value in the Info.plist file of the

- The scheme for WeChat is the assigned key .

- The scheme for Weibo is "wb" + key .
- The scheme for QQ is "tencent" + APPID .
- The Identifier for Alipay is alipayShare, and the scheme for it is 'ap' + APPID .

Initialization

When using the Social sharing component, you first need to create the application information in the corresponding third-party platform, then use this information for registration. The information includes appId, appSecret, and universalLink. The methods used for registration are

When using mPaaS framework

Implement the following method in the classification of `DTFrameworkInterface` , and return the values of key and secret in the form of a dictionary in this met

Note

In version 10.1.60 and later of the mPaaS SDK, the WeChat SDK has been updated to version 1.8.6.1, which required Universal Link verification. Therefore, yo

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    NSDictionary *configDic = @{
        @"weixin" : @{@"key":@"wxc5c09c98c276ac86", @"secret":@"d56057d8a43031bdc178991f6eb8dcd5", @"universalLink":@"https://mpaas.example.com/"},
        @"weibo" : @{@"key":@"1877934830", @"secret":@"1067b501c42f484262c1803406510af0"},
        @"qq" : @{@"key":@"1104122330", @"secret":@"WyZkbNmE6d0rDTL"},
        @"alipay" : @{@"key":@"2015060900117932"}/*The bundleID for the key is "com.alipay.share.demo". If you want to use it for testing, modify it to the key you a ppli
        ed or modify the bundleID to "com.alipay.share.demo"*/
        @"dingTalk" : @{@"key":@"dingoaa4aipzuf2yifw17s"};
    [APSKClient registerAPPConfig:configDic];
}
```

When not using mPaaS framework

Register the key in the initialization method of the accessed application.

Note

In version 10.1.60 and later of the mPaaS SDK, the WeChat SDK has been updated to version 1.8.6.1, which requires Universal Link verification. Therefore, yo

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    NSDictionary *dic = @{
        @"weixin" : @{@"key":@"wxc5c09c98c276ac86", @"secret":@"d56057d8a43031bdc178991f6eb8dcd5", @"universalLink":@"https://mpaas.example.com/"},
        @"weibo" : @{@"key":@"1877934830", @"secret":@"1067b501c42f484262c1803406510af0"},
        @"qq" : @{@"key":@"1104122330", @"secret":@"WyZkbNmE6d0rDTL"},
        @"alipay" : @{@"key":@"2015060900117932"}/*The bundleID for the key is "com.alipay.share.demo". If you want to use it for testing, modify it to the key you appli
        ed or modify the bundleID to "com.alipay.share.demo"*/
        @"dingTalk" : @{@"key":@"dingoaa4aipzuf2yifw17s"};
    [APSKClient registerAPPConfig:dic];
}
```

Basic functions

Social sharing

Trigger the Social sharing panel

You can specify the channel to be displayed while triggering the Social sharing panel.

```
NSArray *channelArr = @[kAPSKChannelQQ, kAPSKChannelLaiwangContacts, kAPSKChannelLaiwangTimeline, kAPSKChannelWeibo, kAPSKChannelWeixin, kAPSKChannel
CopyLink, kAPSKChannelDingTalkSession];

self.launchPad = [[APSKLaunchpad alloc] initWithChannels:channelArr sort:NO];
self.launchPad.delegate = self;
[self.launchPad showForView:[UIApplication sharedApplication] keyWindow] animated:YES;
```

Complete social sharing

Execute the social sharing operation in the callback of the `sharingLaunchpad` method in `@protocol APSKLaunchpadDelegate` .

```

- (void)sharingLaunchpad:(APSKLaunchpad *)launchpad didSelectChannel:(NSString *)channelName
{
    [self shareUrl:channelName];
    [self.launchPad dismissAnimated:YES];
}

- (void)shareUrl:(NSString *)channelName
{
    // Generate data and call the corresponding channel for social sharing.
    APSKMessage *message = [[APSKMessage alloc] init];
    message.contentType = @"url"; //The types are "text", "image", and "url".
    message.content = [NSURL URLWithString:@"www.example.com.cn"];
    message.icon = [UIImage imageNamed:@"1"];
    message.title = @"Title";
    message.desc = @"Description";

    APSKClient *client = [[APSKClient alloc] init];

    [client shareMessage:message toChannel:channelName completionBlock:^(NSError *error, NSDictionary *userInfo) {
        // userInfo is the extension information.
        if(!error)
        {
            //your logistic
        }
        NSLog(@"error = %@", error);
    }];
}

```

Jump back from the channel application

- When using the mPaaS framework, you do not need to manually handle it.
- When not using the mPaaS framework, refer to the following code snippet.

```

- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication annotation:(id)annotation
{
    // After the social sharing is complete, return from the channel application to the source application.
    BOOL ret;
    ret = [APSKClient handleOpenURL:url];
    return ret;
}

```

Open service

Through open interfaces of third-party services, you can call other open services provided by the Social sharing SDK of the third-party channel. The open services that are currently supported are One-time subscription information and Launch Mini Program services provided by WeChat.

Operating procedure for requesting open services

The following code snippet shows the operating procedure for requesting open services:

```

// 1. Create the request object.
APSKOpenServiceRequest *req = [APSKOpenServiceRequest new];
// 2. Set the request type.
req.requestType = APSKOpenServiceRequestTypeLaunchMini;
// 3. Set the required parameters.
MPSKLaunchMiniProgramParam *param = [[MPSKLaunchMiniProgramParam alloc] init];
param.userName = @"xxxxxxx";
param.path = @"/index.html";
param.miniProgramType = MPSKWXMiniProgramTypeTest;

req.param = param;

// 4. Create the client object
APSKClient *client = [APSKClient new];
// 5. Request service
[client requestOpenService:req toChannel:kAPSKChannelWeixin completionBlock:^(NSError *error, NSDictionary *userInfo) {
    // 6. Callback.
    if (error) {
        NSLog(@"%@", error);
    }
}];

```

Jump back from the channel application

After you jump back from the channel application, the handling depends on whether the mPaaS framework is used on the client.

- If the mPaaS framework is used on the client, the framework will handle it after the jump back.
- If the mPaaS framework is not used on the client, refer to the following code snippet to handle it after the jump back.

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication annotation:(id)annotation
{
    // Return from the channel application to the source application.
    BOOL ret;
    ret = [APSKClient handleOpenURL:url];
    return ret;
}
```