

# Ant Technology

Access iOS  
User Guide

Document Version: 20250303



# 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**



蚂蚁集团 ANT GROUP 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   |
|--|---|---|
|  <b>Danger</b>  | A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results. |  <b>Danger:</b><br>Resetting will result in the loss of user configuration data.                                       |
|  <b>Warning</b> | A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results. |  <b>Warning:</b><br>Restarting will cause business interruption. About 10 minutes are required to restart an instance. |
|  <b>Notice</b>  | A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.      |  <b>Notice:</b><br>If the weight is set to 0, the server no longer receives new requests.                              |
|  <b>Note</b>    | A note indicates supplemental instructions, best practices, tips, and other content.  |  <b>Note:</b><br>You can use Ctrl + A to select all files.   |
| >  | Closing angle brackets are used to indicate a multi-level menu cascade.   | Click <b>Settings&gt; Network&gt; Set network type</b> .  |
| <b>Bold</b>  | Bold formatting is used for buttons, menus, page names, and other UI elements.  | Click <b>OK</b> .   |
| 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><br><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.Integration method introduction .....                            | 06 |
| 2.Create an application .....                                      | 07 |
| 3.Integrate by using CocoaPods based on the existing project ..... | 09 |
| 4.Advanced guide .....   | 18 |
| 4.1. mPaaS directory structure .....                               | 18 |
| 4.2. mPaaS iOS framework .....                                     | 19 |
| 4.3. mPaaS Micro Applications and Services .....                   | 31 |
| 4.3.1. Create a micro application .....                            | 31 |
| 4.3.2. Create a service .....                                      | 33 |
| 4.3.3. Manage the micro application and service .....              | 35 |
| 4.3.4. Code sample of micro application .....                      | 37 |
| 4.4. iOS language settings .....                                   | 40 |
| 4.5. Customize the city selection .....                            | 41 |
| 5.Adaption to iOS .....  | 43 |
| 5.1. mPaaS 10.2.3 Support Wireless Security Guard&Blue Sh... ..... | 43 |
| 5.2. mPaaS 10.2.3 Adaptation to Xcode15 .....                      | 45 |
| 5.3. mPaaS 10.2.3 Adaptation to Xcode14 .....                      | 47 |
| 5.4. mPaaS 10.2.3 Adapts to iOS 18 .....                           | 49 |
| 5.5. Adapt mPaaS 10.2.3 to iOS 17 .....                            | 50 |
| 5.6. Use mPaaS 10.2.3 and adapt mPaaS 10.2.3 to iOS 16 .....       | 51 |
| 5.7. Adapt to WKWebView .....                                      | 53 |
| 5.8. Privacy permission .....                                      | 57 |
| 5.9. Upgrade guide of mPaaS 10.1.68 .....                          | 62 |
| 5.10. mPaaS 10.1.60 baseline upgrade guide .....                   | 64 |
| 5.11. mPaaS 10.1.68 baseline adapt to Xcode 13 .....               | 68 |
| 5.12. mPaaS 10.1.68 baseline adapt to iOS 15 .....                 | 68 |

---

|  |    |
|--|----|
| 5.13. mPaaS 10.1.68 baseline adapt to iOS 14 ----- | 69 |
| 5.14. mPaaS 10.1.60 baseline adapt to iOS 13 ----- | 70 |
| 5.15. mPaaS 10.1.32 baseline adapt to iOS 13 ----- | 73 |
| 6.Reference -----                                  | 78 |
| 6.1. Customize the iOS navigation bar -----        | 78 |
| 6.2. Handle iOS conflict -----                     | 88 |
| 6.3. Switch iOS environment -----                  | 90 |
| 7.Flutter Project Access Operation Guide -----     | 94 |
| 8.FAQ of mPaaS framework -----                     | 96 |

# 1. Integration method introduction

Based on the progress and application scenarios of the iOS development project, you are recommended to integrate the mPaaS mobile development platform by using CocoaPods based on the existing project.

According to the progress and usage scenarios of iOS development project, we recommend you to use the method of **Integrate by using CocoaPods based on the existing project** to Integrate mPaaS.

## Integrate by using CocoaPods based on the existing project

If the existing project uses Cocoapods to manage SDK dependencies, we recommend that you use Cocoapods to access. For the procedure, see [Integrate by using CocoaPods based on the existing project](#).

In iOS, mPaaS uses the Objective-C development language. If your project uses the Swift development language, you can introduce mPaaS Objective-C code by bridging.

### Note

If you encounter problems on mPaaS integration, please search the group number 31591197 with DingTalk to join DingTalk group for further communication.

## 2. Create an application

To use mPaaS, you must first create an app in the mPaaS console and download a configuration file.

### Prerequisites

You have a developer account. For details about registering an account, see [Register accounts](#).

### Create an mPaaS app

1. Log on to the [mPaaS console](#).

#### Note

If you are using mPaaS on another platform (such as the Ant Financial open platform), log on to the mPaaS console of the corresponding platform.

2. Click the **Create an application** button.
3. Complete the app information.
  - i. Enter the app name. Example: mPaaS Demo.
  - ii. Click  to upload the app icon. You can skip this step, and the app will use the default icon in this case.
4. Click **OK** to finish creating the app. The application you just created is displayed.

### Download the configuration file

1. On the app list page, click the name of the created app (such as the mPaaS Demo app created in the previous step), and the following page appears:
  - The app name appears at the top left, where you can switch apps.
  - The left side of the page displays a list of component services provided by mPaaS.
2. Click **iOS code configuration** button to go to the **Connect mPaaS to my application** page.
3. On the **Connect mPaaS to my application** page, click **download the configuration file** to go to the **Code Configuration** page.
4. On the **Code configuration** page, enter **Bundle ID**, click the **Download configuration** button, and download the app configuration file in `.config` format to the local computer for subsequent development.

The content of the downloaded configuration file is in `JSON` format, as shown in the following example:



# 3. Integrate by using CocoaPods based on the existing project

This section describes how to generate configurations based on the native plug-in extension mechanism of CocoaPods to quickly integrate mPaaS.

## Prerequisites

- [CocoaPods 1.0.0](#) or a later version has been installed. The project to be connected is the [CocoaPods](#) project.
- The CocoaPods mPaaS plug-in has been installed. If the plug-in has not been installed, run the following command to install it.

```
sh <(curl -s http://mpaas-ios.oss-cn-hangzhou.aliyuncs.com/cocoapods/installmPaaS-CocoaPodsPlugin.sh)
```

- An application has been created in the console, and the `.config` configuration file has been downloaded. For more information, see [Create an application in the console](#).

## Procedure

1. Copy the `.config` configuration file to the root directory (at the same level with `Podfile`) of the project.

### Note

Please ensure the `.config` file's filename is ending with `iOS`. If it is ending with `ios`, it needs to be updated to `iOS` manually.

2. Run the `pod mpaas init` command to automatically process the `Podfile` file and add `plugin`, `source`, and `mPaaS_baseline` configurations. The code for automatic configuration is as follows:

```
plugin "cocoapods-mPaaS"  
source 'https://gitee.com/mpaas/podsspecs.git'  
mPaaS_baseline 'x.x.x'
```

3. Configure the `Podfile` file.
  - i. Modify `mPaaS_baseline` to specify the mPaaS baseline.  
For example, in `mPaaS_baseline '10.2.3'`, `10.2.3` is the baseline version. For version differences, see [Release Note](#).
  - ii. Use `mPaaS_pod` to add mPaaS component dependencies.  
For example, in `mPaaS_pod "mPaaS_Nebula"`, `mPaaS_Nebula` is a component name. The following table shows more component names.

| Component Configuration | Applicable Baseline | Description |
|-------------------------|---------------------|-------------|
|-------------------------|---------------------|-------------|

|                                |          |  |
|--------------------------------|----------|--|
| mPaaS_pod<br>"mPaaS_LocalLog"  | 10.1.32+ | Local log.   |
| mPaaS_pod "mPaaS_Log"          | 10.1.32+ | Mobile analysis: behavior log,<br>automation log, Crash log,<br>and performance log analysis.  |
| mPaaS_pod<br>"mPaaS_Diagnosis" | 10.1.32+ | Diagnosis: Client diagnosis<br>and analysis.   |
| mPaaS_pod "mPaaS_RPC"          | 10.1.32+ | Mobile gateway: provides<br>download, upload, RPC call<br>and other functions.   |
| mPaaS_pod "mPaaS_Sync"         | 10.1.32+ | Mobile synchronization: long<br>connection service.  |
| mPaaS_pod "mPaaS_Push"         | 10.1.32+ | Message push.  |
| mPaaS_pod "mPaaS_Config"       | 10.1.32+ | Switch configuration: pull the<br>corresponding value from the<br>server based on the key to<br>dynamically control the client<br>logic.   |
| mPaaS_pod "mPaaS_Upgrade"      | 10.1.32+ | Upgrade release: provides<br>convenient service to<br>proactively detect and<br>upgrade, which can be used<br>for daily phased release and<br>online new version update<br>reminder. |
| mPaaS_pod "mPaaS_Share"        | 10.1.32+ | Sharing: supports sharing text<br>and pictures to well-known<br>channels such as Weibo,<br>DingTalk, and Alipay friends.   |
| mPaaS_pod "mPaaS_Nebula"       | 10.1.32+ | HTML5 container and offline<br>package: Nebula container,<br>which supports interaction<br>between the front end and<br>the native.  |
| mPaaS_pod "mPaaS_UTDID"        | 10.1.32+ | Device ID: easily and quickly<br>obtained for the application to<br>find a specific device safely<br>and effectively.  |

|                                      |          |   |
|--------------------------------------|----------|---|
| mPaaS_pod<br>"mPaaS_DataCenter"      | 10.1.32+ | Unified storage: provides secure, fast, encryptable KV storage that supports multiple data types. The database DAO supports multiple solutions for persistent data storage.                                 |
| mPaaS_pod<br>"mPaaS_ScanCode"        | 10.1.32+ | Code scan: quickly identifies QR codes and barcodes.  |
| mPaaS_pod "mPaaS_LBS"                | 10.1.32+ | Mobile positioning: positioning solution for mobile clients.  |
| mPaaS_pod<br>"mPaaS_CommonUI"        | 10.1.32+ | Universal UI: universal UI component library that provides various UI components.   |
| mPaaS_pod<br>"mPaaS_BadgeService"    | 10.1.32+ | Badge: badge reminder component of the client, which supports reminder styles such as badge, number, and New. It can automatically manage the badge relationship of a tree structure.                       |
| mPaaS_pod<br>"mPaaS_AlipaySDK"       | 10.1.32+ | Quick Pay function of Alipay: quick payment platform of Alipay.   |
| mPaaS_pod<br>"mPaaS_Multimedia"      | 10.1.32+ | Multimedia component: supports image download, upload, cache and other functions.   |
| mPaaS_pod<br>"mPaaS_MobileFramework" | 10.1.32+ | Mobile framework: client application framework, sub-app management, multi-tab application management, third-party redirection management, viewController redirection, and exception handling and reporting. |
| mPaaS_pod<br>"mPaaS_OpenSSL"         | 10.1.32+ | OpenSSL   |
| mPaaS_pod "mPaaS_TinyApp"            | 10.1.32+ | Applets: integrates lease capabilities.   |

|                              |          |   |
|------------------------------|----------|---|
| mPaaS_pod "MPBaseTest"       | 10.1.32+ | Basic test: basic test module.  |
| mPaaS_pod "mPaaS_CDP"        | 10.1.32+ | Intelligent delivery: configure various intelligent advertisements and presentations to the clients with dynamic launches.                              |
| mPaaS_pod "mPaaS_AliAccount" | 10.1.60+ | Mini program: publish the mini program  |
| mPaaS_pod "mPaaS_ARTVC"      | 10.1.68  | Voice call and video call: Voice call and video call components. This feature supports two-party video calls, group video calls and online conferences. |
| mPaaS_pod "mPaaS_BlueShield" | 10.2.3+  | Blue Shield encryption component: Add the absBase64Code parameter in the config file to automatically generate a Blue Shield image.                     |
| mPaaS_pod "mPaaS_MDC"        | 10.2.3+  | Mobile Dispatch component: Fine-grained domain name strategic scheduling.   |

See the following example of complete Podfile:

```
Pods ) Podfile ) No Selection
1 # mPaaS Pods Begin
2 plugin "cocoapods-mPaaS", :guard_image_version => 6
3 source "https://gitee.com/mpaas/podsspecs.git"
4 mPaaS_baseline '10.2.3' # x.x.x :
5 mPaaS_version_code 34 # This line is maintained by MPaaS plugin automatically. Please don't modify.
6
7 # mPaaS Pods End
8
9 # Uncomment the next line to define a global platform for your project
10 platform :ios, "11.0"
11
12 target "MPRPCDemo_pod" do
13   # Pods for MPRPCDemo_pod
14   #remove_pod "mPaaS_Security"
15   mPaaS_pod 'mPaaS_RPC'
16   mPaaS_pod 'mPaaS_MDC'
17   mPaaS_pod "mPaaS_CommonUI"
18   mPaaS_pod "MPBaseTest"
19   mPaaS_pod "mPaaS_Push"
20   mPaaS_pod "mPaaS_MobileFramework"
21   mPaaS_pod "mPaaS_BlueShield"
22 end
23
```

- Execute `pod mpaas update x.x.x` in the command line, `x.x.x` is the version number of configured baseline, such as `10.2.3`.
- Execute `pod install` to complete integration. You can also add `--verbose` to view log details.

**Note**

If you are prompted when you execute `pod install` that you cannot find the library imported from GitHub's official website, specify the source address for the official GitHub Source at the beginning of podfile: `https://github.com/CocoaPods/Specs.git` .

6. If you find the third-party library conflicts after accessing it, you can remove the specific third-party library. For the specific operation, see [iOS conflict processing](#).

## Upgrading instructions

When a new version of mPaaS is published, you can select the upgrading components, or the general upgrading baseline (namely SDK version).

### Upgrading components

1. When you execute `pod mpaas update x.x.x` in the command line, `x.x.x` is the baseline version number currently in use, such as `10.2.3` .

```
[→ ~ pod mpaas update 10.2.3
1. update 10.2.3 baseline file ...
updateTime : 2023-06-09 10:32:15 +0800
The baseline has no change ...
The current verison is updated to 10.2.3.24
  update 10.2.3 baseline file Done

2. update mPaaS repo ...
Updating spec repo `gitee-mpaas-podspecs`
$ /opt/local/bin/git -C
/Users/yanjinqun/.cocoapods/repos/gitee-mpaas-podspecs fetch origin
--progress
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 22 (delta 12), reused 0 (delta 0), pack-reused 0
From https://gitee.com/mpaas/podspecs
   eed5568c..94fb3f1f master    -> origin/master
$ /opt/local/bin/git -C
/Users/yanjinqun/.cocoapods/repos/gitee-mpaas-podspecs rev-parse --abbrev-ref
HEAD
master
$ /opt/local/bin/git -C
/Users/yanjinqun/.cocoapods/repos/gitee-mpaas-podspecs reset --hard
origin/master
HEAD is now at 94fb3f1f PodSpec Update at 2023-06-19 12:19:25
  update mPaaS repo Done
```

2. Execute `pod install` to complete upgrading for the corresponding components under this baseline.

### Upgrade the baseline

1. In `podfile` , modify the corresponding baseline number of `mPaaS_baseline` to complete upgrading for the general baseline. For example, you can modify the baseline number from `10.1.68` to `10.2.3` . The baseline number supports the standard or custom baseline.

```
Pods ) Podfile ) No Selection
1 # mPaaS Pods Begin
2 plugin "cocoapods-mPaaS", :guard_image_version => 6
3 source "https://gitee.com/mpaas/podspecs.git"
4 mPaaS_baseline '10.2.3' # x.x.x
5 mPaaS_version_code 34 # This line is maintained by MPaaS plugin automatically. Please don't modify.
6
7 # mPaaS Pods End
8
9 # Uncomment the next line to define a global platform for your project
10 platform :ios, "11.0"
11
12 target "MPRPCDemo_pod" do
13   # Pods for MPRPCDemo_pod
14   #remove_pod "mPaaS_Security"
15   mPaaS_pod 'mPaaS_RPC'
16   mPaaS_pod 'mPaaS_MDC'
17   mPaaS_pod 'mPaaS_CommonUI'
18   mPaaS_pod 'MPBaseTest'
19   mPaaS_pod 'mPaaS_Push'
20   mPaaS_pod 'mPaaS_MobileFramework'
21   mPaaS_pod 'mPaaS_BlueShield'
22 end
23
```

2. Execute `pod install` to complete baseline upgrading.

## mPaaS iOS podspec address switch

### Background

The original podspec storage warehouse `code.aliyun.com` used by mPaaS has ceased service (updates will cease on June 1, 2023, and services will cease on June 30, 2023).

Continuing to use the original repo will have the following impacts:

1. When using the mPaaS pod plugin for SDK update, the latest version of each baseline cannot be pulled;
2. After 2023.06.30, using the mPaaS pod plugin cannot pull to any baseline version.

Currently mPaaS has supported all mPaaS versions on `gitee.com`.

### Solutions

#### Upgrade mPaaS pod plugin

Execute the following command to update to the latest mPaaS pod plugin.

```
sh <(curl -s http://mpaas-ios.oss-cn-  
hangzhou.aliyuncs.com/cocoapods/installmPaaS CocoaPodsPlugin.sh)
```

After the execution is complete, execute the `pod plugins installed` command in the terminal to check the version of cocoapods-mPaaS. If it shows `0.9.6` or above, the upgrade is successful.

```
yanjinquan — yanjinquan@B-LMHPW0KT-1948 — ~ — -zsh — 80x24
[→ ~ pod plugins installed

Installed CocoaPods Plugins:
- cocoapods-deintegrate : 1.0.5
- cocoapods-mPaaS      : 0.9.6 (pre_install and post_install hooks)
- cocoapods-plugins   : 1.0.0
- cocoapods-search    : 1.0.1
- cocoapods-trunk     : 1.6.0
- cocoapods-try       : 1.2.0
→ ~
```

## Modify the source configuration in podfile

Replace the origin `source "https://code.aliyun.com/mpaas-public/podspecs.git"` in podfile with

```
source "https://gitee.com/mpaas/podspecs.git" .
```

## API change

This modification only involves the change of the plugin, and there is no change in the use of plugin commands.

## Test validation

After completing the above upgrade and modification configuration operations, you can continue to execute the mPaaS pod plugin related pull command to test whether the latest baseline version and SDK can be pulled.

## Parameter list

You can change some default behaviors of the plug-in by configuring parameters.

### Usage:

Add parameters in the back end of the `plugin "cocoapods-mPaaS"`. See the following examples:

```
Pods > Podfile > No Selection
1 # mPaaS Pods Begin
2 plugin "cocoapods-mPaaS", :guard_image_version => 6
3 source "https://gitee.com/mpaas/podspecs.git"
4 mPaaS_baseline '10.2.3' # x.x.x
5 mPaaS_version_code 34 # This line is maintained by MPaaS plugin automatically. Please don't modify.
6
7 # mPaaS Pods End
```

| Parameters                                | Function                | Applicable versions |
|---|-------------------------|---------------------|
| <code>:guard_image_version =&gt; 6</code> | Generate V6 guard image | ≥ V0.9.6            |
| <code>:guard_image_version =&gt; 5</code> | Generate V5 guard image | ≥ V0.9.6            |

|  |  |                |
|--|--|----------------|
| <pre>:only_frameworks =&gt; true</pre> | In some scenarios (such as the independent framework project), you do not need to add directory files with mPaaS template. | ≥ V0.9.5.0.0.2 |
| <pre>:check_repo =&gt; false</pre>     | In some cases (such as using an intranet proxy), the default added repo is not automatically checked.                      | ≥ V0.9.5.0.0.2 |

**Note**

10.2.3 The baseline version does not need to set `:guard_image_version`, and generates V6 images by default.

## Command list

After installing the cocoapods-mPaaS plug-in, you can use command line tools to assist your development.

| Command   | Function   |
|---|--|
| <pre>pod mpaas init</pre>   | In Podfile, add plugin, source and mPaaS_baseline.   |
| <pre>pod mpaas update &lt;VERSION&gt;</pre>   | Upgrade the baseline. The parameter <VERSION> is the specific baseline number such as 10.2.3. Then upgrade the podspec library.  |
| <pre>pod mpaas update --all</pre>   | In the official version of the plug-in, this command will upgrade the plug-in, and run the installation script again.<br>In the beta version of the plug-in, this command will implement features of the official version, and upgrade the local baseline. |
| <pre>pod mpaas info</pre>   | Show the complete information of the baseline and the corresponding component.   |
| <pre>pod mpaas info &lt;NAME&gt; &lt;VERSION&gt;</pre><br>, in which <VERSION> is optional. | Filter the information about a module name.  |
| <pre>pod mpaas info --only-mPaaS</pre>  | Show some default baseline information, which is easy to be pasted to Podfile.   |

|                                       |   |
|---------------------------------------|---|
| <pre>pod mpaas open</pre>             | Directly open the <code>.xcworkspace</code> file from the command line. |
| <pre>pod mpaas version</pre>          | Show the complete baseline used in the current project.                 |
| <pre>pod mpaas version --plugin</pre> | Show the version number of the current cocoapods-mPaaS plug-in.         |

# 4. Advanced guide

## 4.1. mPaaS directory structure

After importing the cloud data to a mPaaS-iOS-framework-based project or a native-framework-based project, the following directory is added to the project directory.

### 🔍 Note

In versions 10.1.32 and above, among all the directories under **MPaaS > Targets > mPaaS Demo**, only `APMobileFramework` and `mPaas` are kept. If you upgrade from an earlier version, the directories and categories of the other components will no longer be generated.

Directory structure is as follows:

```
├─ MPaaS
│  ├── mpaas_sdk.config
│  ├── Targets
│  │   └─ mPaaS Demo (Project Target name)
│  │       ├── mPaaS Demo-mPaaS-Headers.h
│  │       ├── mPaaS Demo-Prefix.pch
│  │       ├── APMobileFramework
│  │       ├── mPaas
│  │       ├── meta.config
│  │       └─ yw_1222.jpg
│  ├── Resources
│  └─ Frameworks
```

In the above directory:

- `mpaas_sdk.config` : Information about the modules added in the current project, including version, added time, resource file, and so on, which are automatically maintained by the mPaaS plugin. Do not modify it manually.
- `mPaaS Demo-mPaaS-Headers.h` : Header file of the mPaaS module that the current project depends on, which is automatically maintained by the mPaaS plugin. Do not modify it manually.
- `mPaaS Demo-Prefix.pch` : The reference of pch file, automatically adding `mPaaS Demo-mPaaS-Headers.h` into the mPaaS module's header file.
- `APMobileFramework` : The category managed by the lifecycle of mPaaS framework.
- `mPaas` : The category of `MPaaSInterface` .
- `meta.config` : The cloud metadata downloaded from mPaaS console.
- `yw_1222.jpg` : The security guard signature verification image generated through the `base64code` field in the metadata, used in mobile gateway signature verification. You can delete this image if you don't need mobile gateway.
- `Resources & Frameworks` : The mPaaS module's resource file and binary file directory, serving as the union of mPaaS modules used by all Targets in the current project. They are automatically maintained by the mPaaS plugin. Do not modify it manually.

**Note**

Since all Targets share the same `Resources & Frameworks`, different Targets cannot use different versions of the same module simultaneously. Do not modify these two directories. The framework added to `Build Phase` varies by the module selected by each Target.

## 4.2. mPaaS iOS framework

mPaaS iOS framework originates from the development framework of Alipay client. In line with the design idea of Framework, mPaaS iOS isolates business into multiple relatively independent modules and aims for achieving high cohesion and low coupling between modules.

mPaaS iOS framework directly takes over the lifecycle of application, and responsible for Host startup, managing application lifecycle, processing and distributing the delegate events of `UIApplication`, managing each business modules (MicroApplication and Services) in a unified way, etc.

This article gives a detailed introduction to the mPaaS iOS framework.

### Host startup

Through the replacement of the main function of the program, the lifecycle of the application is directly taken over. The whole startup process is as follows:

```
main -> DFClientDelegate -> Open Launcher application
```

### Application lifecycle management

After you access mPaaS framework, it completely replaces `AppDelegate`. The entire lifecycle of the application is managed by the framework, but you can still implement the delegate methods in different stages of the application's lifecycle. The framework provides the access method for all delegate methods in the `UIApplicationDelegate`, you only need to override the corresponding method in `Category`.

The life cycle method is declared as follows (see `DTFrameworkInterface.h` file for more information):

```
/**
 * The framework needs to implement certain initialization logic in
 didFinishLaunching, but this method will be called before execution.
 */
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions;

/**
 * The framework calls back this method, making the accessed application taking over its own didFinishLaunching logic.
 * When DTFrameworkCallbackResultReturnYES or DTFrameworkCallbackResultReturnNO is returned, they are directly returned to the system, without executing the subsequent logic.
 * This method is called back before starting BootLoader, application can make the framework exit in advance by returning DTFrameworkCallbackResultReturnYES or DTFrameworkCallbackResultReturnNO, without running the default BootLoader.
 * Use the default implementation in the framework, override is normally not required.
```

```
*
* @return : To continue run the framework, or return YES/NO to the system.
*/
- (DTFrameworkCallbackResult)application:(UIApplication *)application
handleDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions;

/**
 * The framework needs to implement certain initialization logic in
didFinishLaunching, but this method will be called after all the logics are done.
 */
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NS
Dictionary *)launchOptions;

/**
 * The framework calls back this method in advance, allowing the accessed application
to process the notification message in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework broadcasts the me
ssage to global listeners through UIApplicationDidReceiveRemoteNotification, and calls
completionHandler(UIBackgroundFetchResultNoData).
 * When DTFrameworkCallbackResultReturn is returned, it means the accessed application
has completely processed the message, and the framework stops executing the subsequent
logic.
 */
- (DTFrameworkCallbackResult)application:(UIApplication *)application
didReceiveRemoteNotification:(NSDictionary *)userInfo fetchCompletionHandler:(void (^)(
UIBackgroundFetchResult result))completionHandler;

/**
 * The framework calls back this method in advance, allowing the accessed application
to process the notification message in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework broadcasts the me
ssage to global listeners through UIApplicationDidReceiveLocalNotification.
 * When DTFrameworkCallbackResultReturn is returned, it means the accessed application
has completely processed the message, and the framework stops executing the subsequent
logic.
 */
- (DTFrameworkCallbackResult)application:(UIApplication *)application
didReceiveLocalNotification:(UILocalNotification *)notification;

/**
 * The framework calls back this method in advance, allowing the accessed application
to process the notification message in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework broadcasts the me
ssage to global listeners through UIApplicationDidReceiveLocalNotification, and calls c
ompletionHandler().
 * When DTFrameworkCallbackResultReturn is returned, it means the accessed application
has completely processed the message, and the framework stops executing the subsequent
logic.
 */
- (DTFrameworkCallbackResult)application:(UIApplication *)application
handleActionWithIdentifier:(NSString *)identifier forLocalNotification:
(UILocalNotification *)notification completionHandler:(void (^)(()))completionHandler;

/**
```

```
* The framework calls back this method in advance, allowing the accessed application
to get deviceToken.
* When DTFrameworkCallbackResultContinue is returned, the framework broadcasts the me
ssage to global listeners through UIApplicationDidRegisterForRemoteNotifications.
* When DTFrameworkCallbackResultReturn is returned, it means the accessed application
has completely processed, and the framework stops executing the subsequent logic.
*/
- (DTFrameworkCallbackResult)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken;

/**
* The framework calls back this method in advance when it fails to obtain
deviceToken.
* When DTFrameworkCallbackResultContinue is returned, the framework continues to exec
ute, no other logic currently.
* When DTFrameworkCallbackResultReturn is returned, the framework stops executing the
subsequent logic, no other logic currently.
*/
- (DTFrameworkCallbackResult)application:(UIApplication *)application
didFailToRegisterForRemoteNotificationsWithError:(NSError *)error;

/**
* The framework notifies sharing component (if there is, and
shouldAutoactivateShareKit returns YES) in advance, if the sharing component cannot pro
cess it, this method is called back to allow the accessed application to process openUR
L.
* When DTFrameworkCallbackResultReturnYES or DTFrameworkCallbackResultReturnNO is ret
urned, the framework directly returns to the system, without executing the subsequent l
ogic.
* When DTFrameworkCallbackResultContinue is returned, the framework continues to proc
ess the URL, and distribute it to SchemeHandler and other classes for further processin
g.
*
* Comparing with the system method, this method has an additional newURL parameter, a
llowing the application to return a different URL after processing. If the function ret
urns DTFrameworkCallbackResultContinue and assign value to newURL, the framework will u
se the new URL for subsequent processing.
*/
- (DTFrameworkCallbackResult)application:(UIApplication *)application openURL:(NSURL *)
url newURL:(NSURL **)newURL sourceApplication:(NSString *)sourceApplication annotation:
(id)annotation;

/**
* The framework calls back this method in advance.
* When DTFrameworkCallbackResultContinue is returned, the framework continues to exec
ute, no other logic currently.
* When DTFrameworkCallbackResultReturn is returned, the framework stops executing the
subsequent logic, no other logic currently.
*/
- (DTFrameworkCallbackResult)applicationWillResignActive:(UIApplication *)application;

/**
* The framework calls back this method in advance.
* When DTFrameworkCallbackResultContinue is returned, the framework continues to exec
ute, no other logic currently.
```

```
ute, no other logic currently.
 * When DTFrameworkCallbackResultReturn is returned, the framework stops executing the
subsequent logic, no other logic currently.
 */
- (DTFrameworkCallbackResult)applicationDidEnterBackground:(UIApplication
*)application;

/**
 * The framework calls back this method in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework continues to exec
ute, no other logic currently.
 * When DTFrameworkCallbackResultReturn is returned, the framework stops executing the
subsequent logic, no other logic currently.
 */
- (DTFrameworkCallbackResult)applicationWillEnterForeground:(UIApplication
*)application;

/**
 * The framework calls back this method in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework continues to exec
ute and give event to sharing component (if there is, and shouldAutoactivateShareKit re
turns YES). If the entire application is not loaded, BootLoader is called.
 * When DTFrameworkCallbackResultReturn is returned, the framework stops executing the
subsequent logic, no other logic currently.
 */
- (DTFrameworkCallbackResult)applicationDidBecomeActive:(UIApplication *)application;

/**
 * The framework calls back this method in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework continues to exec
ute, no other logic currently.
 * When DTFrameworkCallbackResultReturn is returned, the framework stops executing the
subsequent logic, no other logic currently.
 */
- (DTFrameworkCallbackResult)applicationWillTerminate:(UIApplication *)application;

/**
 * The framework calls back this method in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework continues to exec
ute, no other logic currently.
 * When DTFrameworkCallbackResultReturn is returned, the framework stops executing the
subsequent logic, no other logic currently.
 */
- (DTFrameworkCallbackResult)applicationDidReceiveMemoryWarning:(UIApplication *)applic
ation;

/**
 * The framework calls back this method in advance, allowing the accessed application
to process the Watch message in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework broadcasts the Wa
tch message to global listeners through
UIApplicationWatchKitExtensionRequestNotifications.
 * When DTFrameworkCallbackResultReturn is returned, it means the accessed application
has completely processed the message, and the framework stops executing the subsequent
logic.
```

```
*/
- (DTFrameworkCallbackResult)application:(UIApplication *)application
handleWatchKitExtensionRequest:(NSDictionary *)userInfo reply:(void (^)(NSDictionary *replyInfo))reply;

/**
 * The framework calls back this method in advance, allowing the accessed application
to process the message in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework broadcasts the message
to global listeners through UIApplicationUserActivityNotifications, and returns NO
to the system in the end.
 * When DTFrameworkCallbackResultReturnYES or DTFrameworkCallbackResultReturnNO is returned,
the framework directly returns to the system, without executing the subsequent logic.
 */
- (DTFrameworkCallbackResult)application:(UIApplication *)application
continueUserActivity:(NSUserActivity *)userActivity restorationHandler:(void (^)(NSArray *restorableObjects))restorationHandler;

/**
 * The framework calls back this method in advance, allowing the accessed application
to process the message of 3D Touch quick entry in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework processes the URL
brought by shortcutItem, and calls completionHandler() to return whether it has been processed.
 * When DTFrameworkCallbackResultReturn is returned, the framework directly returns to
the system, without executing the subsequent logic.
 */
- (DTFrameworkCallbackResult)application:(UIApplication *)application
performActionForShortcutItem:(UIApplicationShortcutItem *)shortcutItem
completionHandler:(void (^)(BOOL))completionHandler;

/**
 * Background Fetch mechanism callback
 * completionHandler must be called back in 30 seconds, otherwise the process will be
terminated.
 * To enable this mechanism, you need to configure the fetch option of Background Modes,
and then call the following method in didFinishLaunching. See documentation for more
information.
 * [application
setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalMinimum];
 * The default implementation is null, you need to process it in your own way.
 */
- (void)application:(UIApplication *)application performFetchWithCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler;
```

## Division of application modules

mPaaS framework has defined **MicroApplication** and **Service** to separate different modules. Take “whether it has an UI interface” as criteria, the Framework classifies different modules into **MicroApplication** and **Service** and implements lifecycle management on the modules via **Context**.

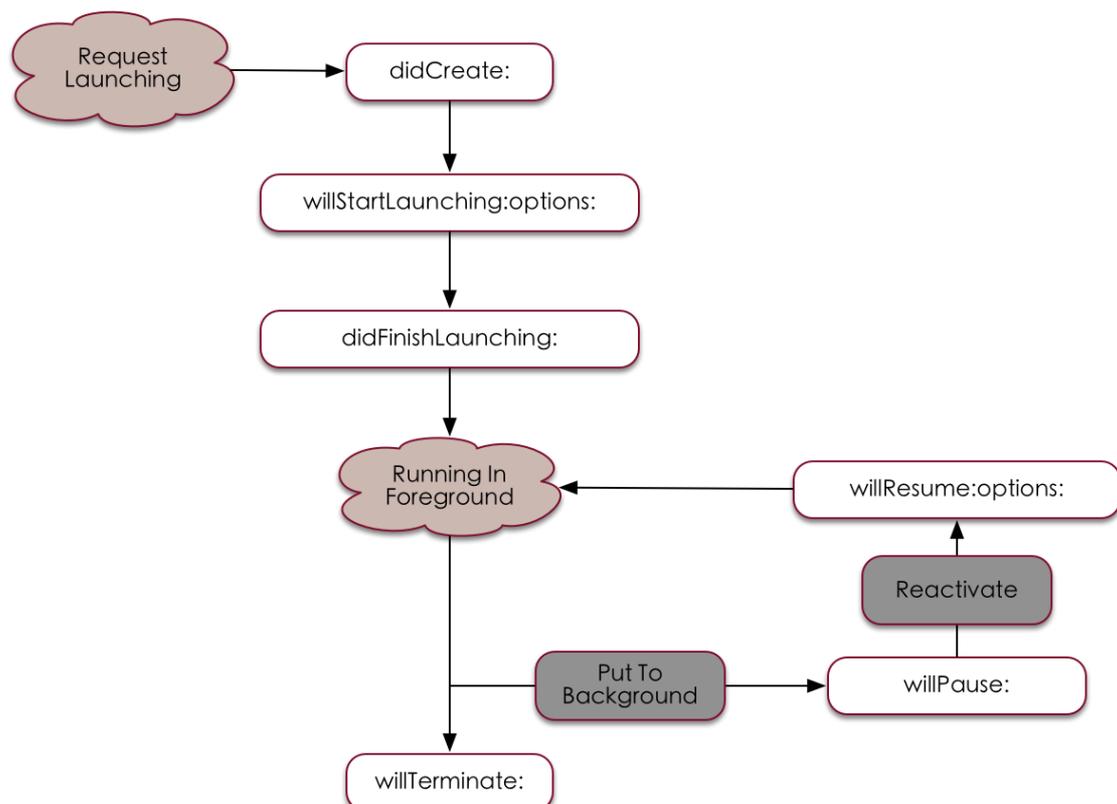
| Terminology      | Definition   |
|------------------|--|
| MicroApplication | A micro application with UI on client at runtime             |
| Service          | Lightweight abstracted service provided by client at runtime |
| Context          | Context of client micro-component at runtime                 |

This section introduces the concepts of micro application, service, and context. See [Create a micro application](#) for more information.

## MicroApplication

In the process of developing application based on mPaaS iOS framework, we generally set the independent service with UI as a micro application (e.g.: transfer, mobile top-up and other services in Alipay) and isolate it from other services to achieve high independence and zero interdependence among micro applications.

Each micro application has its own lifecycle. The overall process is as follows:



The callback methods of micro application through the whole lifecycle (see `DTMicroApplicationDelegate.h` file for more information):

```
@required  
/**
```

```
/**
 * Request the delegate of application object to return root view controller.
 *
 * @param application: Application object.
 *
 * @return: The root view controller of application.
 */
- (UIViewController *)rootControllerInApplication:(DTMicroApplication *)application;
@optional
/**
 * Notify the application delegate that the application object has been instantiated.
 *
 * @param application: Application object.
 */
- (void)applicationDidCreate:(DTMicroApplication *)application;
/**
 * Notify the application delegate that the application will be launched.
 *
 * @param application: Launched application object.
 * @param options: Running parameters of the application.
 */
- (void)application:(DTMicroApplication *)application willStartLaunchingWithOptions:(NSDictionary *)options;
/**
 * Notify the application delegate that the application is launched already.
 *
 * @param application: Launched application object.
 */
- (void)applicationDidFinishLaunching:(DTMicroApplication *)application;
/**
 * Notify the application delegate that the application will be paused and put into back
ground.
 *
 * @param application: Launched application object.
 */
- (void)applicationWillPause:(DTMicroApplication *)application;
/**
 * Notify the application delegate that the application will be reactivated.
 *
 * @param application: The application object to be activated.
 */
- (void)application:(DTMicroApplication *)application willResumeWithOptions:
(NSDictionary *)options;
/**
 * Notify the application delegate that the application has been activated.
 *
 * @param application: The application object to be activated.
 */
- (void)applicationDidResume:(DTMicroApplication *)application;
/**
 * Notify the application delegate that the application has been activated.
 *
 * @param application: The application object to be activated, together with parameter v
ersion.
 */
```

```
- (void)application:(DTMicroApplication *)application didResumeWithOptions:
(NSDictionary *)options;
/**
 * Notify the application delegate that the application will quit.
 *
 * @param application: Application object.
 */
- (void)applicationWillTerminate:(DTMicroApplication *)application;
/**
 * Notify the application delegate that the application will quit.
 *
 * @param application: Application object.
 * @param animated: Whether to quit in animated way.
 */
- (void)applicationWillTerminate:(DTMicroApplication *)application animated:
(BOOL)animated;
/**
 * Inquire the application delegate whether the application can quit or not.
 * Note: The delegate returns NO in some special cases. If it defaults to Yes, t
he application can quit.
 *
 * @param application: Application object.
 *
 * @return: Whether the application can quit or not.
 */
- (BOOL)applicationShouldTerminate:(DTMicroApplication *)application;
```

## Service

mPaaS iOS framework regards the Framework without UI as service. The differences between microapplication and service are as follows:

- Microapplication serves as an independent business process while service is used to provide general service.
- Service is stateful. Once started, the service exists through the whole lifecycle of the client and can be acquired at any time; microapplication will be destroyed after exit.

Relevant interfaces of service management (see `DTService.h` file for more information):

```
@required
/**
 * Start a service.
 * Note:
 * The framework will call the method after initialization.
 * The service can start an application only when the method is called.
 */
- (void)start;
@optional
/**
 * A service is created.
 */
- (void)didCreate;
/**
 * The service will be destroyed.
 */
- (void)willDestroy;
```

## Context

Context is the control center of the whole client framework, performing unified management on the interaction and jumps among micro applications and services, with the following responsibilities:

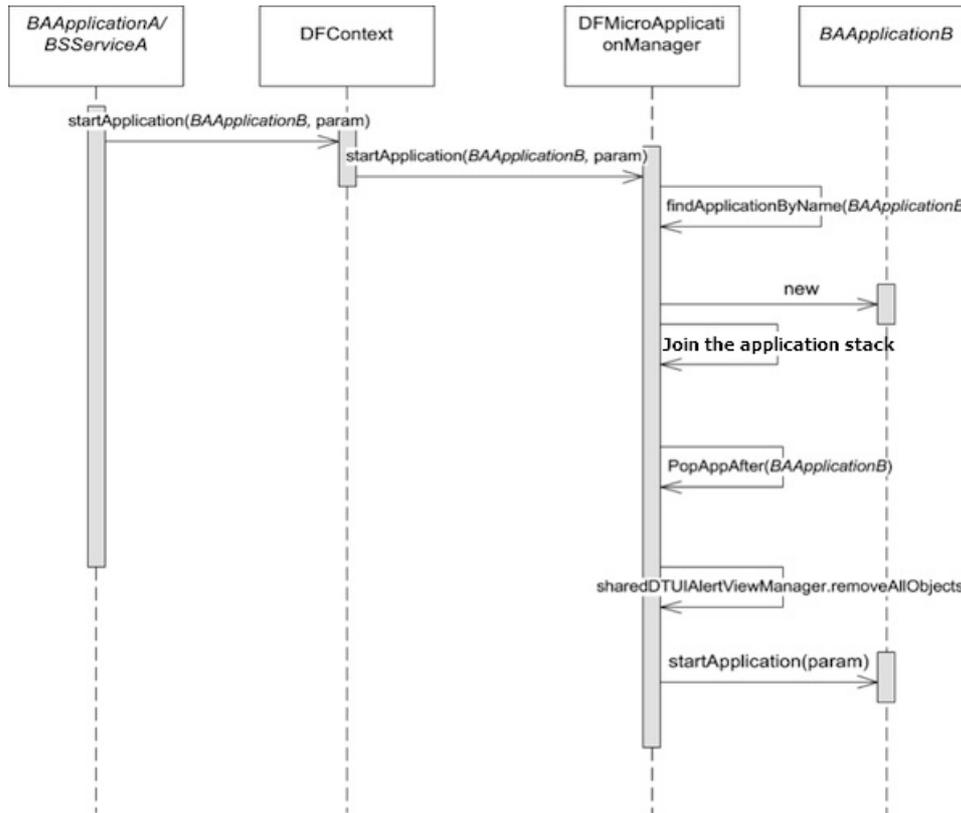
- Provide an interface for starting micro application. Users can quickly find, close and manage the jumps of the micro application through using name;
- Provide an interface for starting service, managing the registration, discovery and deregistration of services.

## Manage micro application

- Relevant interfaces of micro application management (see `DTContext.h` file for more information):

```
/**
 * Start an application as per the given name.
 *
 * @param name: Name of the application to be started.
 * @param params: The parameters need to be forwarded to another application when an app
lication is started.
 * @param animated: Specify whether to display animation when starting an application.
 *
 * @return: Return YES if the application is successfully started, otherwise NO.
 */
- (BOOL)startApplication:(NSString *)name params:(NSDictionary *)params animated:(BOOL)
animated;
/**
 * Start an application as per the given name.
 *
 * @param name: Name of the application to be started.
 * @param params: The parameters need to be forwarded to another application when an app
lication is started.
 * @param launchMode: Specify the method of starting application.
 *
 * @return: Return YES if the application is successfully started, otherwise NO.
 */
- (BOOL)startApplication:(NSString *)name params:(NSDictionary *)params launchMode:(DTM
icroApplicationLaunchMode)launchMode;
/**
 * Find the specified application.
 *
 * @param name: Name of the application to find.
 *
 * @return: Return corresponding application object if the specified application is in t
he application stack, otherwise nil.
 */
- (DTMicroApplication *)findApplicationByName:(NSString *)name;
/**
 * Return the application which is on the top of the stack currently, namely the applica
tion visible to users.
 *
 * @return: Current visible applications.
 */
- (DTMicroApplication *)currentApplication;
```

- Micro application starting process:



## Service management

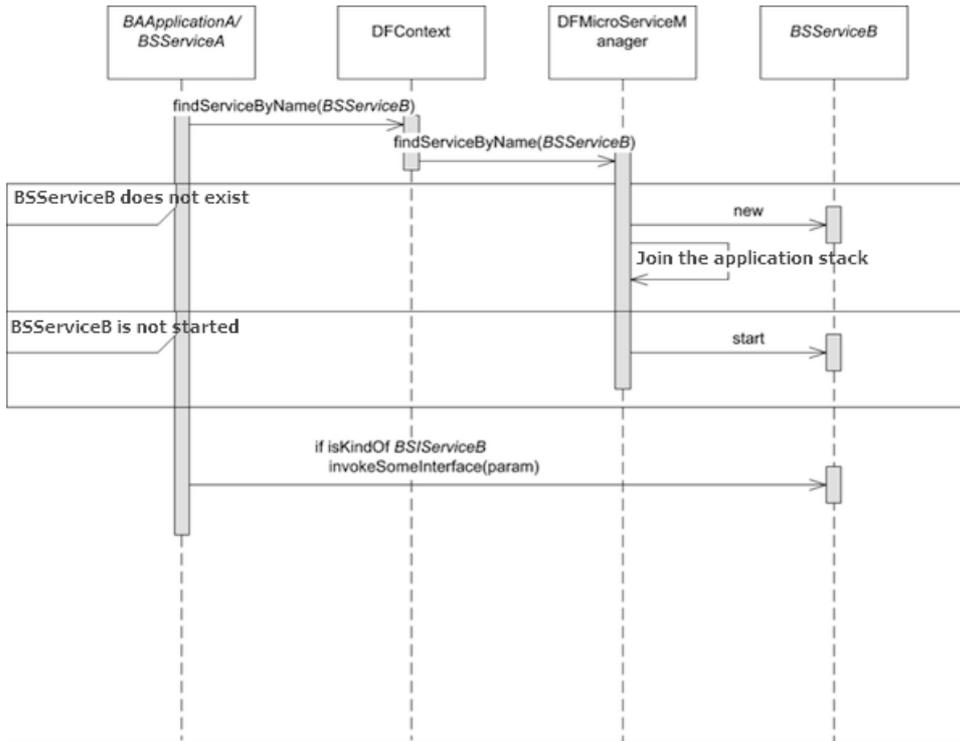
- Relevant interfaces of service management (see `DTContext.h` file for more information):

```
/**
 * Find a service as per the given name.
 *
 * @param name: Service name
 *
 * @return: Return a service object if the service with given name is found, otherwise null.
 */
- (id) findServiceByName: (NSString *) name;

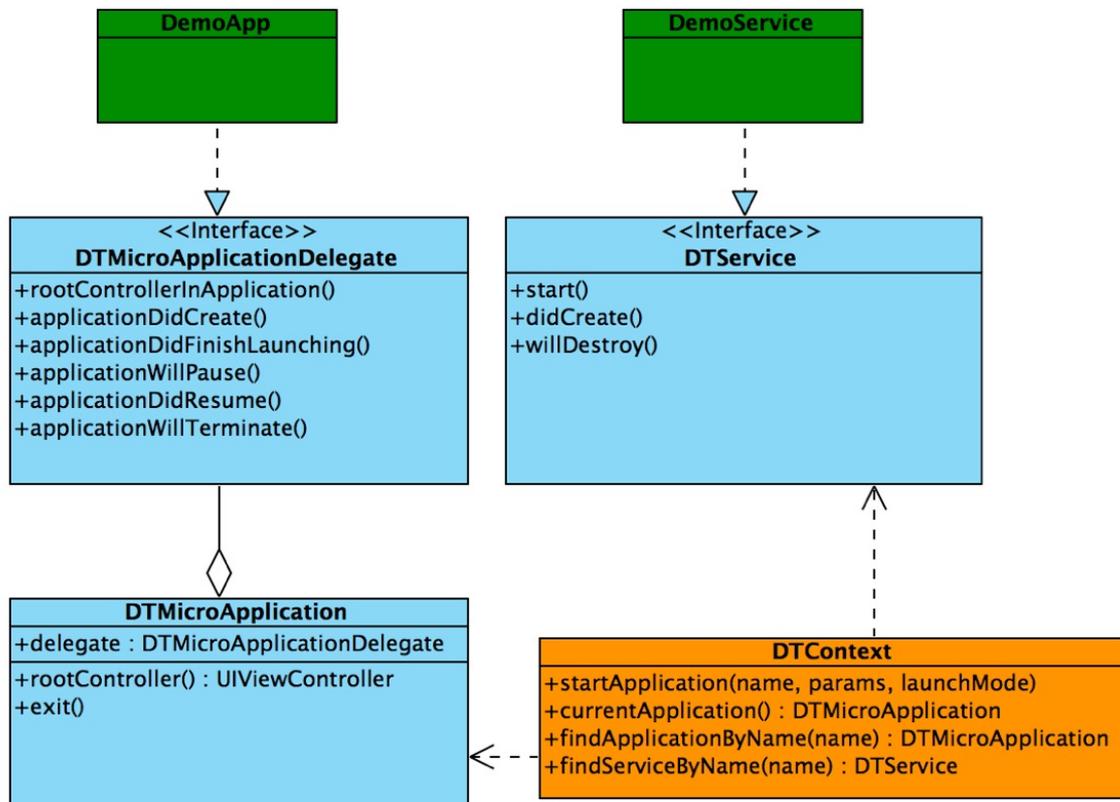
/**
 * Register a service.
 *
 * @param name: Service name
 */
- (BOOL) registerService: (id) service forName: (NSString *) name;

/**
 * Deregister an existing service.
 *
 * @param name: Service name
 */
- (void) unregisterServiceForName: (NSString *) name;
```

- Service starting process:



The UML class diagram illustrating how context manages micro application and service is shown below:



## 4.3. mPaaS Micro Applications and Services

### 4.3.1. Create a micro application

In the process of developing apps based on the mPaaS iOS framework, the independent service with a UI is often configured as a micro application (such as transfer in Alipay and recharging for mobile phones), which is isolated from other services and implements its own service logic in In the process of developing applications based on the mPaaS iOS framework, the independent business with UI interface is generally set as a micro application (such as transfer in Alipay, mobile phone recharge, etc.), which is isolated from other services and implements its own service logic in the micro application. To add a micro application, you must add micro-application template code and register the micro application.

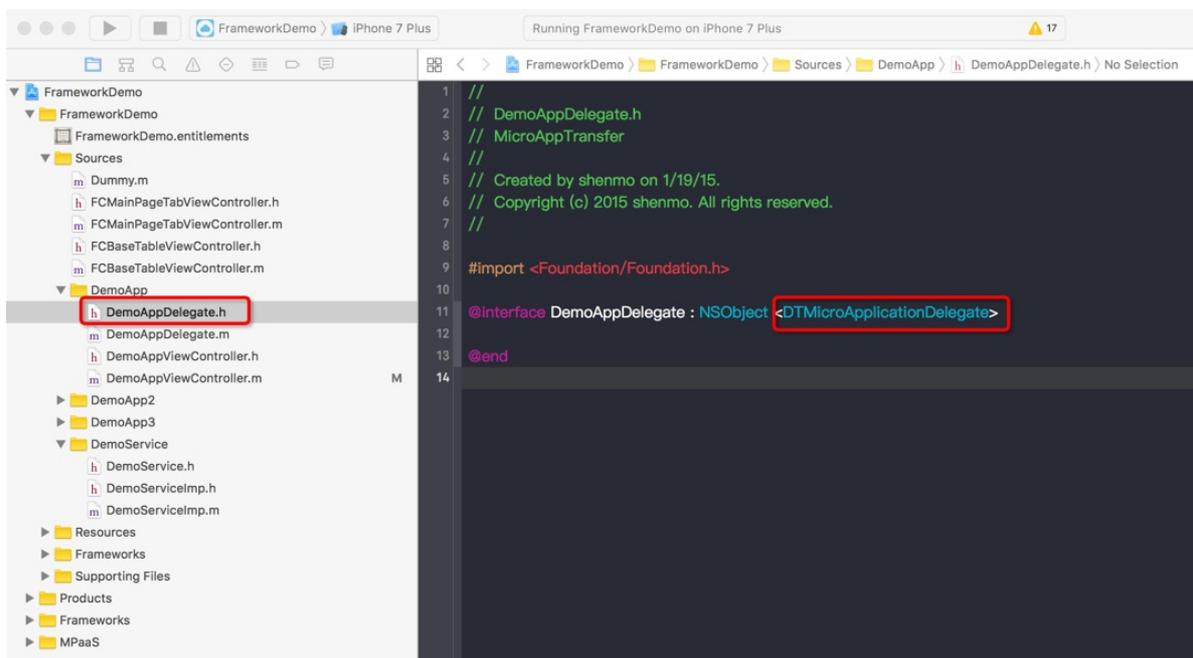
#### Sample code

Visit [iOS framework-demo](#) to download the sample code of the iOS mobile framework.

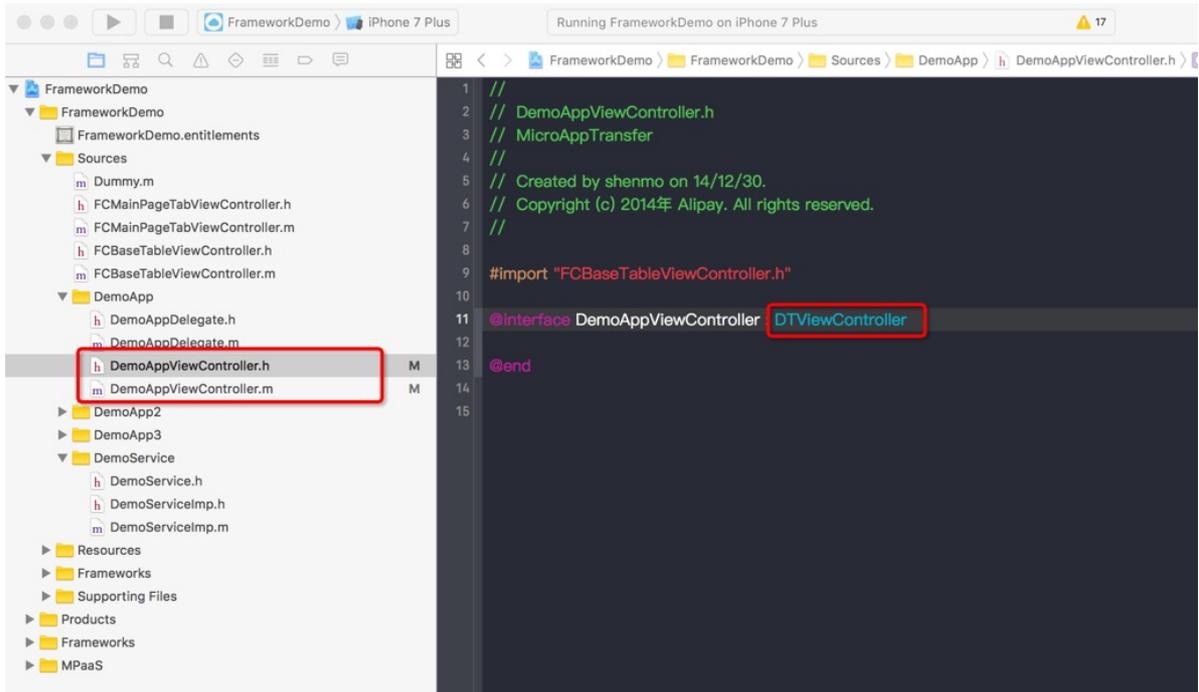
#### Procedure

##### 1. Add micro-application template code.

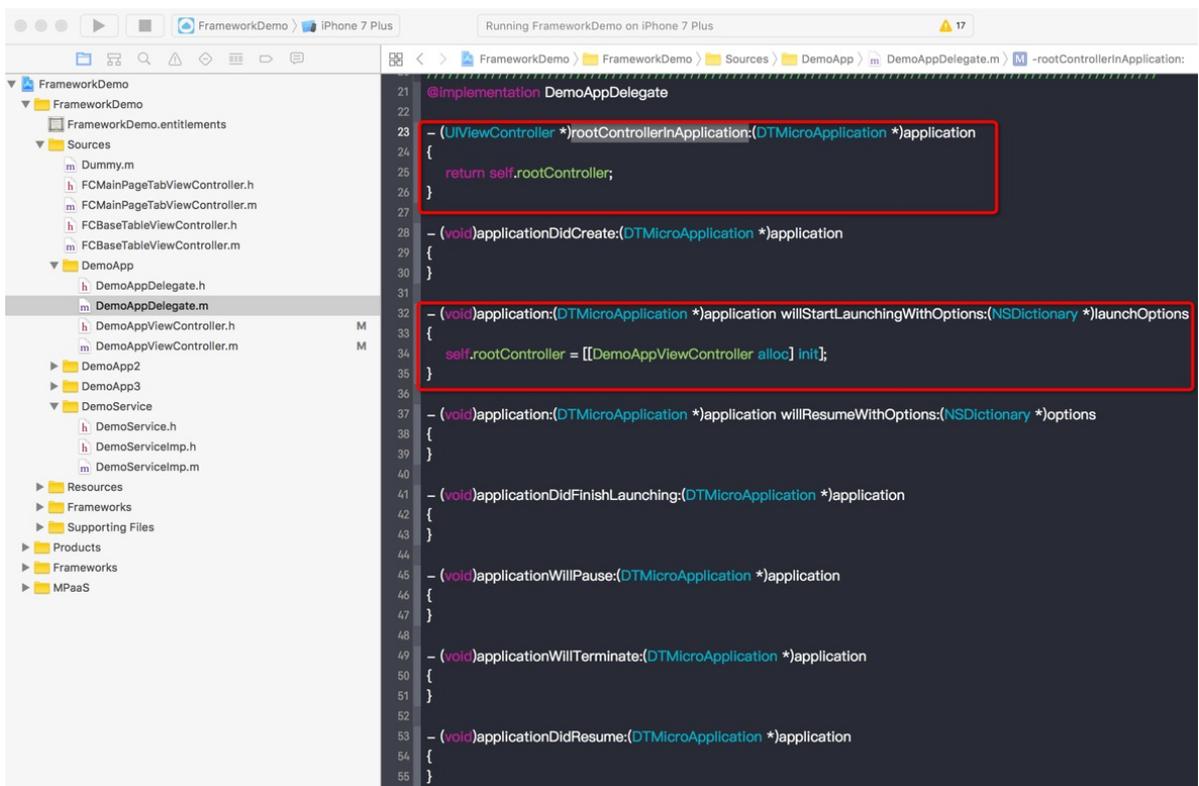
1. Create a micro-application delegation class and implement the delegation method of the micro-application manager of the mPaaS iOS framework, `DTMicroApplicationDelegate`.



2. Create `rootViewController` for the micro application, which can inherit the `DTViewController` base class provided by the mPaaS iOS framework.

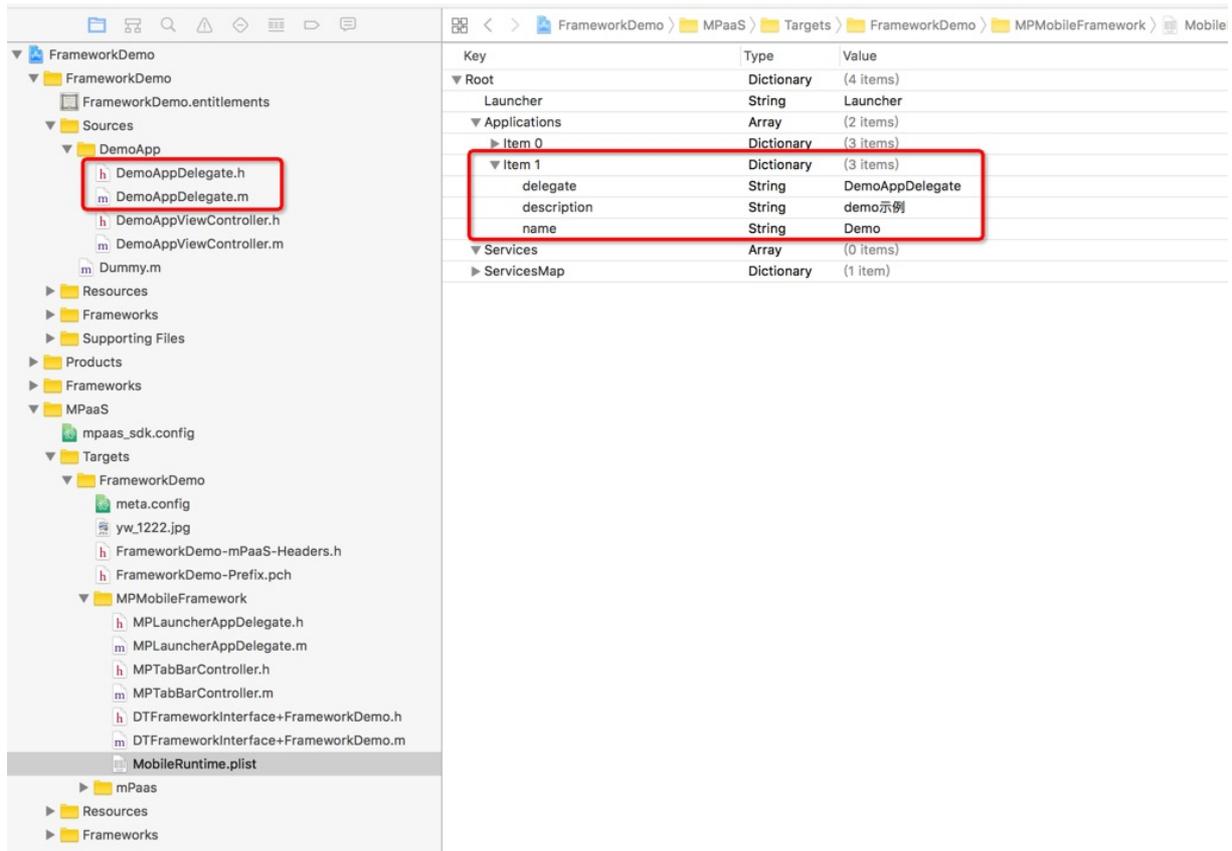


- Specify `rootViewcontroller` for the micro application. You can perform service actions in the lifecycle of the micro application by using its delegation method.



## 2. Register the micro application.

You can manage the created micro application with the framework only after registering the micro application in `MobileRuntime.plist`.



| Field       | Description   |
|-------------|---|
| Delegate    | The class name of <code>DTMicroApplicationDelegate</code> for application implementation. |
| Description | The description of the application.   |
| Name        | The name of the application. The framework context finds the application by this name.    |

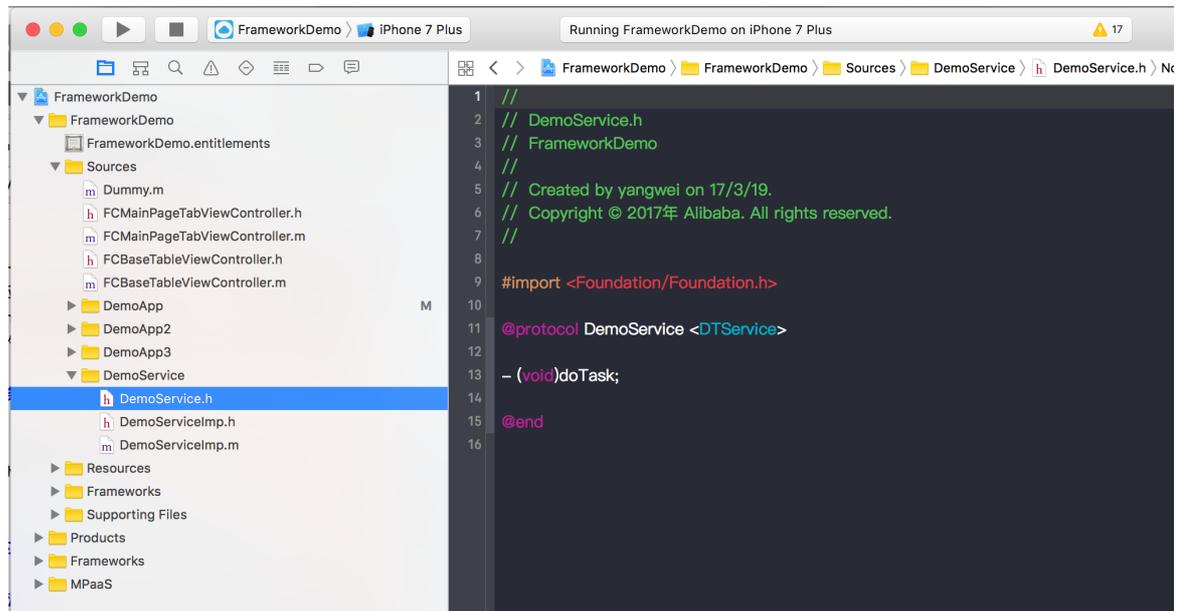
### 4.3.2. Create a service

In the process of developing an application based on mPaaS iOS framework, you can set the general functions without UI as a service (e.g. login) which can be easily obtained by other micro applications or services through the whole App running period. To add a service, you must add service template codes, and register a service.

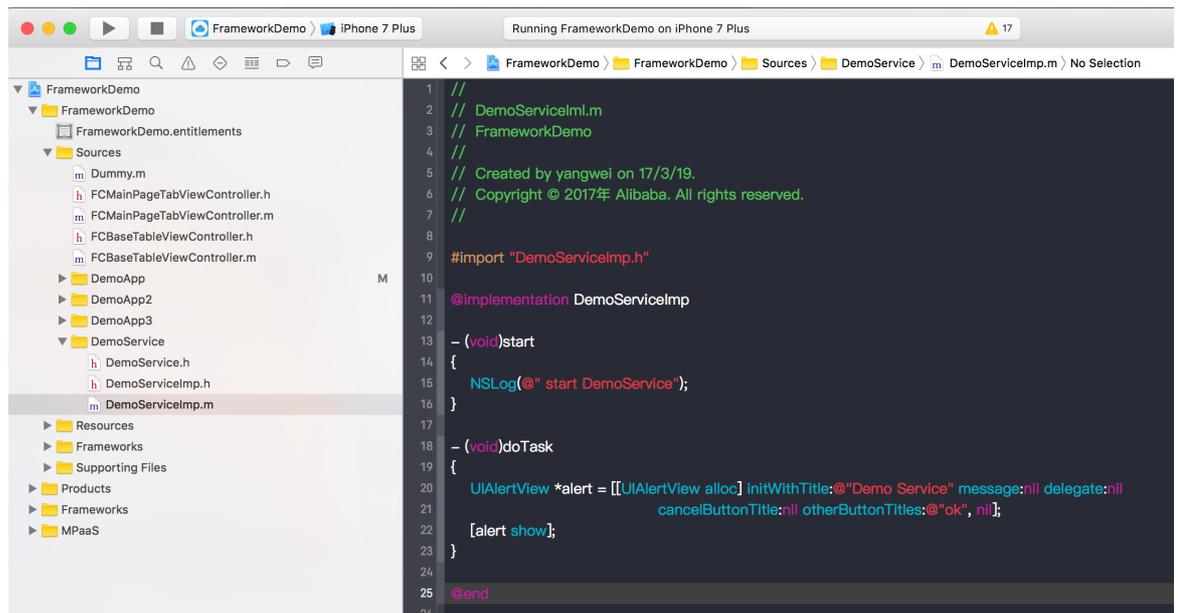
#### Procedure

1. Add service template codes.

i. **Define the service's protocol and expose the external interface method.**

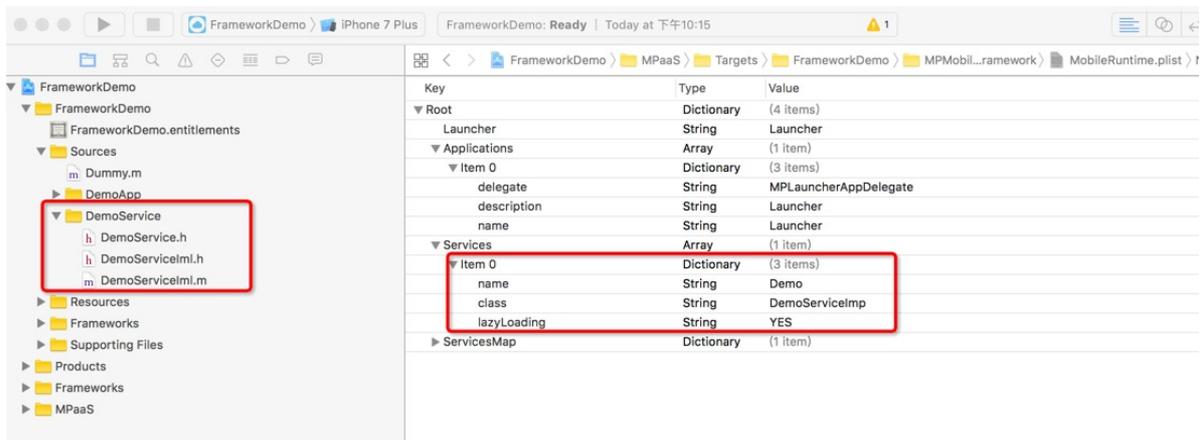


ii. **Define the class to implement the service interface method.**



## 2. Register a service.

**Similar to the micro application, the newly created services can be managed in a unified way via the framework only when they are registered in `MobileRuntime.plist` .**



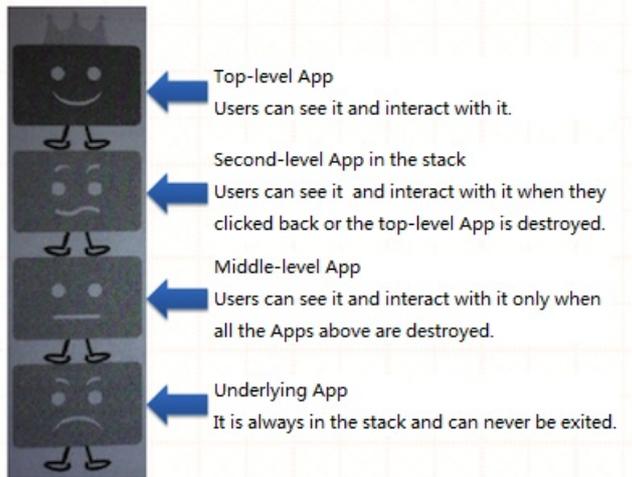
| Field       | Description   |
|-------------|---|
| Name        | The unique identifier of a service.   |
| Class       | The implementation class of service. When creating a service, the framework utilizes a run-time reflection mechanism to create the instances of the service implementation class.   |
| lazyLoading | Whether to delay loading. If Yes, the service will not be instantiated when the framework starts. Only when being used can the service be instantiated and launched. If No, the service will be instantiated and launched when the framework starts. It defaults to No. |

### 4.3.3. Manage the micro application and service

After dividing the business into micro application and service, you will not only achieve high cohesion and low coupling among modules, but also can manage micro application and service by virtue of the context provided by mPaaS iOS framework, including micro application-to-micro application, service-to-service and micro application-to-service jumping and data transmission.

#### Manage micro applications

Framework context manages the jumps of all micro applications in a unified way through stack, and in compliance with the following rules:



- Based on mPaaS iOS framework, you can quickly find a micro application by using its `name` and start another micro application in the current micro application.

```
- (void)pushSubApp2
{
    [DTContextGet() startApplication:@"20000002" params:@{
    launchMode:kDTMicroApplicationLaunchModePushWithAnimation};
}
```

- The upper-level micro application in the stack can quickly jump to the root application at the bottom.

```
- (void)exitToLauncher
{
    //Since Launcher is at lower-level, starting Launcher means exiting from all up
    per-level applications and back to Launcher
    [DTContextGet() startApplication:@"Launcher" params:nil
    animated:kDTMicroApplicationLaunchModePushNoAnimation];
}
```

- Quickly exit from the current micro application.

```
- (void)exitSelf
{
    [[DTContextGet() currentApplication] exitAnimated:YES];
}
```

- Quickly exit from the started micro application.

```
- (void)exitApp2
{
    // If current top-level application is app3, it is workable to force app2 and i
    ts windows to exit.
    [DTContextGet() findApplicationByName:@"20000002"] forceExit];
}
```

## Manage services

- Based on mPaaS iOS framework, you can quickly start another service in the current micro application.

```
- (void) findService
{
    id<DemoService> service = [DTContextGet() findServiceByName:@"DemoService"];
    [service doTask];
}
```

## 4.3.4. Code sample of micro application

The Demo introduces the hierarchical relations among mPaaS micro applications. For more information about iOS framework, see [mPaaS iOS framework](#).

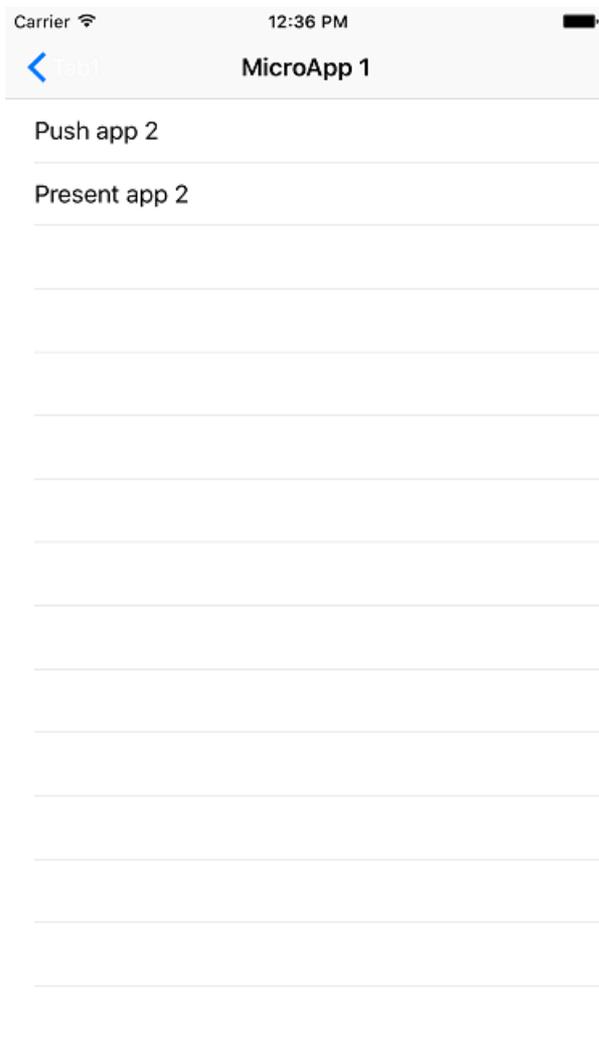
### Download codes

Please refer to [Get code sample](#) and download the Demo codes locally.

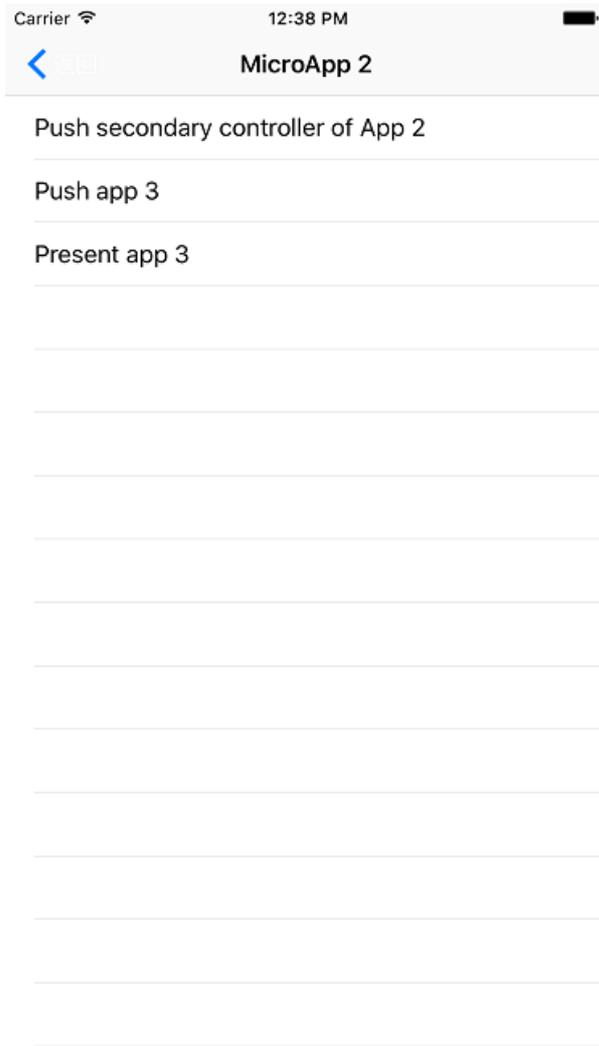
```
git clone git@git.cloud.alipay.com:mPaaS-Demos/FrameworkDemo.git
```

### Micro application hierarchy demonstration

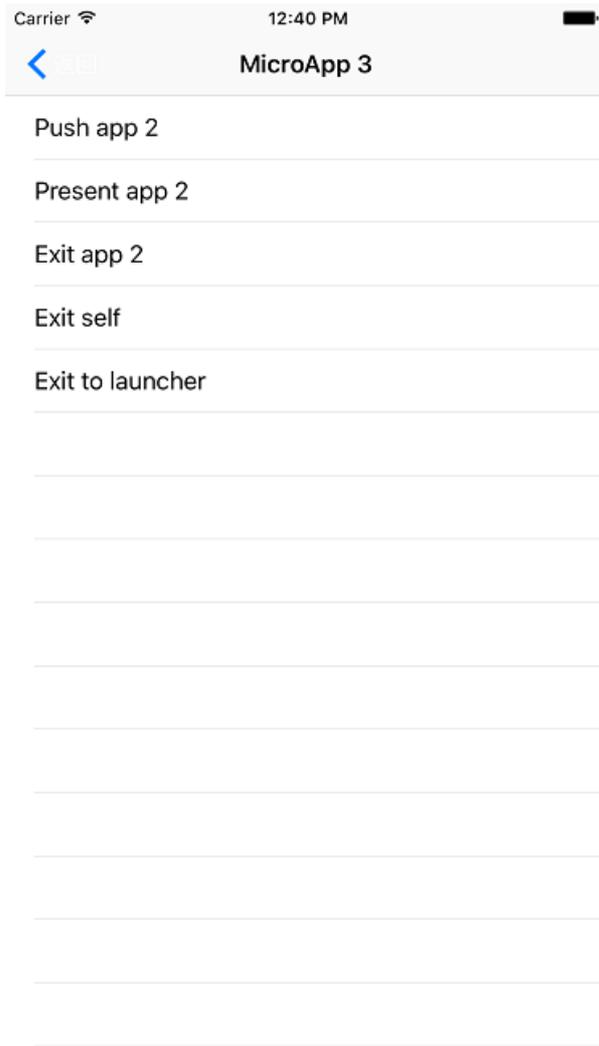
- Click **Push app** to start MicroApp 1;



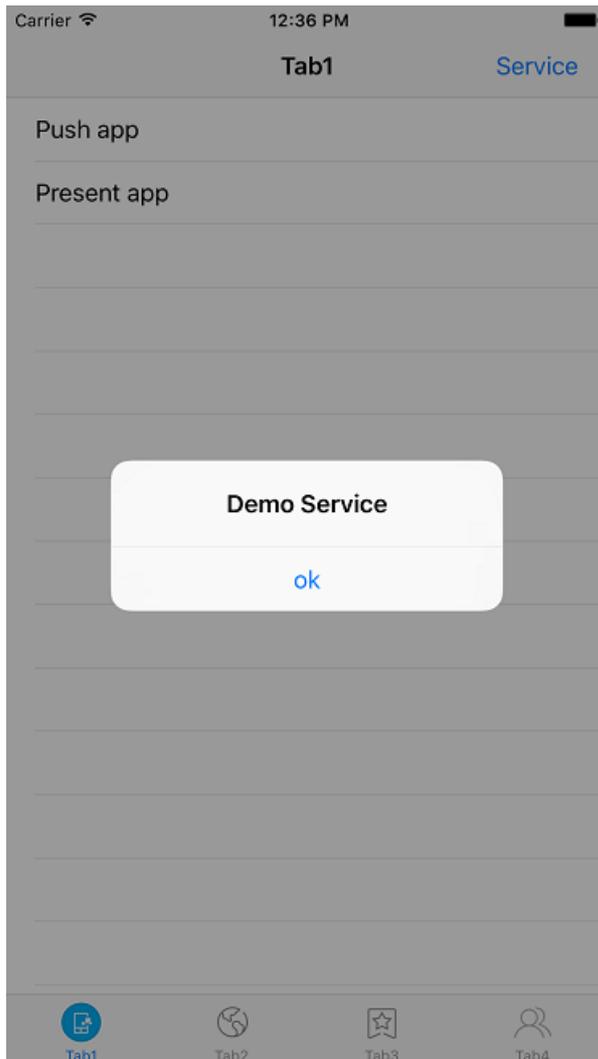
- Click **Push app 2** to start MicroApp 2;



- Click **Push app 3** to start MicroApp 3;



- If app 3 is on the top, click **Exit app 2** to force app 2 and its windows to exit;
- If app 3 is on the top, click **Exit self** to exit app 3 itself and back to app 2;
- If app 3 is on the top, click **Exit to launcher** to back to the root application and force app 1, app2, and app 3 to exit;
- Meanwhile, you can start a service in the micro application of **Launcher**.



## 4.4. iOS language settings

It will introduce the method of setting language in the process of integrating mPaaS into iOS client.

When integrating with an iOS project, you can configure language settings for your iOS app.

### Use the system language by default

1. You can add [Languages.bundle.zip](#) to your project to configure languages supported by the current app.
2. When the app is successfully launched, initialize the multilingual framework

```
//#import <mPaas/APLanguage.h>.  
[APLanguageSetting sharedSetting];
```

### Obtain the current app language

You can obtain the current app language by using the following command:

```
NSString *currentLanguage = [APLanguageSetting currentLanguage].name;
```

## Modify the current app language

In the `Languages.bundle` file of the project, you can view the language currently supported by the app : You can modify the current app language by using the following command:

```
[APLanguageSetting setCurrentLanguageWithName:@"en"];
```

## Support multilingual copywriting

1. Add multilingual bundle files.

- i. Add the strings files corresponding to the languages currently supported by the app.
- ii. Set the path that stores the multilingual files:

```
[[APLanguageBundleLoader sharedLoader] setCustomLanguagesBundlePath:@""];
```

2. Define text strings in strings files.

The implementation principle of strings files is as follows:

- The format of each text string in a strings file is as follows: The left side of the equal sign is the key of a text string while the expression on the right side is the content to be displayed in the corresponding language.

For example : `"BeeCityPicker : City Choice" = "City Choice"` .

- For the same text string, its key must be the same in all strings file. We recommend that you use a combination of the bundle name and the text string in Chinese as the key.

For example : `"BeeCityPicker : City Choice"` .

3. Configure the text strings.

For text strings that need to be provided in multiple languages, do not use hardcoded expressions. You can use the `__Text` macro. For example:

```
self.navigationItem.title = __TEXT(@"BeeCityPicker",@"BeeCityPicker:City Selection",  
@"City Selection");
```

- `@"BeeCityPicker"` : The `bundle` name of a text string in a strings file. Generally, it is the name of a module resource bundle.
- `@"BeeCityPicker:City selection"` : The key of a text string in a strings file.
- `@"City selection"` : The content to be returned by default when the text string corresponding to a specified key cannot be found in the strings files.

## 4.5. Customize the city selection

This article introduces the method of user-defined city selection in the process of integrating mPaaS into iOS client.

When integrating with an iOS project, you can customize the city selection.

### Note

This function is valid only in the baseline versions 10.1.68.27 and later.

## Customize the city file

### All cities

1. Create a city file with an extension of `.txt` . The file content format is as follows:
  - Field 1: adcode.
  - Field 2: City name.
  - Field 3: City name in Chinese Pinyin. This field is used to configure the first letter on the right.
2. Set the path that stores the custom city file. The custom city file is saved under the path of the bundle file, for example, `BeeCityPicker.bundle/citiesWithCounty.txt` . The SDK will automatically read the file name:

```
[BeeCityPickerAdapter sharedInstance].customHotCityTextFile =  
@"BeeCityPicker.bundle/citiesWithCounty.txt";
```

## Popular cities

1. Create a popular city file. The file content is same as the city file created in [All cities](#).
2. Set the path that stores the custom popular city file.

```
[BeeCityPickerAdapter sharedInstance].customHotCityTextFile =  
@"BeeCityPicker.bundle/hotCities.txt";
```

## Customize cities in a mini program

For details about how to customize cities in a mini program, see [Select cities](#).

# 5. Adaption to iOS

## 5.1. mPaaS 10.2.3 Support Wireless Security Guard & Blue Shield Switch

### Background information

The matching of wireless security guard client SDK and wireless security guard pictures is one of the basic dependency capabilities of mPaaS and is widely used in mPaaS products. In order to further improve the compatibility of mPaaS products in various scenarios and meet higher compliance requirements, mPaaS provides Blue Shield capability as an alternative to wireless security guard capability to support scenarios that wireless security guard cannot meet.

### Current situation

At present, mPaaS has completed the adaptation and testing of supporting wireless security guard switching Blue Shield in the baseline version of iOS 10.2.3.25 and above. If you use the baseline of 10.1.68 or earlier, please upgrade to the latest version of 10.2.3.

### Upgrade Baseline

Upgrade the baseline version to **10.2.3.25** or later.

### Current baseline is 10.1.68 primary baseline

Please refer to [mPaaS 10.2.3 adaptation Xcode14](#) to upgrade to the latest baseline of 10.2.3 and make relevant adaptation.

### The current baseline is a custom baseline

If you are using a custom baseline, search for the group number 41708565 to join the DingTalk group or submit a ticket to consult the corresponding after-sales and technical helpdesk.

### Update method

#### CocoaPods-based upgrade

Follow these steps to install the latest SDK of the **10.2.3** version:

1. First, make sure that the version number of the mPaaS component in the Podfile is **10.2.3**.
2. Execute the `pod mpaas update 10.2.3` .  
If a command error is reported, run the `pod mpaas update --all` command to update the plug-in and then run the command again.
3. Execute the `pod install` .

### Upgrade Toolchain & Switch Blue Shield

#### Use mPaaS CocoaPods Plugin

Run the following command to install or upgrade the plug-in:

```
sh <(curl -s http://mpaas-ios.oss-cn-hangzhou.aliyuncs.com/cocoapods/installmPaaS-CocoaPodsPlugin.sh)
```

Run the following command to view the version of the plug-in:

```
pod plugins installed
```

Installed CocoaPods Plugins:

```
cocoapods-deintegrate : 1.0.5
cocoapods-mPaaS : 0.9.6 (pre_install and post_install hooks)
cocoapods-plugins : 1.0.0
cocoapods-search : 1.0.1
cocoapods-trunk : 1.6.0
cocoapods-try : 1.2.0
```

The 0.9.6 in the preceding figure shows the version of the plug-in. For more information, see Use CocoaPods to access the plug-in based on an existing project.

### Remove the wireless security guard component

1. Add `remove_pod "mPaaS_Security"` to the podfile.
2. If there are `mPaaS_pod "mPaaS_Security"` in the podfile, remove them.
3. To remove the wireless security guard image, you must first delete the `yw_1222.jpg` image from the project and remove the `base64Code` and `v6Base64Code` fields (if present) from the `mPaaS` configuration file `meta.config`.

### Add a Blue Shield component

Add `mPaaS_pod "mPaaS_BlueShield"` to the podfile.

### Generate a Blue Shield image

Add the `absBase64Code` parameter in the `meta.config` file, and perform pod install operations according to the mPaaS cocoaPods Plugin. As shown in the figure, `abs_1222` is the generated blue shield image.



### Code Switching Blue Shield

Add the `MPSignatureInterface` classification and rewrite the `securityComponentType` method to return the `MPSecurityComponentTypeBS`. The `#import` `<MPSignatureAdapter/MPSignatureAdapter.h>` header file needs to be imported in the `MPSignatureInterface` classification.

```
//  
// MPSignatureInterface+demopod.m  
// MPRPCDemo_pod  
//  
// Created by yanjinqun on 6/8/23.  
// Copyright © 2023 mPaaS. All rights reserved.  
//  
  
#import "MPSignatureInterface+demopod.h"  
  
@implementation MPSignatureInterface (demopod)  
- (MPSecurityComponentType)securityComponentType {  
    return MPSecurityComponentTypeBS;  
}  
@end
```

## List of libraries that support Blue Shield updates

- Mobile Gateway Service
- mobile dispatching center
- Data synchronization
- Multimedia
- Mini program
- Hotpatch
- Big tunnel
- Some internal dependent components

## Scope of test verification

After the Blue Shield switch is completed, perform a regression test on the app based on the preceding change list.

# 5.2. mPaaS 10.2.3 Adaptation to Xcode15

## Background information

Apple requires that all apps submitted to the App Store must be built using Xcode 15 or later from April 29, 2024. For details, please refer to [Apple's official statement](#). Xcode 15 no longer supports the construction of systems before iOS 12. For details, please refer to [Xcode 15 Release Notes](#).

Apple announced that starting from May 1, 2024, if the app code uses the API in the [API list](#) listed by Apple, the relevant reasons need to be stated in the app's privacy list. For example, if the app code uses the SDK in the [three-party SDK list](#) listed by Apple, the three-party SDK needs to provide corresponding privacy list configuration before uploading a new app or a new version of the app to the App Store Connect. For details, please refer to [Apple's official statement](#).

## Current status

Currently, mPaaS has completed the adaptation and testing of Xcode 15 in baseline versions 10.2.3.40 and later. If you use baseline 10.1.68 (obsolete) or earlier, upgrade to the latest version 10.2.3 and no longer support systems earlier than iOS 12.

## Upgrade Baseline

Select an upgrade method based on your current baseline version.

### Current baseline is 10.2.3 primary baseline

Please refer to [mPaaS 10.2.3 Usage and Adaption to iOS 17](#) to upgrade to the latest baseline of 10.2.3 and make relevant adaptation.

### The current baseline is a custom baseline

If you are using a custom baseline, search for the group number 41708565 to join the DingTalk group or submit a ticket to consult the corresponding after-sales and technical helpdesk.

## Upgrade modes

### CocoaPods-based upgrade

Follow these steps to install the latest SDK of the **10.2.3** version:

1. First, make sure that the version number of the mPaaS component in the Podfile is **10.2.3**.

2. Execute the `pod mpaas update 10.2.3`.

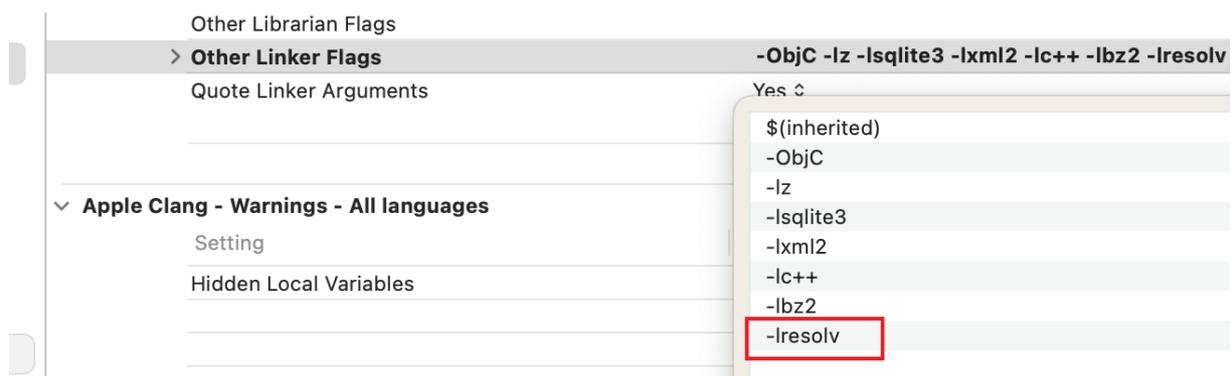
If a command error is reported, run the `pod mpaas update --all` command to update the plug-in and then run the command again.

3. Execute the `pod install`.

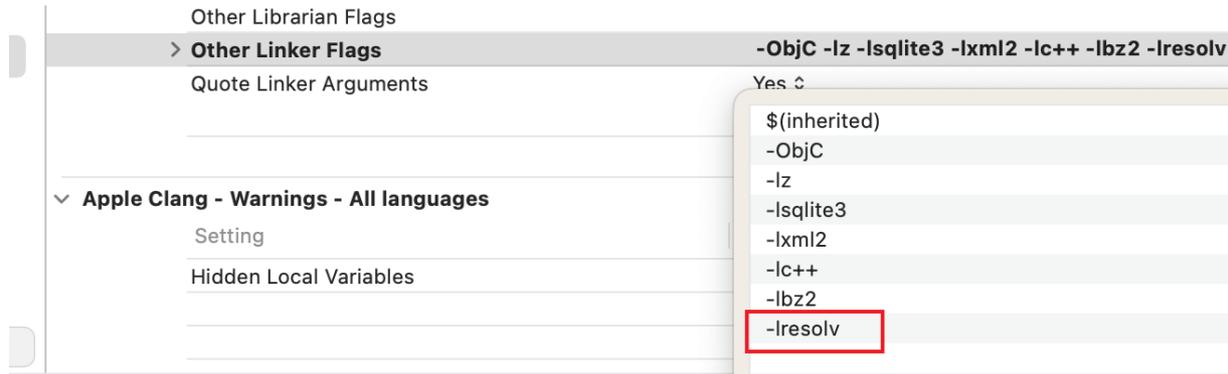
## Description

1. When you use the **Extension plug-in** to upgrade to the baseline version 10.2.3.30 or later, add modules such as threat awareness and national secret,

In the main project, you need to manually add the system library `resolv` to the Other link flag in Building Setting.



Otherwise, the following error occurs:



## API changes

There is no change in the interface usage for this Xcode15 adaptation.

## Library manifest adapted to Xcode 15 update

- Mobile gateways
- Mobile scheduling
- Data synchronization
- message push
- National Secret
- Multimedia
- Mini Program
- Security Keyboard
- Some internal dependent components

## Scope of test verification

As the upgrade of Apple's tool chain is a black box operation, it often brings stability and other problems. After the adaptation of App to Xcode 15 is completed, it is recommended to conduct a comprehensive regression test on App.

In addition, because the URL parsing of the installation package packaged with Xcode 15 is updated from RFC1738/1808 to RFC 3986 in iOS 17 and later, URL-related parsing may be abnormal. For example, if API operations such as `URLWithString` and `InitWithString` are used, please verify and adapt the URL. For more information, see [Apple official instructions](#).

# 5.3. mPaaS 10.2.3 Adaptation to Xcode14

## Background information

Apple requires that all apps submitted to the App Store must be built using Xcode 14.1 or later from April 25, 2023. Please refer to [Apple's official statement](#) for details. In addition, Xcode 14 no longer supports building iOS projects with ARMv7, ARMv7s, and i386 architectures, and no longer supports building systems before iOS 11. For more information, see [Xcode 14 release notes](#). For the new tool chain, App needs to be adapted.

## Current Status

Currently, mPaaS has completed the adaptation and testing of Xcode 14 in the baseline version 10.2.3.19 and later. If you use the baseline version 10.1.68 or earlier, upgrade to the latest version 10.2.3.

## Upgrade Baseline

Select an upgrade method based on your current baseline version.

### Current baseline is 10.1.68 primary baseline

Please refer to [mPaaS 10.2.3 Usage and Adaptation to iOS 16](#) to upgrade to the latest baseline of 10.2.3 and make relevant adaptation.

### The current baseline is a custom baseline

If you are using a custom baseline, search for the group number 41708565 to join the DingTalk group or submit a ticket to consult the corresponding after-sales and technical helpdesk.

## Upgrade mode

### CocoaPods-based upgrade

Follow these steps to install the latest SDK of the **10.2.3** version:

1. First, make sure that the version number of the mPaaS component in the Podfile is **10.2.3**.
2. Execute the `pod mpaas update 10.2.3` .  
If a command error is reported, run the `pod mpaas update --all` command to update the plug-in and then run the command again.
3. Execute the `pod install` .

## Description

At present, Mac machines that do not support M1 and M2 chips are run directly using the simulator. If you want to use the simulator, you can add relevant configuration items, set the `Excluded Architectures` in the test `Build Settings` , and add `arm64` .

## API changes

This Xcode14 adaptation has no changes in interface usage for the time being.. If you are using the 10.1.68 or related customized baseline, upgrade to the latest 10.2.3 baseline and make relevant adaptation by referring to [mPaaS 10.2.3 Usage and Adaptation to iOS 16](#).

## Library manifest adapted to Xcode 14 update

- Mobile gateways
- Mobile scheduling
- Data synchronization
- Message push
- SSL
- Multimedia
- Miniapp
- Mobile Security Keyboard
- Some internal dependent components
- Related components that do not involve code adaptation have been rebuilt using Xcode 14, and all libraries have been updated.

## Scope of test verification

As the upgrade of Apple's tool chain is a black box operation, it often brings stability and other problems. After the adaptation of App to Xcode 14 is completed, it is recommended to conduct a comprehensive regression test on App.

## 5.4. mPaaS 10.2.3 Adapts to iOS 18

This topic describes the adaptation of the mPaaS 10.2.3 baseline for iOS 18 and the adaptation work that users need to complete.

### Background information

iOS 18 will be officially released in September 2024. App needs to be adapted to the new system features and interfaces. At present, mPaaS has completed the adaptation and testing of iOS 18 beta(6) in the baseline version of 10.2.3.50 or later.

### Current status

As a basic library, mPaaS has completed the adaptation and testing of iOS 18 beta(6) under the ipa package built by Xcode 15. If you plan to launch your application in the Apple App Store, **use Xcode 15 to package the application**.

The related toolchain for Xcode 16 is being perfected. After the tool chain is perfected, mPaaS will also release a version adapted to iOS 18 under Xcode 16 construction.

### Upgrade baseline

Upgrade the baseline version to **10.2.3.50** or later.

### Current baseline is 10.1.68 primary baseline

Please refer to [mPaaS 10.2.3 adaptation Xcode15](#) to upgrade to the latest baseline of 10.2.3 and make relevant adaptation.

### Current baseline is a custom baseline

If you are using a custom baseline, search for the group number 41708565 to join the DingTalk group or submit a ticket to consult the after-sales and technical support staff to inquire whether it is possible to switch to the 10.2.3 baseline.

### Upgrade methods

#### Upgrade based on CocoaPods

Follow these steps to install the latest SDK of the **10.2.3** version:

1. First, make sure that the version number of the mPaaS component in the Podfile is **10.2.3**.
2. Execute the **pod mpaas update 10.2.3**. If a command error is reported, run the **pod mpaas update --all** command to update the plug-in and then run the command again.
3. Execute the **pod install**.

### API changes

The iOS18 Release Note does not show that certain API changes can cause adaptation problems.

### iOS 18 adaptation

iOS 18 will be officially released in September 2024. APP needs to be adapted to the new system features and interfaces.

- This baseline has been adapted and tested under the ipa package built by Xcode 15 based on version iOS 18 beta(6).
- If you plan to launch your application in the Apple App Store, **use Xcode 15 to package it**.

- The related toolchain for Xcode 16 is being perfected. After the tool chain is perfected, mPaaS will also release a version adapted to iOS 18 under Xcode 16 construction.

The following components are mainly involved in the adaptation point of iOS 18, including but not limited to the following components:

- Foundation framework;
- Scan code;
- Mini programs;
- Gateway;
- Offline packages;
- Some internal dependent components;

Apps that use the above components need to undergo complete testing and verification under iOS 18 to ensure functionality, compatibility, and stability.

## Processing of custom baselines

If you are using a custom baseline, you may need to re-customize it based on the new version. [Submit a ticket](#) or contact mPaaS technical support staff for confirmation.

# 5.5. Adapt mPaaS 10.2.3 to iOS 17

This topic describes the adaptation of the mPaaS 10.2.3 baseline to iOS 17 and the adaptation work that users need to complete.

## Background information

iOS 17 will be officially released in September 2023. App needs to be adapted to the new system features and interfaces. At present, mPaaS has completed the adaptation and testing of iOS 17 beta(7) in the baseline version of version 10.2.3.30 or later.

## Current status

As a basic library, mPaaS has completed the adaptation and testing of iOS 17 beta(7) under the ipa package built by Xcode 14. If you plan to launch your application in the Apple App Store, **use Xcode 14 to package the application**.

Xcode 15's related toolchain is being perfected. After the tool chain is perfected, mPaaS will also release a version adapted to iOS 17 under Xcode 15 construction.

## Upgrade Baseline

Upgrade the baseline version to **10.2.3.30** or later.

## Current baseline is 10.1.68 primary baseline

Please refer to [mPaaS 10.2.3 adaptation Xcode14](#) to upgrade to the latest baseline of 10.2.3 and make relevant adaptation.

## The current baseline is a custom baseline

If you are using a custom baseline, search for the group number 41708565 to join the DingTalk group or submit a ticket to consult the corresponding after-sales and technical support.

## Upgrade modes

### CocoaPods-based upgrade

Follow these steps to install the latest SDK of the **10.2.3** version:

1. First, make sure that the version number of the mPaaS component in the Podfile is **10.2.3**.

2. Execute the **pod mpaas update 10.2.3**. If a command error is reported, run the **pod mpaas update --all** command to update the plug-in and then run the command again.
3. Execute the **pod install**.

## API changes

The iOS 17 Release Note does not show that specific API changes can cause adaptation issues.

## iOS 17 adaptation

It was officially released in September 2023. APP needs to be adapted to the new system features and interfaces.

- This baseline has been adapted and tested under the ipa package built by Xcode 14 based on version iOS 17 beta(7).
- If you plan to launch your application in the Apple App Store, **use Xcode 14 to package it**.
- Xcode 15's related toolchain is being perfected. After the tool chain is perfected, mPaaS will also release a version adapted to iOS 17 under Xcode 15 construction.

The adaptation points of iOS 17 mainly involve audio encoding and decoding, privacy, including calendar and album permission function changes, and related components involved are as follows, including but not limited to the following components:

- Basic frameworks
- Audio and Video
- Scan
- Mini program management
- Share
- Some internal dependent components

Apps that use the above components need to undergo complete testing and verification under iOS 17 to ensure functionality, compatibility, and stability.

## Handling of custom baselines

If you are using a custom baseline, you may need to re-customize it based on the new version. [Submit a ticket](#) or contact mPaaS technical support for confirmation.

# 5.6. Use mPaaS 10.2.3 and adapt mPaaS 10.2.3 to iOS 16

## Baseline description

This baseline is based on the latest product set of Mobile PaaS (mPaaS) 10.1.68.53. Before you merge the baseline, you need to integrate the modifications, new components, compliance enhancement, and other capabilities of various custom baselines. The following features are involved:

## Security compliance

Security Guard SDK provides enhanced security capabilities and supports the digital signatures using Chinese national cryptography.

## New components

- mPaaS GMSSL (private cloud)

- Mobile Dispatch Center (MDC)
- Application Performance Management (APM)
- Face verification (private cloud)
- Mobile terminal environment detection (private cloud)
- Mobile Threat Awareness Platform or device risk consulting (private cloud)
- Intelligent Dual-Recording System

## Update components

- OpenSSL is fully upgraded to solve conflicts with open source libraries.
- AlipaySDK is upgraded for the adaptation to iOS 16.

## Upgrade the baseline

You can select an upgrade method based on your current baseline version.

### Current baseline: 10.1.68 primary baseline

#### Upgrade based on CocoaPods

To install the latest SDKs of the baseline version 10.2.3, perform the following steps:

1. Make sure that the mPaaS version is 10.2.3 in Podfile.
2. Run the `pod mpaas update 10.2.3` command.

If an error occurs, you need to run the `pod mpaas update --all` command to update the plug-in, and then rerun the command.

3. Run the `pod install --repo-update` command.

### Current baseline: a custom baseline other than 10.1.68

If you do not know whether to switch to the 10.2.3 baseline, you can search for group number 41708565 to join the DingTalk group and consult mPaaS Customer Service.

## Update configurations

### Configure the version of Security Guard images to V6

In the 10.2.3 baseline, the Security Guard SDK has been upgraded to support digital signatures using Chinese national cryptography. Therefore, after the upgrade, you need to replace the version of Security Guard images with the V6 version. When you use the 10.1.68 baseline, the version of Security Guard images is V5 by default. To use a plug-in to generate the image in the V6 version and replace the original Security Guard image file, that is, yw\_1222.jpg, perform the following steps:

To configure the Security Guard image, [submit a ticket](#) or contact mPaaS Customer Service.

## Hotpatch update

If you want to integrate the hotpatch feature into the current project, you can consult mPaaS Customer Service to upgrade the Hotpatch SDK.

## Security protection configuration

When you develop an app based on the mPaaS framework, the app that uses the HTML5 Container or Mini Program SDK calls JavaScript API (JSAPI) more often. In order to prevent app abuse when loading external pages, we recommend that you verify the URL when you open the URL and call the JSAPI. In this case, you need to make sure that only the URL within an app can be loaded to open or restrict JSAPI scopes and permissions of the URL. This can avoid the problems when authentication is not performed.

For information about specific authentication methods, see the following documents:

1. [Set JSAPI authentication.](#)
2. [Set the URL whitelist.](#)

## API changes

Changes of API usage are not involved in this adaptation.

## Adapt to iOS 16

Apple released iOS 16 in September 2022. The apps must be adapted to new system features and APIs of iOS16.

- mPaaS 10.2.3 baseline has completed the adaptation and testing for iOS 16 beta 8 under the .ipa packages built by Xcode 13.
- If you plan to publish your apps on Apple App Store, you need to **use Xcode 13 for packaging**.
- The toolchains for Xcode 14 are being improved. After the toolchains are improved, mPaaS will release a version adapted to iOS 16 under the .ipa packages built by Xcode 14.

The adaptations for iOS 16 mainly involve updating screen orientation methods, using the clipboard, and using pictures. The following components are involved:

- HTML5 Container and offline packages
- Mini Program
- Multimedia
- Face verification
- Mobile Security Keyboard (MSK)
- Basic framework
- Event tracking monitoring
- Mobile Real-Time Communication (MRTC)
- Optical Character Recognition (OCR)
- Scan
- Map
- Some internal dependent components

For the apps that use the preceding components, you need to perform the complete test and verification under iOS 16 to ensure functionality, compatibility, and stability.

## 5.7. Adapt to WKWebView

WKWebView is the new-generation built-in browser component in iOS 8 introduced by Apple. It is designed to replace the outdated UIWebView component. This component features multi-process rendering, including page scrolling without affecting image resource loading, crash protection without affecting the main process, and reduced memory usage. Compared with UIWebView, WKWebView has greatly improved its performance, stability, and user experience. After several years of iteration (from iOS 8 to iOS 9, iOS 10, iOS 11, and iOS 12), WKWebView has gradually solved all the problems that occurred in the early stage of its launch and improved its stability.

After iOS 12 was released, Apple APIs began to prompt users to gradually discard the UIWebView API. Since August 2019, when developers submit their apps with the UIWebView component to App Store for approval, they are prompted with the following warning to remind them to switch to WKWebView soon.

Additionally, Apple announced on December 23, 2019 that App Store will no longer accept new apps developed by using UIWebView from April 2020, and will no longer accept updates to existing apps developed by using UIWebView from December 2020.

In response to this situation, mPaaS has been adapted to WKWebView and supports switching from UIWebView to WKWebView. In order to ensure the stability of HTML5 pages after the switchover from UIWebView to WKWebView, the adaptation process of mPaaS to WKWebView is divided into the following two stages:

- Stage 1: Since November 2019, the mPaaS baseline supports the coexistence of UIWebView and WKWebView, and gradually switches to WKWebView through canary deployment capabilities.
- Stage 2: Since March 2020, the mPaaS baseline deletes all UIWebView-related code, and all HTML5 services are switched to WKWebView.

The mPaaS 10.1.60 baseline has completed stage 1 of adaptation. For users who integrate mPaaS HTML5 container and mini program components, they need to upgrade to the latest 10.1.60 baseline as soon as possible by observing the following instructions ([Upgrade the baseline](#)), and to switch to WKWebView ([Use WKWebView](#)).

## Upgrade the baseline

Based on app release situations, users who integrate mPaaS HTML5 container and mini program components need to take actions based on the following principles and upgrade the baseline to adapt to WKWebView.

- For apps launched in App Store before April 2020: To ensure the stability of switching your existing services to WKWebView, we recommend that you upgrade to the 10.1.60 baseline to support online canary deployment and rollback. For upgrade instructions, see [Instructions for upgrading to version 10.1.60](#).
- For new apps that are not yet launched in App Store before April 2020: Since App Store will no longer accept new apps with UIWebView after April, you must use version 10.1.68 without UIWebView-related code and be prepared for service regression testing and verification. For upgrade instructions, see [Instructions for upgrading to mPaaS 10.1.68-Beta](#).

## Use WKWebView

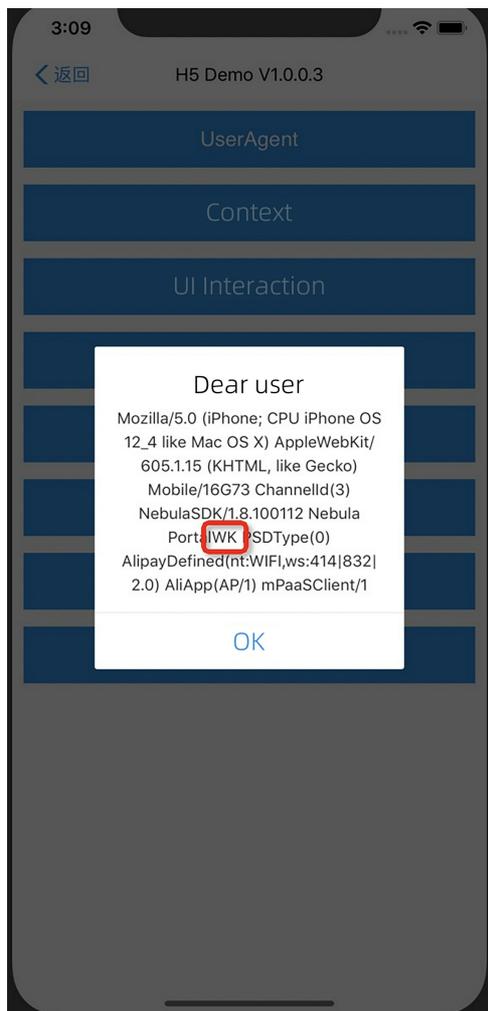
mPaaS containers use UIWebView to load HTML5 pages by default. The mPaaS framework supports two WKWebView enablement methods: **global enablement** and **canary deployment enablement**.

### The 10.1.60 baseline

In the 10.1.60 baseline, UIWebView and WKWebView coexist in mPaaS containers, where UIWebView is used to load HTML5 pages by default. You can globally enable WKWebView in the following way to make all pages that are loaded by mPaaS containers use WKWebView.

```
- (void)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    //...
    // Globally enable WKWebView
    [MPNebulaAdapterInterface sharedInstance].nebulaUseWKArbitrary = YES;
    //...
}
```

After you enable WKWebView, you can view the UA of the current HTML5 page. If the UA contains the `WK` string shown in the following figure, this page has been successfully switched to use WKWebView.



## Other instructions

After WKWebView is globally enabled, in order to ensure the functional stability of online HTML5 pages, the mPaaS framework provides **online stop-loss** capabilities to help you quickly switch WKWebView to UIWebView. The procedure is as follows.

Add a [configuration switch](#) to the real-time publishing component to prevent it from using UIWebView for offline packages or URLs.

The **Key** (key) of the configuration switch is `h5_wkArbitrary`, and the **Value** (value) is as follows:

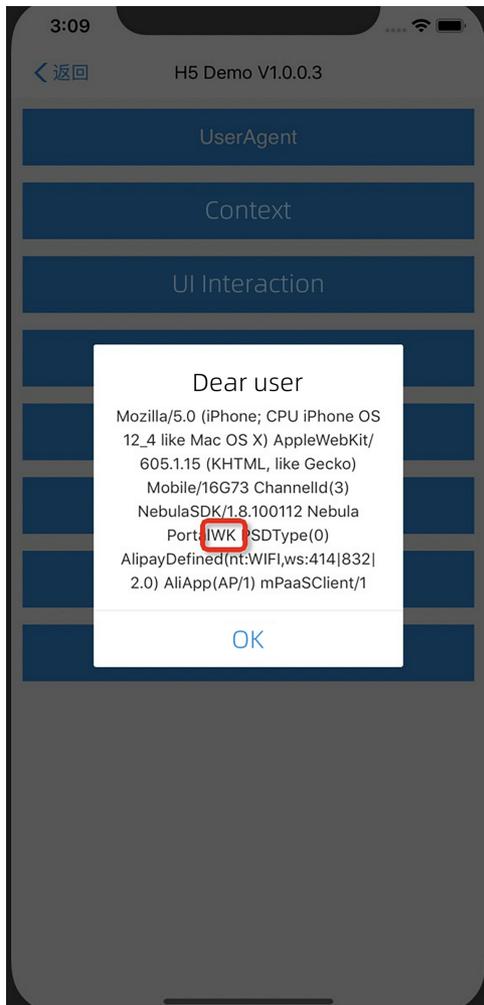
```
{
  "enable": true,
  "enableSubView": false,
  "exception": [
    {
      "appId": "^(70000000|20000193)$"
    },
    {
      "url": "https://invoice[.]starbucks[.]com[.]cn/"
    },
    {
      "url": "https://front[.]verystar[.]cn/starbucks/alipay-invoice"
    }
  ]
}
```

The following table describes the value configuration items.

| Configuration item |       | Description  | Remarks   |
|--------------------|-------|--|---|
| enable             |       | Whether to enable WKWebView. <code>true</code> for yes and <code>false</code> for no.  | The default is <code>false</code> .   |
| enableSubView      |       | Whether to enable WKWebView for the embedded webView in the mini program. <code>true</code> for yes and <code>false</code> for no. | The default is <code>false</code> .   |
| exception          | appId | WKWebView is not used for all HTML5 pages that match the appId regular expression in offline packages.                             | The default is <code>nil</code> . This field is valid only when <code>enable</code> is set to <code>true</code> . |
|                    | url   | WKWebView is not used for all HTML5 pages that match the url regular expression.   |   |

## The 10.1.68-beta baseline

Containers in version 10.1.68-beta use WKWebView to load offline packages and mini programs by default. You can view the UA of the current HTML5 page. If the UA contains the `WK` string shown in the following figure, the current page has been successfully switched to use WKWebView.



## 5.8. Privacy permission

### Background

The regulatory authority requires that the app cannot call sensitive relevant APIs before the user clicks the "Agree" button in the privacy agreement window. In order to meet this regulatory requirement, the baseline for mPaaS iOS 10.1.60.27 and later (60-series versions) and 10.1.32.18 and later (32-series versions) support this feature. You can modify your project as needed by referring to this document.

### Usage

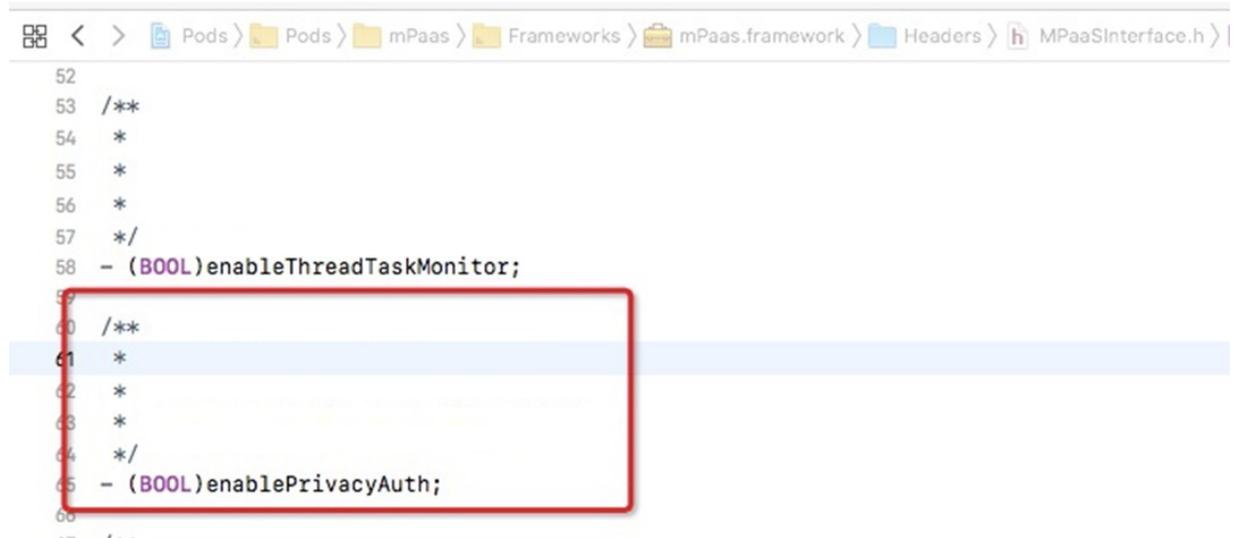
Different usage methods are required depending on whether the mPaaS iOS framework is allowed to host the lifecycle of the app. By checking whether `DFApplication` and `DFClientDelegate` are enabled for the framework in the `main.m` file of the project, you can determine whether the mPaaS iOS framework is allowed to host the lifecycle of the app. If `DFApplication` and `DFClientDelegate` are enabled, the hosting is allowed.

```
return UIApplicationMain(argc, argv, @"DFApplication", @"DFClientDelegate"); // NOW USE
MPAAS FRAMEWORK
```

## Host the lifecycle of the app by the framework

### 1. Allow privacy pop-up prompts.

In `MPaaSInterface` category , rewrite the `enablePrivacyAuth` API method and return `YES` .



```
52
53 /**
54 *
55 *
56 *
57 */
58 - (BOOL)enableThreadTaskMonitor;
59
60 /**
61 *
62 *
63 *
64 */
65 - (BOOL)enablePrivacyAuth;
66
--
--
```

**\*\*Sample code\*\***

```
``objectivec
@implementation MPaaSInterface (Portal)

- (BOOL)enablePrivacyAuth
{
return YES;
}

@end
````
```

### 2. Implement permission pop-up windows.

Rewrite the `- (DTFrameworkCallbackResult)application:(UIApplication *)application privacyAuthDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions completionHandler:(**void** (^)(**void**))completionHandler;` method provided by the framework.



```
< > Pods > Pods > A...k > F...s > A...k > H...s > DTFrameworkInterface.h > M -application:privacyAuthDidFinishLaunchingWithOptions:completionHandler: <
316
317 /**
318 *
319 *
320 *
321 *
322 */
323 - (DTFrameworkCallbackResult)application:(UIApplication *)application privacyAuthDidFinishLaunchingWithOptions:(NSDictionary
*)launchOptions completionHandler:(void (^)(void))completionHandler;
324
--
--
```

#### Sample code

```
`` `objectivec
- (DTFrameworkCallbackResult)application:(UIApplication *)application
privacyAuthDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
completionHandler:(void (^)(void))completionHandler
{
    UIWindow *authWindow = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    authWindow.backgroundColor = [UIColor redColor];
    authWindow.windowLevel = UIWindowLevelStatusBar+5;
    AuthViewController *vc = [[AuthViewController alloc] init];
    vc.completionHandler = completionHandler;
    vc.window = authWindow;
    authWindow.rootViewController = vc;
    [authWindow makeKeyAndVisible];

    return DTFrameworkCallbackResultContinue;
}
`` `
```

### 3. Start the mPaaS framework.

After the user clicks **Agree** for authorization, call back `completionHandler` to continue to start the mPaaS framework. The sample code is as follows:

```
#import <UIKit/UIKit.h>

NS_ASSUME_NONNULL_BEGIN

@interface AuthViewController : UIViewController

@property (nonatomic, copy) void (^completionHandler)(void);
@property (nonatomic, strong) UIWindow *window;

@end

NS_ASSUME_NONNULL_END
```

```
#import "AuthViewController.h"

@interface AuthViewController ()<UIAlertViewDelegate>

@end

@implementation AuthViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view.

    [self showAlertWithTitle:@"Privacy permissions"];
}

- (void)showAlertWithTitle:(NSString *)title
{
    if ([title length] > 0) {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:title
  message:nil
  delegate:self
  cancelButtonTitle:@"Cancel"
  otherButtonTitles:@"OK", nil];

        [self.window makeKeyWindow];
        [alert show];
    }
}

- (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex
{
    if (buttonIndex == 1) {
        if (self.completionHandler) {
            self.completionHandler();
            self.window.rootViewController = nil;
            self.window = nil;
        }
    }else {
        exit(0);
    }
}

@end
```

## 4. Manually initialize container Context.

If you have integrated the HTML5 container, offline package, and mini program components, you must manually initialize container Context in the `- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions` method. The code sample is as follows:

```
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    ...
    // Initialize container Context.
    [MPNebulaAdapterInterface setNBContextWhenEnablePrivacyAuth];

    ...
}
```

## Host the lifecycle of the app not by the framework

### 1. Support privacy pop-up windows.

In `MPaaSInterface` category, rewrite the `enableUserOverWriteAuthAlert` API method and return the corresponding privacy permission status.



```
< > Pods > Pods > mPaas > Frameworks > mPaas.framework > Headers > MPaaSInterface.h >
00
67 /**
68 *
69 *
70 *
71 */
72 - (BOOL)enableUserOverWriteAuthAlert;
73
```

#### Sample code

```
@implementation MPaaSInterface (mPaasDemo)

- (BOOL)enableUserOverWriteAuthAlert {
    // If the user has clicked "Agree" for privacy terms, "NO" is returned, which indicates that mPaas components can normally call relevant APIs.
    // Otherwise, "Yes" is returned, which indicates that mPaas components will hold the calls of relevant APIs.
    return ![NSUserDefaults standardUserDefaults] boolForKey:@"xx_pr"];
}

@end
```

### 2. Prevent the early reporting of log tracking.

If you have accessed tracking-related components, you must call the `MPAnalysisHelper` `holdUploadLogUntilAgreed` method in the startup process to prevent the early reporting of log tracking.

#### Note

You can determine whether tracking-related components have been accessed by checking whether `APRemoteLogging.framework` exists.

**Sample code** (we recommend that you call it at the earliest possible time)

```
8
9 #import "AppDelegate.h"
10 #import <MPMasAdapter/MPMasAdapter.h>
11
12 @interface AppDelegate ()
13
14 @end
15
16 @implementation AppDelegate
17
18
19 - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
20     // Override point for customization after application launch.
21     [MPAnalysisHelper holdUploadLogUntilAgreed];
22
23     return YES;
24 }
25
26
27 - (void)applicationWillResignActive:(UIApplication *)application {
28     // Sent when the application is about to move from active to inactive state. This can occur for certain types of tempor
```

### 3. Manually initialize container Context.

If you have integrated the HTML5 container, offline package, and mini program components, you must manually initialize container Context in the `- (void)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions` method. The code sample is as follows:

```
- (void)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    ...
    // Initialize container Context.
    [MPNebulaAdapterInterface setNBCContextWhenEnablePrivacyAuth];
    ...
}
```

## 5.9. Upgrade guide of mPaaS 10.1.68

### mPaaS 10.1.68 release notes

1. Starting from the 10.1.68 baseline, UIWebView has been officially discarded, and only WKWebView is supported. For details, see [mPaaS is adapted to WKWebView](#). App Store will no longer accept new apps that use UIWebView from April 2020, and will no longer accept updates to apps that use UIWebView from December 2020. Upgrade to the 10.1.68 baseline as soon as possible to adapt to WKWebView.
2. Xcode 11 is supported to build static library packages and is fully compatible with Xcode 11 development.

### mPaaS 10.1.68 upgrade instructions

#### Use CocoaPods for upgrade

#### Prerequisites

The CocoaPods mPaaS plug-in has been installed.

- If you have not installed the CocoaPods mPaaS plug-in, execute the following script on the terminal to install the plug-in.

```
sh <(curl -s http://mpaas-ios.oss-cn-  
hangzhou.aliyuncs.com/cocoapods/installmPaaS CocoaPodsPlugin.sh)
```

- If you have installed the CocoaPods mPaaS plug-in, directly run the upgrade command `pod mpaas update --all` to upgrade the plug-in. For details about using the CocoaPods mPaaS plug-in, see [Use CocoaPods for access based on the native framework](#).

## Procedure

1. Change the SDK version to **10.1.68** in Podfile.



```
plugin "cocoapods-mPaaS"  
source "https://code.aliyun.com/mpaas-public/podsspecs.git"  
mPaaS_baseline '10.1.68' # This line is maintained by MPaaS plugin automatically. Please  
mPaaS_version_code 2 # This line is maintained by MPaaS plugin automatically. Please  
don't modify.
```

2. Run the command `pod mpaas update 10.1.68` to install the latest SDK of version **10.1.68** for the baseline.
3. Run the `pod install` or `pod update` command as needed to upgrade the SDK to version 10.1.68 in the project.

## Follow-up steps

If you encounter the following error when accessing CocoaPods:

```
Invalid `Podfile` file: [!] No mPaaS_Nebula : 10.1.68 found !!! Check name & version in  
Podfile.
```

Try this solution:

1. Run the command `gem list | grep 'mPaaS'` to view the CocoaPods plug-in version, as shown in the following figure.



```
[TT-MAC:MPH5Demo_pod 2] $ gem list | grep 'mPaaS'  
cocoapods-mPaaS (0.9.5)
```

2. If the CocoaPods plug-in version is earlier than 0.9.5, execute the following script to reinstall the plug-in.

```
sh <(curl -s http://mpaas-ios.oss-cn-  
hangzhou.aliyuncs.com/cocoapods/installmPaaS CocoaPodsPlugin.sh)
```

## Component usage and upgrade instructions

If your current baseline version is earlier than 10.1.60 and integrates the HTML5 container and mini program components, read the following instructions in detail.

- Read [HTML5 container upgrade guide for Version 10.1.60](#) for details about upgrading HTML5 containers and offline packages.
- Read [Upgrade guide for mini program Version 10.1.60](#) for details about upgrading applets.

## Component API changes

Starting from the 10.1.32 baseline, the mPaaS component adds an adaptation layer. If your baseline is not using adaptation-layer APIs, read [mPaaS 10.1.32 is adapted to iOS 13](#) first.

We recommend that you use the APIs of the adaptation layer after upgrading the SDK. For details, see the following upgrade instructions for different components:

- [Mobile Gateway Service](#)
- [Configure project](#)
- [HTML5 Offline Packages](#)
- [Mobile Sync Service](#)
- [Client diagnosis](#)
- [Publishing Management](#)

#### Note

We strongly recommend that you modify the code and use middle-layer (adapter) methods instead of directly using underlying methods, because certain underlying methods may be modified or discarded in future versions. You may need to take lots of time adapting them in future updates if you continue to use them.

## Handle custom libraries

The components of the 10.1.68 baseline incorporate customization requirements. However, if you included custom libraries in your dependencies and upgraded the SDK from an earlier version (such as 10.1.32) to version 10.1.68, you may need to customize the custom libraries again based on the new version for security reasons. To do this, search for the group number 41708565 with DingTalk to join DingTalk group to ask.

## The sharing component

Third-party SDKs in the sharing component of version 10.1.68 has been upgraded, including the WeChat, Weibo, and QQ connection SDKs. Since the sharing of WeChat and QQ added the Universal Link feature in the latest version, be sure to adapt to the new SDKs, including:

1. The application configuration information of the corresponding platform, which can be viewed in app management (under the third-party developer account), is updated. For the specific adaptation method, visit the reference link.
2. For WeChat sharing, the "universalLink" field must be added to the configuration information of the mPaaS sharing component. The value of this field is the actual Universal Link address.

# 5.10. mPaaS 10.1.60 baseline upgrade guide

## About mPaaS 10.1.60

1. The 10.1.60 baseline now supports WKWebView. For details, see [Version 10.1.60 is adapted to WKWebView](#). Since App Store will no longer accept new apps that use UIWebView from April 2020, and will no longer accept updates to apps that use UIWebView from December 2020. For details, see [Apple's official announcement](#). For this reason, developers need to replace UIWebView with WKWebView.
2. The 10.1.60 baseline has been adapted to iOS 13 and Xcode 11. For details, see [mPaaS 10.1.60 is adapted to iOS 13](#).

3. The 10.1.60 baseline adds the mini program component. The official version of the mini program has a complete set of APIs, with greatly improved stability and compatibility. For mini program upgrade, see [Mini program upgrade instructions](#). For details about the debugging, preview, and publishing functions added for the mini program IDE, see [Develop mini programs](#).
4. The 10.1.60 baseline dramatically optimizes HTML5 containers, provides a more simplified access process, and significantly improves its compatibility and stability. For the upgrade of HTML5 containers and offline packages, see the [HTML5 container upgrade guide](#).
5. The 10.1.60 baseline adds MCDP (Mobile Content Delivery Platform) assembly. MCDP provides the feature to personalize advertising in applications, supports personalized advertising to targeted groups, and helps APP operators reach users accurately and timely. Please read [Mobile Content Delivery](#) for more information.
6. The 10.1.60 baseline greatly improves the compatibility and stability of overall components and its functionality. For the release notes for this version, see [iOS SDK release notes](#).
7. The 10.1.60 baseline no longer supports iOS 8.

## mPaaS 10.1.60 upgrade instructions

### Use CocoaPods for upgrade

#### Prerequisites

The CocoaPods mPaaS plug-in has been installed.

- If you have not installed the CocoaPods mPaaS plug-in, execute the following script on the terminal to install the plug-in.

```
sh <(curl -s http://mpaas-ios.oss-cn-hangzhou.aliyuncs.com/cocoapods/installmPaaS CocoaPodsPlugin.sh)
```

- If you have installed the CocoaPods mPaaS plug-in, directly run the upgrade command `pod mpaas update --all` to upgrade the plug-in. For details about using the CocoaPods mPaaS plug-in, see [Use CocoaPods for access based on the native framework](#).

#### Procedure

1. In Podfile, change the SDK version to **10.1.60**.



```
1 # mPaaS Pods Begin
2 plugin "cocoapods-mPaaS"#, :only_frameworks => true
3 source "https://code.aliyun.com/mpaas-public/podspecs.git"
4 mPaaS_baseline '10.1.60' # 请将 x.x.x 替换成真实基线版本
5 mPaaS_version_code 22 # This line is maintained by MPaaS plug:
6 # mPaaS Pods End
7 # -----
8
```

2. Run the command `pod mpaas update 10.1.60` to install the latest SDK of version **10.1.60** for the baseline.
3. Run the `pod install` or `pod update` command as needed to upgrade the SDK to version 10.1.60 in the project.

#### Follow-up steps

If you encounter the following error when accessing CocoaPods:

```
Invalid `Podfile` file: [!] No mPaaS_Nebula : 10.1.60-beta found !!! Check name & version in Podfile.
```

Try this solution:

1. Run the command `gem list | grep 'mPaaS'` to view the CocoaPods plug-in version, as shown in the following figure.

```
[TT-MAC:MPH5Demo_pod 2 ~]$ gem list | grep 'mPaaS'
cocoapods-mPaaS (0.9.5)
```

2. If the CocoaPods plug-in version is earlier than 0.9.5, execute the following script to reinstall the plug-in.

```
sh <(curl -s http://mpaaS-ios.oss-cn-
hangzhou.aliyuncs.com/cocoapods/installmPaaS-CocoaPodsPlugin.sh)
```

## Component usage and upgrade instructions

In the 10.1.60 baseline, the accessibility and usability of HTML5 container and mini program have been greatly improved. If you have used these components, read the following instructions in detail.

- Read [HTML5 container upgrade guide for version 10.1.60](#) for details about upgrading HTML5 containers and offline packages.
- Read [Upgrade guide for mini program Version 10.1.60](#) for details about upgrading applets.

## Component API changes

Starting from the 10.1.32 baseline, the mPaaS component adds an adaptation layer. If your baseline is not using adaptation-layer APIs, read [mPaaS 10.1.32 is adapted to iOS 13](#) first.

We recommend that you use the APIs of the adaptation layer after upgrading the SDK. For details, see the following upgrade instructions for different components:

- [Mobile Gateway](#)
- [HTML5 Offline Packages](#)
- [Mobile Sync Service](#)
- [Client diagnosis](#)
- [Publishing Management](#)

### Note

- Pay special attention to the [directory](#) and [info.plist](#) configuration changes for mPaaS components in the project.
- We strongly recommend that you modify the code and use middle-layer (adapter) methods instead of directly using underlying methods, because certain underlying methods may be modified or discarded in future versions. You may need to take lots of time adapting them in future updates if you continue to use them.

## Change of the directory structure

Among the component category directories and files under the `MPaaS` directory of the project, only `APMobileFramework` and `mPaas` are kept after the upgrade. All the other directories, such as `MPHotpatchSDK` and `APRemoteLogging`, are automatically removed. If there are any custom files saved under these directories, you need to back up them in advance. For details of the directory structure, see [mPaaS directory structure](#).

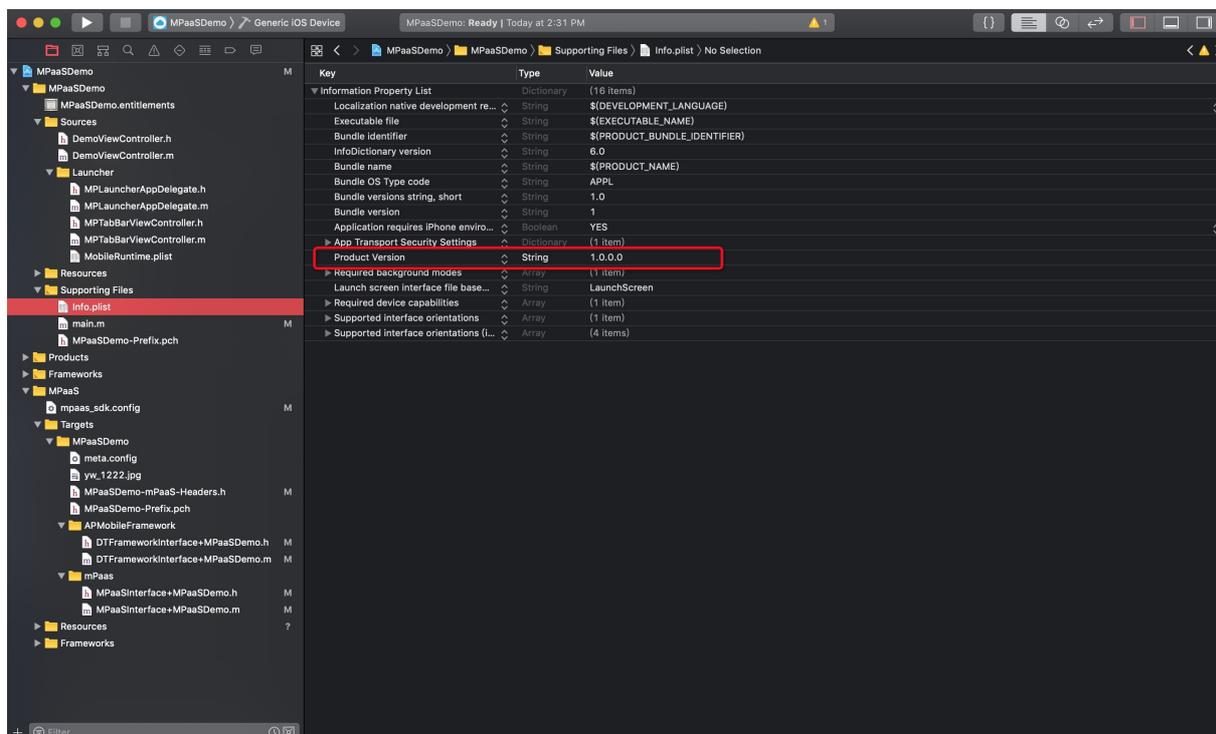
## Change of Info.plist

The following figure shows the related mPaaS fields inserted in the Info.plist file of the project before the upgrade.

| Key                                    | Type       | Value                         |
|----------------------------------------|------------|-------------------------------|
| Information Property List              | Dictionary | (19 items)                    |
| Localization native development re...  | String     | \$(DEVELOPMENT_LANGUAGE)      |
| 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 version                         | String     | 1                             |
| Application requires iPhone enviro...  | Boolean    | YES                           |
| App Transport Security Settings        | Dictionary | (1 item)                      |
| Required background modes              | Array      | (1 item)                      |
| Launch screen interface file base...   | String     | LaunchScreen                  |
| Required device capabilities           | Array      | (1 item)                      |
| Supported interface orientations       | Array      | (1 item)                      |
| Supported interface orientations (i... | Array      | (4 items)                     |
| Product Version                        | String     | 1.0.0.0                       |
| Product ID                             | String     | 570DA89281533_IOS-default     |
| mPaaS                                  | Dictionary | (4 items)                     |
| mPaaSInternal                          | Dictionary | (2 items)                     |

In 10.1.32 and later versions, only the `Product Version` field is required. After the baseline is upgraded, the plug-in automatically removes the `Product ID`, `mPaaS`, and `mPaaSInternal` fields. If the plug-in fails to remove these fields, you need to delete them manually. The following figure shows the fields after the upgrade.

**Note:** Do not delete the `Product Version` field when you delete the fields manually.



## Handle custom libraries

The components of the 10.1.60 baseline incorporate customization requirements. However, if you included custom libraries in your dependencies and upgraded the SDK from an earlier version (such as 10.1.32) to version 10.1.60, you may need to customize the custom libraries again based on the new version for security reasons. To do this, [submit a ticket](#) or contact mPaaS technical support personnel for consultation.

## 5.11. mPaaS 10.1.68 baseline adapt to Xcode 13

### Background

Since April.25th 2022, apps that are to be submitted to App Store must be built based on Xcode 13. For the new tool chain, the App needs to be adapted.

### Status quo

mPaaS has adapted and tested for Xcode 13 in 10.1.68.47 and above version baseline.

### Upgrade SDK/components

#### Use CocoaPods for upgrade

Follow the steps below to install the latest SDK version 10.1.68:

1. In Podfile, change the SDK version to **10.1.68**.
2. Run the command `pod mpaas update 10.1.68` . If it prompts an error, you need to update the plug-in first through the `pod mpaas update -all` command, and then execute command `pod mpaas update 10.1.68` again.
3. Run the command `pod install` .

### API changes

No API changes are required for this adaptation.

### Handle custom baselines

If you are using a custom baseline, you may need to customize the baseline again based on the new version. To do this, please search for group number 41708565 with DingTalk to join DingTalk group to contact mPaaS technical support staff for consultation.

### List of libraries adapted to iOS 15 updates

- The map component upgrades the default Amap to version 7.1.14.
- Share component.
- Some internal dependencies.

We recommend that you perform full regression testing in iOS 15 after you have adapted mPaaS 10.1.68 to iOS 15.

### Verification scope

Since the upgrade of Apple's toolchain is a black-box operation, it often brings stability and other issues. After the app is adapted to Xcode 13, it is recommended to conduct a comprehensive regression test

## 5.12. mPaaS 10.1.68 baseline adapt to iOS 15

This topic describes how to adapt mPaaS with the baseline 10.1.68 to iOS 15.

### Background

Apple has officially released iOS 15 in September, 2021. The apps must be adapted to new system features and APIs. Currently, mPaaS has adapted and tested for iOS 15 in the 10.1.68.38 and later baselines.

## Status quo

As the basic library, mPaaS has adapted and tested for iOS 15 built using the IPA package generated by Xcode 12. If you plan to launch your apps in Apple App Store, **you must use Xcode 12 for packaging**. The tool chain for Xcode 13 is being improved. After the tool chain is improved, mPaaS will release a version adapted to iOS 15 built under Xcode 13.

## Upgrade the SDK or components

### Upgrade based on CocoaPods

Install the latest SDK of version **10.1.68** by performing the following steps:

1. Check that the mPaaS version is **10.1.68** in Podfile.
2. Run the `pod mpaas update 10.1.68` command. If an error is returned, run the `pod mpaas update --all` command to update the plug-in and then rerun the command.
3. Run the `pod install` command.

## API changes

No API changes are required for this adaptation.

## Handle custom baselines

If you are using a custom baseline, you may need to customize the baseline again based on the new version. To do this, search for group number 41708565 with DingTalk to join DingTalk group to contact mPaaS technical support staff for consultation.

## List of libraries adapted to iOS 15 updates

- Mini program
- HTML5 container
- Some internally dependent components

We recommend that you perform full regression testing in iOS 15 after you have adapted mPaaS 10.1.68 to iOS 15.

# 5.13. mPaaS 10.1.68 baseline adapt to iOS 14

## Background

Apple officially released iOS 14 on September 17, 2020. App must be adapted to new system features and APIs. Currently, mPaaS with the baseline 10.1.68.17 or later has been adapted and tested for iOS 14.

Important: As the basic library, the mPaaS version 10.1.68.27 and later have been adapted to iOS 14 built under Xcode 12.

## Upgrade the SDK or components

### Upgrade based on Extension plug-in

Use the mPaaS Xcode Extension plug-in to upgrade SDK/components. You can choose the following two methods :

- Update product set
- Upgrade Baseline

Choose the upgrade method according to your own situation :

- If you have already used the Extension plug-in to manage component dependencies, however, the current baseline version is lower than 10.1.68. You can use the **upgrade baseline function** to upgrade to version 10.1.68.

 **Note**

You can view the currently used baseline version in the **upgrade baseline** of the plug-in.

- If you have already used plug-in to manage component dependencies, and the baseline version is 10.1.68, then you can use **Update product set** feature to upgrade modules you are using.
- If you have not used plug-in to manage component dependencies, you can upgrade by following steps:
  - i. [Install mPaaS Xcode Extension](#).
  - ii.

## Upgrade based on CocoaPods

Install the latest SDK of version **10.1.68** by completing the following steps:

1. Check that the mPaaS version is **10.1.68** in Podfile.
2. Run the command `pod mpaas update 10.1.68` .

If an error is returned, run the command `pod mpaas update --all` to update the plug-in and then rerun the previous command.

3. Run the `pod install` command.

## Handle custom baselines

If you are using a custom baseline, you may need to customize the baseline again based on the new version. To do this, [submit a ticket](#) or contact mPaaS technical support personnel for consultation.

# 5.14. mPaaS 10.1.60 baseline adapt to iOS 13

## Background

iOS 13 was officially released on September 20, 2019. During the testing of the iOS 13 beta and official versions, we found that some behaviors of the system changed. Therefore, you must perform app adaptation before using it, otherwise problems such as functional exceptions and crashes may occur.

Before the mPaaS adaptation, the major impact on the mPaaS SDK built by Xcode 10 on iOS 13 devices is as follows. **Since iOS 13 optimizes app startup and modifies the mirror loading mechanism, the system category may overwrite the category methods defined in the SDK. As a result, custom methods cannot return expected results.**

**Important:** As the basic library, the mPaaS version [10.1.60.26](#) and later have been adapted to iOS 13 built under Xcode 11.

## Upgrade the SDK or components

### Upgrade based on CocoaPods

Install the latest SDK of version **10.1.60** by completing the following steps.

1. Check that the mPaaS version is **10.1.60** in Podfile.

2. Run the command `pod mpaas update 10.1.60` .

If an error is returned, run the command `pod mpaas update --all` to update the plug-in and then rerun the previous command.

3. Run the `pod install` command.

### API changes

The mPaaS component added an adaptation layer in version 10.1.32 and later. We recommend that you use the API of the adaptation layer after upgrading the SDK. For details, see the following upgrade instructions for different components:

- [Mobile Gateway Service](#)
- [Configure project](#)
- [HTML5 Offline Packages](#)
- [Mobile Sync Service](#)
- [Client diagnosis](#)
- [Publishing Management](#)

#### Notes:

- Pay special attention to the [directory](#) and [info.plist](#) configuration changes for mPaaS components in the project.
- We strongly recommend that you modify the code and use middle-layer (adapter) methods instead of directly using underlying methods, because certain underlying methods may be modified or discarded in future versions. You may need to take lots of time adapting them in future updates if you continue to use them.

### Change of the directory structure

Among the component category directories and files under the `MPaaS` directory of the project, only `APMobileFramework` and `mPaas` are kept after the upgrade. All the other directories, such as `APRemoteLogging` , is automatically removed. If there are any custom files saved under these directories, you need to back up them in advance. For details of the directory structure, see [mPaaS directory structure](#).

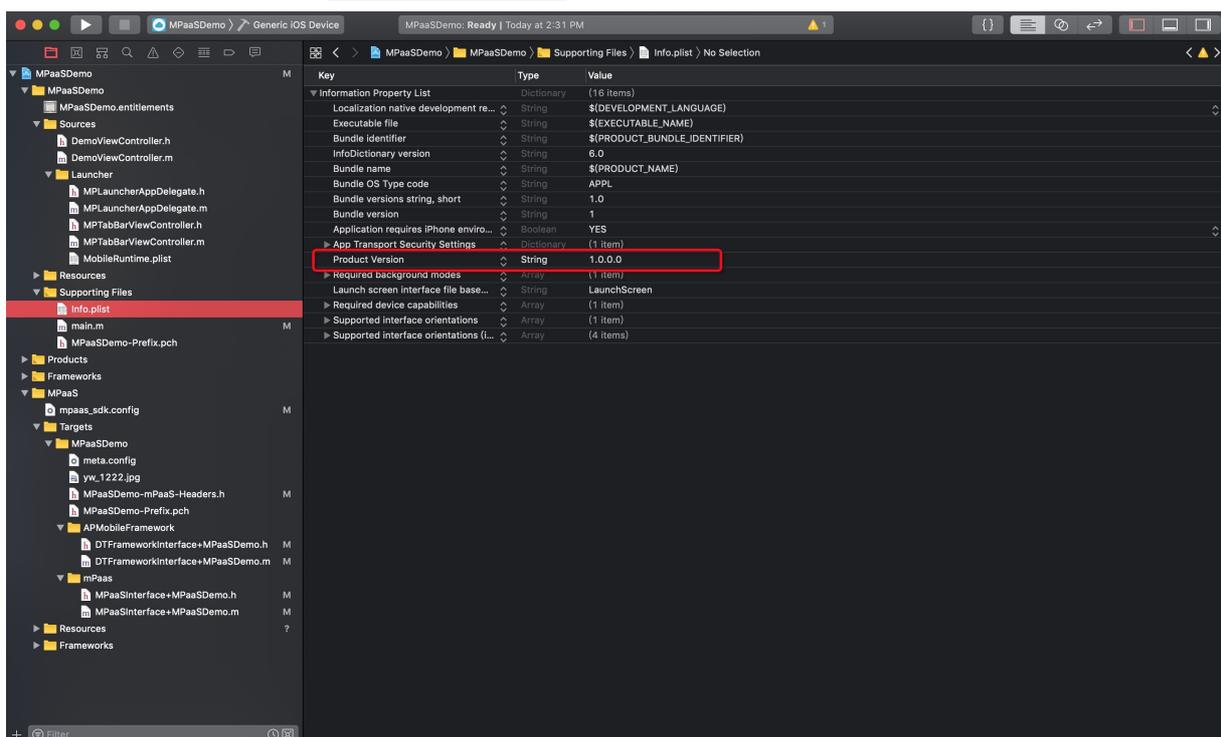
### Change of Info.plist

The following figure shows the related mPaaS fields inserted in the Info.plist file of the project before the upgrade.

| Key                                    | Type       | Value                         |
|----------------------------------------|------------|-------------------------------|
| Information Property List              | Dictionary | (19 items)                    |
| Localization native development re...  | String     | \$(DEVELOPMENT_LANGUAGE)      |
| 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 version                         | String     | 1                             |
| Application requires iPhone enviro...  | Boolean    | YES                           |
| App Transport Security Settings        | Dictionary | (1 item)                      |
| Required background modes              | Array      | (1 item)                      |
| Launch screen interface file base...   | String     | LaunchScreen                  |
| Required device capabilities           | Array      | (1 item)                      |
| Supported interface orientations       | Array      | (1 item)                      |
| Supported interface orientations (i... | Array      | (4 items)                     |
| Product Version                        | String     | 1.0.0.0                       |
| Product ID                             | String     | 570DA89281533_IOS-default     |
| mPaaS                                  | Dictionary | (4 items)                     |
| mPaaSInternal                          | Dictionary | (2 items)                     |

In 10.1.32 and later versions, only the `Product Version` field is required. After the baseline is upgraded, the plug-in automatically removes the `Product ID`, `mPaaS`, and `mPaaSInternal` fields. If the plug-in fails to remove these fields, you need to delete them manually. The following figure shows the fields after the upgrade.

**Note:** Do not delete the `Product Version` field when you delete the fields manually.



## Handle custom libraries

The components in version 10.1.60 incorporate customization requirements. However, if your dependencies include custom libraries, you must take the following actions to handle them accordingly for security reasons.

- If you upgraded the SDK from an earlier version (such as 10.1.20) to 10.1.60, you may need to customize custom libraries again based on the new version. To do this, [submit a ticket](#) or contact mPaaS technical support personnel.

- If the SDK version is 10.1.60, update certain components. See [List of libraries adapted to iOS 13 updates](#) to check whether your custom libraries are contained in the list.
  - If no, you can continue to use these custom libraries.
  - If yes, you may need to customize them again. To do this, [submit a ticket](#) or contact mPaaS technical support personnel.

## List of libraries adapted to iOS 13 updates

- mPaas
- MPDataCenter
- MPPushSDK
- APMultimedia
- BEEAudioUtil
- BeeCapture
- BeeCityPicker
- BeeMediaPlayer
- BeePhotoBrowser
- BeePhotoPicker
- NebulaAppBiz
- NebulaBiz
- NebulaSecurity
- NebulaKernel
- NebulaSDKPlugins
- NebulaSDK
- NebulaConfig
- NebulaTinyAppDebug
- NebulaNetwork
- TinyAppCommon
- APConfig
- AntUI
- MPPromotion
- BeeLocation
- MPMpaaSService
- TinyAppService
- AMap

## 5.15. mPaaS 10.1.32 baseline adapt to iOS 13

**Important:** Since June 28, 2020, mPaaS has discontinued the maintenance of the 10.1.32 baseline. We recommend that you use the [10.1.68](#) or [10.1.60](#) baseline.

## Background

iOS 13 was officially released on September 19, 2019. During the testing of iOS 13, we found that some behaviors of the system changed. Therefore, you must perform app adaptation before using it. Otherwise problems such as functional exceptions and crashes may occur.

Before the adaptation of mPaaS to iOS 13, the major impact on the mPaaS SDK built by Xcode 10 on iOS 13 devices is as follows: **Since iOS 13 optimizes app startup and modifies the mirror loading mechanism, the system category may overwrite the category methods defined in the SDK. As a result, custom methods cannot return expected results.**

## Status quo

As a basic library, mPaaS has been adapted to iOS 13 built under Xcode 10. Since mPaaS is currently adapted only for Xcode 10 packaging, **you must use Xcode 10 for packaging** and submit the package to App Store.

The tool chain for Xcode 11 is not yet complete. With the enhancement of the tool chain, mPaaS will release a version adapted to iOS 13 built under Xcode 11.

## Upgrade the SDK or components

### Upgrade based on CocoaPods

Install the latest SDK of version **10.1.32** by completing the following steps.

1. Check that the mPaaS version is **10.1.32** in Podfile.

2. Run the command `pod mpaas update 10.1.32`.

If an error is returned, run the command `pod mpaas update --all` to update the plug-in and then rerun the previous command.

3. Run the `pod install` command.

## API changes

The mPaaS component added an adaptation layer in version 10.1.32. We recommend that you use the API of the adaptation layer after upgrading the SDK. For details, see the following upgrade instructions for different components:

- [Mobile Gateway Service](#)
- [HTML5 Offline Packages](#)
- [Mobile Sync Service](#)
- [Client diagnosis](#)
- [Publishing Management](#)

### Notes:

- Pay special attention to the [directory](#) and [info.plist](#) configuration changes for mPaaS components in the project.
- We strongly recommend that you modify the code and use middle-layer (adapter) methods instead of directly using underlying methods, because certain underlying methods may be modified or discarded in future versions. You may need to take lots of time adapting them in future updates if you continue to use them.

## Change of the directory structure

Among the component category directories and files under the `MPaaS` directory of the project, only `APMobileFramework` and `mPaas` are kept after the upgrade. All the other directories, such as `APRemoteLogging`, is automatically removed. If there are any custom files saved under these directories, you need to back up them in advance. For details of the directory structure, see [mPaaS directory structure](#).

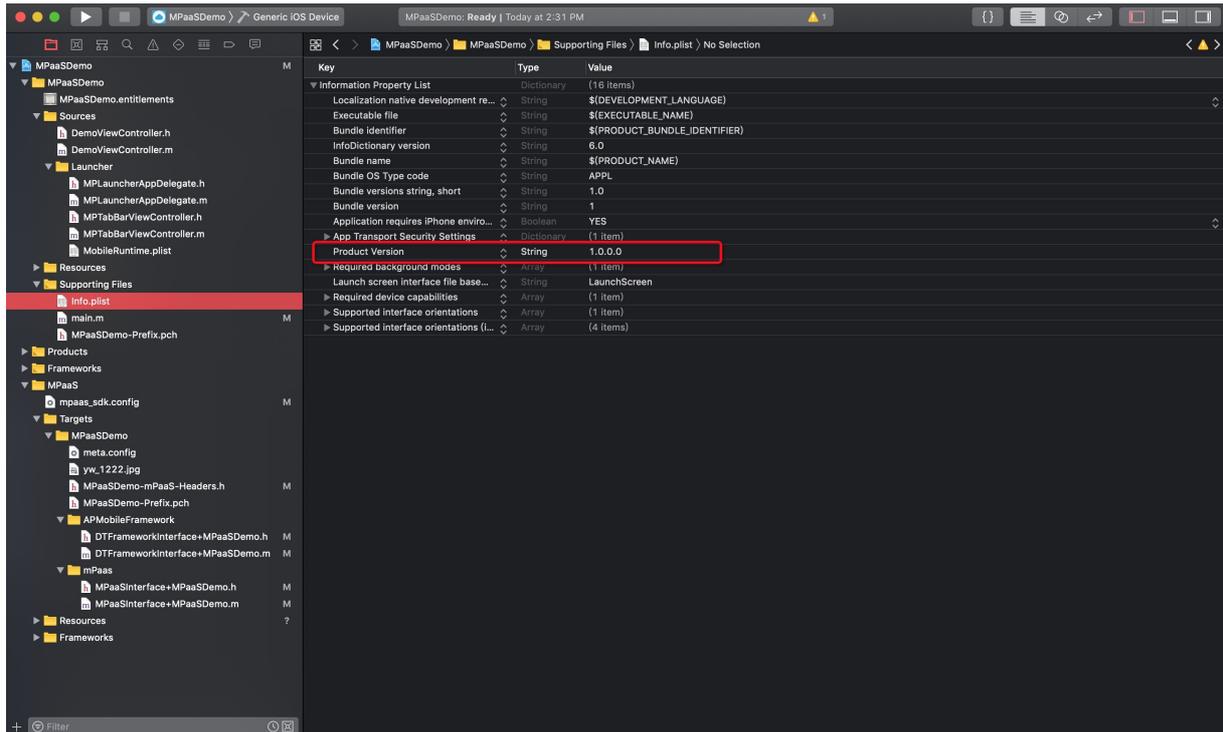
## Change of Info.plist

The following figure shows the related mPaaS fields inserted in the `Info.plist` file of the project before the upgrade.

| Key                                    | Type       | Value                         |
|----------------------------------------|------------|-------------------------------|
| Information Property List              | Dictionary | (19 items)                    |
| Localization native development re...  | String     | \$(DEVELOPMENT_LANGUAGE)      |
| 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 version                         | String     | 1                             |
| Application requires iPhone enviro...  | Boolean    | YES                           |
| App Transport Security Settings        | Dictionary | (1 item)                      |
| Required background modes              | Array      | (1 item)                      |
| Launch screen interface file base...   | String     | LaunchScreen                  |
| Required device capabilities           | Array      | (1 item)                      |
| Supported interface orientations       | Array      | (1 item)                      |
| Supported interface orientations (i... | Array      | (4 items)                     |
| Product Version                        | String     | 1.0.0.0                       |
| Product ID                             | String     | 570DA89281533_IOS-default     |
| mPaaS                                  | Dictionary | (4 items)                     |
| mPaaSInternal                          | Dictionary | (2 items)                     |

In 10.1.32 and later versions, only the `Product Version` field is required. After the baseline is upgraded, the plug-in automatically removes the `Product ID`, `mPaaS`, and `mPaaSInternal` fields. If the plug-in fails to remove these fields, you need to delete them manually. The following figure shows the fields after the upgrade.

**Note:** Do not delete the `Product Version` field when you delete the fields manually.



## Handle custom libraries

The components in version 10.1.32 incorporate customization requirements. However, if your dependencies include custom libraries, you must take the following actions to handle them accordingly for security reasons.

- If you upgraded the SDK from an earlier version to 10.1.32, you may need to customize custom libraries again based on the new version. To do this, [submit a ticket](#) or contact mPaaS technical support personnel for confirmation.
- If the SDK version is 10.1.32, upgrade certain components. See [List of libraries adapted to iOS 13 updates](#) to check whether your custom libraries are contained in the list.
  - If no, you can continue to use these custom libraries.
  - If yes, you may need to customize them again. To do this, [submit a ticket](#) or contact mPaaS technical support personnel.

## List of libraries adapted to iOS 13 updates

- mPaaS
- MPDataCenter
- APMultimedia
- BEEAudioUtil
- BeeCapture
- BeeCityPicker
- BeeMediaPlayer
- BeePhotoBrowser
- BeePhotoPicker
- NebulaAppBiz
- NebulaBiz

- NebulaSDKPlugins
- APConfig
- AntUI
- NebulaSDK
- TinyAppCommon
- MPPromotion

# 6. Reference

## 6.1. Customize the iOS navigation bar

During app development, it is often required to customize the top navigation bar. This topic describes how to customize the navigation bar on a page created **based on the MPaaS framework** and including [customizing the theme of the app](#) and [customizing the navigation bar style for a specific page](#).

### Basic concepts

#### Distribution of navigation bar elements

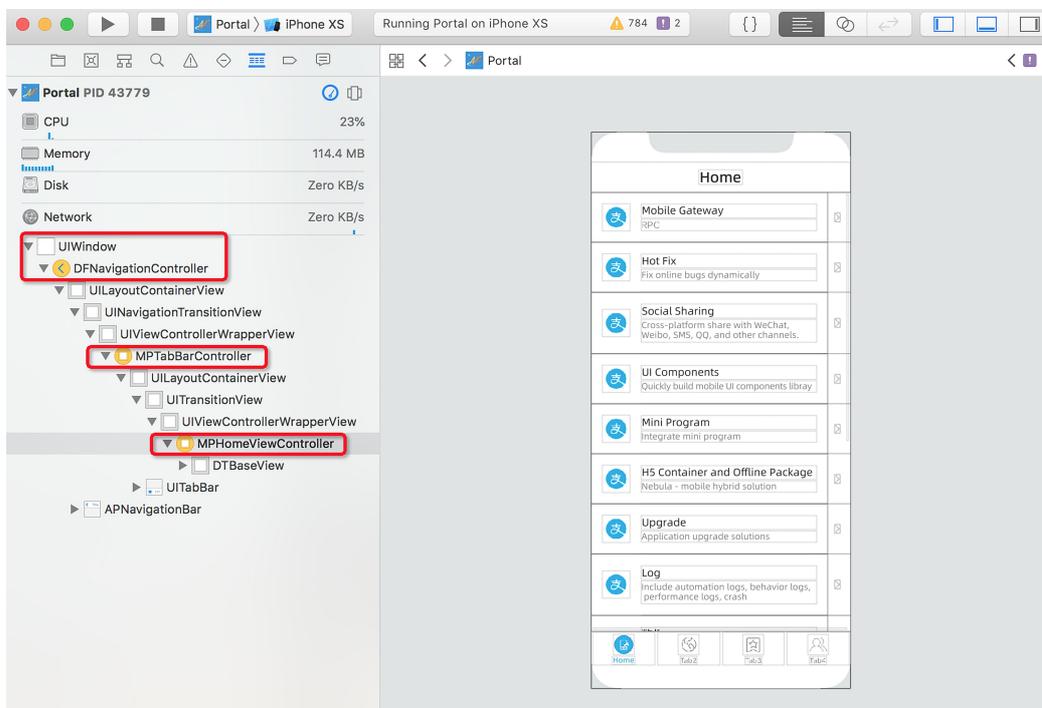
Navigation bar elements are mainly distributed in three areas. Generally, navigation bar customization involves modifying these areas.



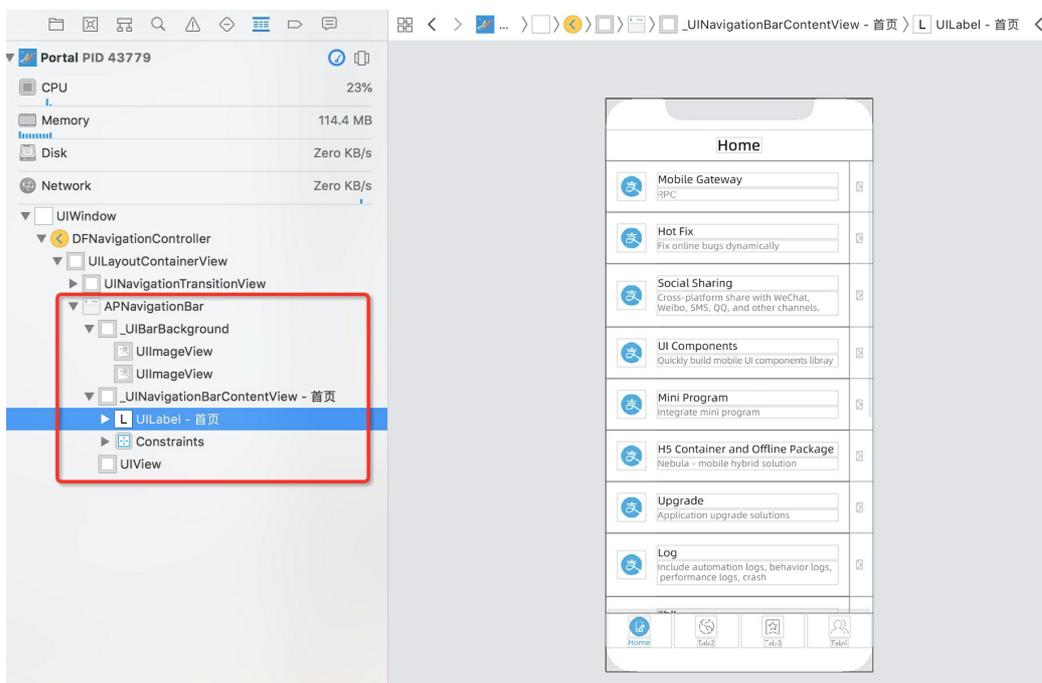
1. Back: The Back button control area that is created by the base class of the mPaaS page. The default format is back arrow + "Back".
2. Title/Subtitle: The title bar control area, which is not displayed by default. To display this area, call the system method to set the title of the current page.
3. Option menu: The page menu option area, which is not displayed by default. To display this area, call the system method to set rightNavigationItem for the current page.

#### Navigation bar structure

- As shown in the following figure, the default UI structure for apps created based on the mPaaS framework is as follows: `window/navigationController > tableViewController > viewController` embedded for each tab. In other words, the root of the main window of the app is a `UINavigationController` object, and the root of `UINavigationController` is a `UITabBarController`.



- The preceding UI structure shows that the entire app has only one `navigationController` globally. Therefore, all pages share the same navigation bar (created by using `APNavigationBar` by default).



- For unifying the navigation bar styles of all pages, it is required that in the mPaaS app, all VCs where a page resides must inherit `DTViewController`, including the native and HTML5 pages.

- The default theme of the app created based on the mPaaS framework is characterized by a white background, black text, and blue button.



## Customize the theme of the app

Each app has its own theme. Modify the default theme of the mPaaS app as follows:

- To modify the background color of the navigation bar, back control area, or title control area, rewrite the `au_defaultTheme_extraInfo` method of the `AUThemeManager` class and modify the return values of the following keys:
  - API method

```
@interface AUThemeManager(AUExtendInfo)
/* Default theme is set for Alipay client, and is modifiable for independent apps.
 * n this method only need to return a key-value pair different from the default the
 * me. Please use the key defined in AUTheme.h.
 */
+ (NSDictionary *)au_defaultTheme_extraInfo;

@end
/*
 * Example
 * +(NSDictionary*)au_defaultTheme_add_Info
 * {
 *     NSMutableDictionary *dict = [NSMutableDictionary alloc] init];
 *     dict[TITLEBAR_BACKGROUND_COLOR] = AU_COLOR_APP_GREEN; // AUTitleBar background
 *     dit[TITLEBAR_TITLE_TEXTCOLOR1] = [UIColor redColor]; // AUTitleBar title color
 *     ...
 *     return dict;
 * }
 */
```

- Sample code

```
@implementation AUThemeManager (Portal)

+ (NSDictionary *)au_defaultTheme_extraInfo
{
    NSMutableDictionary *dict = [[NSMutableDictionary alloc] init];
    dict[TITLEBAR_BACKGROUND_COLOR] = @"COLOR(#108EE9,1)"; // Set the background color of the navigation bar
    dict[TITLEBAR_LINE_COLOR] = @"COLOR(#108EE9,1)"; // Set the color of the separation line or sideline at the bottom of the navigation bar
    dict[TITLEBAR_TITLE_TEXTCOLOR] = @"COLOR(#ffffff,1)"; // Set the title text color of the navigation bar
    dict[TITLEBAR_TITLE_TEXTSIZE_BOLD] = @"FONT(18)"; // Set the title text font size of the navigation bar
    dict[TITLEBAR_TEXTCOLOR] = @"COLOR(#ffffff,1)"; // Set the Back button color of the navigation bar

    return dict;
}

@end
```

 **Note**

Note: You must set the color value in the COLOR(#108EE9,1) format, otherwise an error will be returned.

- To modify the Back button icon in theme configuration, rewrite the `au_default_backButtonImg` method in the `AUBarButtonItem` class.

- API method

```
#import "AUUILoadDefine.h"//The program automatically generates.
#ifdef ANTUI_UI_TitleBar_AUBarButtonItem//The program automatically generates.
//
// AUBarButtonItem+AUExtendInfo.h
// AntUI
//
// Copyright © 2017 Alipay. All rights reserved.
//
#import "AUBarButtonItem.h"

@interface AUBarButtonItem(AUExtendInfo)

//Default return button is a blue icon for Alipay, and is modifiable for independent apps.
+ (UIImage *)au_default_backButtonImg;

@end
```

- Sample code

```
@implementation AUBarButtonItem (CGBBarButtonItem)

+ (UIImage *)au_default_backButtonImg
{
    // Customize the Back icon
    return ACommonUILoadImage(@"back_button_normal_white");
}

@end
```

- Modify the Back button style and text for all pages.

## Customize the navigation bar style for a specific page

In addition to themes, sometimes you need to customize the navigation bar style for the current page, for example, modify the background color or Back button style. mPaaS provides different methods for modification at different times.

- **Before loading a page**, to modify the navigation bar color on the basis of the default navigation bar style, implement the method defined in `DTNavigationBarAppearanceProtocol` and modify the color of the corresponding area.
  - API method

```
@protocol DTNavigationBarAppearanceProtocol<NSObject>

@optional

/** Whether this DTViewController needs to automatically hide navigationBar. The default value is NO, If a ViewController in business needs to hide navigationBar, reload this method and return YES.
**/
-(BOOL)autohideNavigationBar;

/** If the current VC needs to set a hidden navigation bar as fully transparent, and set the current page with the return copy matching the framework logic., reload this method and return an instance of APCustomerNavigationView.
-(UIView *)customNavigationBar;

/** If a viewcontroller needs to set its titlebar as opaque and assign a color to the titlebar, rewrite this method and return the expected color.
* Only for the pushed VC. VC in tabbar can not modify the translucent property of navigationBar.
*/
-(UIColor *)opaqueNavigationBarColor;

/**
* If a viewcontroller needs to modify the style of status bar, rewrite this method and return the expected style.
*/
- (UIStatusBarStyle)customStatusBarStyle;

/**
* If a viewcontroller wants to modify the color of the navigation bar title, please override this method and return the desired color.
*/
- (UIColor *)customNavigationBarTitleColor;
```

- Sample code

```
#pragma mark DTNavigationBarAppearanceProtocol: Modify the navigation bar style when entering the page.
- (UIColor *)opaqueNavigationBarColor
{
    // Set the background color of the navigation bar to red for the current page .
    return [UIColor redColor];

    // // Set the navigation bar to transparent for the current page
    // return [UIColor colorWithRGB:0xff0000 alpha:0];
}

- (BOOL)autohideNavigationBar
{
    // Set whether to hide the navigation bar for the current page
    return NO;
}

- (UIStatusBarStyle)customStatusBarStyle
{
    // Set the status bar style for the current page
    return UIStatusBarStyleDefault;
}

- (UIColor *)customNavigationBarBackButtonTitleColor
{
    // Set the text color of the Back button for the current page
    return [UIColor greenColor];
}

- (UIImage *)customNavigationBarBackButtonImage
{
    // Set the Back icon for the current page.
    return [UIImage imageNamed:@"back_button_normal_white"];
}

- (UIColor *)customNavigationBarTitleColor
{
    // Set the title color for the current page.
    return [UIColor greenColor];
}
```

- **After a page is opened**, you can modify the navigation bar style and the menu button on the right side of the page during your operations. For example, you can make the background color to change gradually when you slide the slider. Modification of the following areas is supported:
  - Background area: Hide/Show the navigation bar, set the navigation bar to transparent, modify the background color of the navigation bar, and modify the color of the status bar.

```
- (void)gotoHideNavigator
{
    // Hide the navigation bar
    [self.navigationController.navigationBar setHidden:YES];
}

- (void)gotoShowNavigator
{
    // Show the navigation bar
    [self.navigationController.navigationBar setHidden:NO];
}

- (void)gotoTransparency
{
    // Set the navigation bar to transparent
    [self.navigationController.navigationBar setNavigationBarTranslucentStyle];
}

- (void)gotoUpdateBackgroundColor
{
    // Modify the background color of the navigation bar
    [self.navigationController.navigationBar setNavigationBarStyleWithColor:[UIColor whiteColor] translucent:NO];
    [self.navigationController.navigationBar setNavigationBarBottomLineColor:[UIColor whiteColor]];
}

- (void)gotoUpdateStatusBarStyle
{
    // Modify the status bar color
    [[UIApplication sharedApplication]
    setStatusBarStyle:UIStatusBarStyleLightContent];
}
```

- **Back control area:** Modify the default text and color of the Back button, modify the default arrow style of the Back button, and reset the style of the Back button.

```
- (void)gotoUpdateBackTitleColor
{
    // Modify the default text color of the Back button
    NSArray *leftBarButtonItems = self.navigationController.leftBarButtonItems;
    if ([leftBarButtonItems count] == 1) {
        if (leftBarButtonItems[0] && [leftBarButtonItems[0] isKindOfClass:
[AUBarButtonItem class]]) {
            AUBarButtonItem *backItem = leftBarButtonItems[0];
            backItem.titleColor = [UIColor blackColor];
        }
    }
}

- (void)gotoUpdateBackImage
{
    // Modify the default arrow style of the Back button
    NSArray *leftBarButtonItems = self.navigationController.leftBarButtonItems;
    if ([leftBarButtonItems count] == 1) {
        if (leftBarButtonItems[0] && [leftBarButtonItems[0] isKindOfClass:
[AUBarButtonItem class]]) {
            AUBarButtonItem *backItem = leftBarButtonItems[0];
            backItem.backButtonItemImage = APCommonUILoadImage(@"back_button_normal");
        }
    }
}

- (void)gotoUpdateBackItem
{
    // Reset the style of the Back button
    self.navigationController.leftBarButtonItem = [AUBarButtonItem
BarButtonItemWithImageType:AUBarButtonItemImageTypeDelete target:self
action:@selector(onClickBack)];
}

- (void)onClickBack
{
    [self.navigationController popViewControllerAnimated:YES];
}
```

- Title control area: Modify the default title color, set the title and subtitle, and enable the title to be displayed as a picture. ````
- (void)gotoUpdateTitleColor{ // Modify the title color  
[self.navigationController.navigationBar  
setNavigationBarTitleTextAttributesWithTextColor:[UIColor blackColor];}
- (void)gotoTwoTitle{ // Modify the title style: title and subtitle  
self.navigationItem.titleView = [[AUDoubleTitleView alloc] initWithTitle:@"Title"  
detailTitle:@"Subtitle"];}
- (void)gotoTitleImage{ // Modify the title style to picture UIImageView \*imageView =  
[[UIImageView alloc]  
initWithImage:APCommonUILoadImage(@"illustration\_ap\_expectation\_alert");  
imageView.frame = CGRectMake(0, 0, self.self.view.width-100, 64);  
self.navigationItem.titleView = imageView;}```

- Menu control area: Set a single or multiple menu buttons on the right.

```
- (void)gotoSetOptionsMenu
{
    // Set a single button on the right side
    self.navigationItem.rightBarButtonItem = [AUIBarButtonItem
barButtonItemWithImageType:AUIBarButtonItemImageTypeGroupChat target:self
action:@selector(onClickRightItem)];
}

- (void)gotoSetTwoOptionsMenu
{
    // Set two buttons on the right side
    UIBarButtonItem *item1 = [AUIBarButtonItem
barButtonItemWithImageType:AUIBarButtonItemImageTypeGroupChat target:self
action:@selector(onClickRightItem)];
    UIBarButtonItem *item2 = [AUIBarButtonItem
barButtonItemWithImageType:AUIBarButtonItemImageTypeHelp target:self
action:@selector(onClickRightItem)];
    self.navigationItem.rightBarButtonItemItems = @[item1, item2];
}
```

- **Immersive navigation bar:** In immersive mode, the navigation bar is transparent when you access the page, and opaque after you slide to a specified position. The details are as follows:
  - Set the navigation bar to be transparent when you access the page. Rewrite the following API in the VC where the current page resides.

```
- (UIColor *)opaqueNavigationBarColor
{
    // Set the navigation bar to transparent for the current page
    return [UIColor colorWithRGB:0xff0000 alpha:0];
}
```

- After sliding to a specified position, modify the styles of the background area, back area, title area, and menu area of the navigation bar.

```
- (void)gotoUpdateBackgroundColor
{
    // Modify the background color of the navigation bar
    [self.navigationController.navigationBar setNavigationBarStyleWithColor:[UIColor whiteColor] translucent:NO];
    [self.navigationController.navigationBar setNavigationBarBottomLineColor:[UIColor whiteColor]];
}

- (void)gotoUpdateBackTitleColor
{
    // Modify the default text color of the Back button
    NSArray *leftBarButtonItems = self.navigationItem.leftBarButtonItems;
    if ([leftBarButtonItems count] == 1) {
        if (leftBarButtonItems[0] && [leftBarButtonItems[0] isKindOfClass:[AUBarButtonItem class]]) {
            AUBarButtonItem *backItem = leftBarButtonItems[0];
            backItem.titleColor = [UIColor blackColor];
        }
    }
}

- (void)gotoUpdateTitleColor
{
    // Modify the title color
    [self.navigationController.navigationBar setNavigationBarTitleTextAttributesWithTextColor:[UIColor blackColor]];
}
```

## 6.2. Handle iOS conflict

When accessing mPaaS, the mPaaS SDK may conflict with other open-source libraries or third-party libraries introduced into the project, leading to project compilation failure. This topic introduces the solutions to two common types of conflicts.

Based on the types of libraries that cause the conflict, two categories of solutions are available:

- mPaaS custom libraries: If custom libraries of the mPaaS SDK conflict with other libraries in the project, you must use these custom mPaaS libraries.
- Non-mPaaS custom libraries: If conflicting mPaaS SDK libraries are not mPaaS custom libraries, you can delete the libraries introduced by mPaaS.

### Solutions to conflicting mPaaS custom libraries

If conflicting mPaaS SDK libraries are custom libraries, you must use these custom mPaaS libraries.

| Open-source library name | mPaaS library name | Conflict solution |
|--------------------------|--------------------|-------------------|
|--------------------------|--------------------|-------------------|

|                 |                   |                                                                                                                                                                                                |
|-----------------|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AlipaySDK       | AlipaySDK         | The mPaaS version, which solves conflicts with modules such as mPaaS RPC and UTDID, must be used. At the same time, the mPaaS_RPC component needs to be integrated.                            |
| OpenSSL         | APOpenSSL         | The mPaaS version, which optimizes the original national secret algorithm, must be used. For more details, please refer to <a href="#">How to solve the OpenSSL conflict in iOS projects</a> . |
| protocolBuffers | APProtocolBuffers | The mPaaS version must be used.                                                                                                                                                                |

## Solutions to non-mPaaS custom libraries

If conflicting mPaaS SDK libraries are not mPaaS custom libraries, you can delete the libraries introduced by mPaaS. The deletable libraries are shown in the following table. For details, see [Remove conflicting third-party libraries](#) to remove the conflicting libraries.

| Components supported by remove_pod | Open-source libraries contained |
|------------------------------------|---------------------------------|
| mPaaS_SDWebImage                   | SDWebImage                      |
| mPaaS_Masonry                      | Masonry                         |
| mPaaS_MBProgressHUD                | MBProgressHUD                   |
| mPaaS_TTTAttributedLabel           | TTTAttributedLabel              |
| mPaaS_Lottie                       | Lottie                          |
| mPaaS_AMap                         | AMapSearchKit                   |
|                                    | AMapFoundationKit               |
|                                    | MAMapKit                        |
| mPaaS_Security                     | SecurityGuard<br>SGMain         |
| mPaaS_APWebP                       | WebP                            |

## Remove conflicting third-party libraries

If the conflicting mPaaS SDK library is not a mPaaS custom library, you can delete the library introduced by mPaaS according to the following procedures.

## Procedure

1. Install the beta version of the cocoapods-mPaaS plug-in.

### Note

The beta version of the cocoapods-mPaaS plug-in is supported only in the 10.2.3 baseline.

```
sh <(curl -s http://mpaas-ios-test.oss-cn-hangzhou.aliyuncs.com/cocoapods/installmPaaS CocoaPodsPlugin.sh)
```

After the installation is completed, run the command `pod mpaas version --plugin` to verify that the installed version is the beta version.

2. Run the `pod mpaas update 10.2.3` command again to update the local baseline.
3. Introduce `remove_pod "mPaaS_xxx"` to podfile and be sure to place `remove_pod` before a common `mPaaS_pod` command.

For example, to remove SDWebImage, run this command: `remove_pod "mPaaS_SDWebImage"`.

```
remove_pod "mPaaS_SDWebImage"  
mPaaS_pod "mPaaS_CommonUI"  
pod 'xxx' # The corresponding third-party native library
```

4. After you remove the mPaaS component library, you can run the `pod install` command to introduce the native version.

## 6.3. Switch iOS environment

During app development, the app environment (namely, workspace) may occasionally change, and the app may be developed in multiple workspaces in parallel. mPaaS provides a tool for you to conveniently switch among environments (workspaces) during development. There are two types of workspace switching modes:

- Static workspace switching
- Dynamic workspace switching

### Static workspace switching

In static workspace switching, the default `meta.config` configuration file in the project is manually replaced on the client and then repackaged for access in a new workspace.

### Note

This method is only applicable to scenarios where only the configuration information of the current application environment is updated.

1. Replace the `meta.config` file of the current project with the mPaaS plug-in.
2. Delete and then reinstall the app and the new workspace configuration information takes effect immediately.

### Dynamic workspace switching

In dynamic workspace switching, workspace options in mobile phone settings are modified to dynamically modify the app workspace information without repackaging on the client.

#### Note

- This mode applies to a scenario where an app is developed in multiple workspaces in parallel and is frequently switched among them in the development phase.
- The dynamic workspace switching feature is only supported on Apsara Stack.

Restricted by the mPaaS security signature verification mechanism, updating workspace configuration information modifies the WSG signature verification image `yw_1222.jpg`. Therefore, dynamic workspace switching has two restrictions.

- The dynamic workspace switching mode applies only to the development phase. Therefore, be sure to delete the corresponding configuration before the app is released.
- Signature verification for network requests must be disabled in the mPaaS console. Otherwise, requests will fail due to incorrect signature verification image information.

```
![ ddd](http://docs-aliyun.cn-hangzhou.oss.aliyun-inc.com/assets/pic/111262/AntCloud_zh/1552905359956/%E5%9B%BE%E7%89%874.png)
```

## Workspace information configuration

1. Enable dynamic workspace switching. Specifically, rewrite the `enableSettingService` method in `category` of `MPaaSInterface` and return YES.

```
@implementation MPaaSInterface (Portal)

- (BOOL)enableSettingService
{
    return YES;
}

@end
```

2. Download the workspace configuration file [Settings.bundle.zip](#), and then add it to the project.
3. Understand the workspace configuration information in `Setting.bundle`.
  - `Setting.bundle` provides four workspaces by default, corresponding to the 4 configuration files: Debug, Sit, Release, and Custom respectively.
  - Among them, Debug, Sit and Release are preset workspaces. You need to download the configuration files of the target workspaces from the mPaaS console. Rename them to Debug, Sit, and Release, and use them to replace the example files in the original `Setting.bundle`.

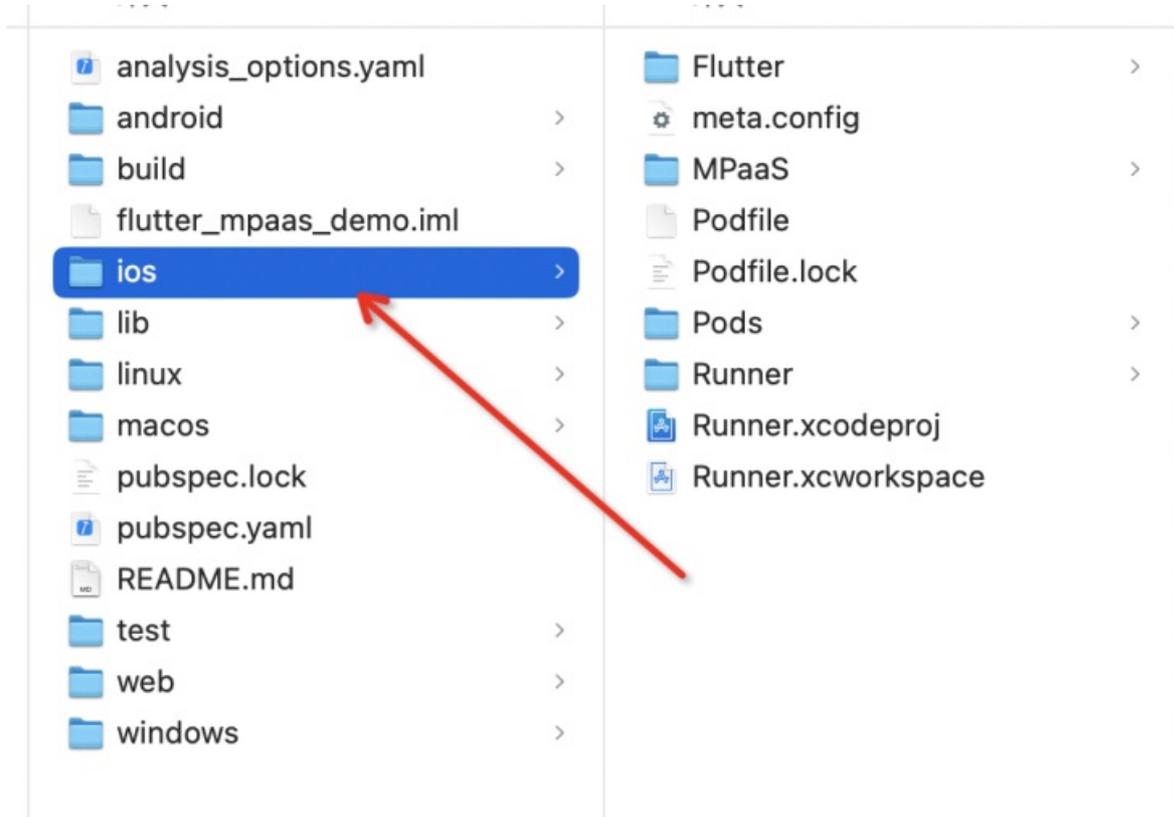


- To switch to another workspace, go to **Mobile phone Settings > mPaaS app > Settings > Category** and select the workspace. End the process and start it again for the new workspace to take effect.

# 7. Flutter Project Access Operation Guide

## Prerequisites

Xcode has been installed and a Flutter project has been created.



## Access procedure

For more information about how to access a Flutter project, see [Access a Flutter project by using CocoaPods](#).

## FAQ

When a Flutter project uses a CocoaPods-mPaaS plug-in to access the mPaaS framework, the following error may occur:

```
### Error
...
Errno::ENOENT - No such file or directory @ rb_check_realpath_internal - (eval)
(eval):41:in `realpath'
(eval):41:in `block in run'
/Users/vinsent/.rvm/rubies/ruby-2.7.2/lib/ruby/gems/2.7.0/gems/cocoapods-mPaaS-0.9.5/lib/cocoapods-mPaaS/mPaaSPodfile.rb:8:in `target'
(eval):40:in `run'
/Users/vinsent/.rvm/rubies/ruby-2.7.2/lib/ruby/gems/2.7.0/gems/cocoapods-mPaaS-0.9.5/lib/cocoapods-mPaaS/mPaaSPodfile.rb:39:in `eval'
/Users/vinsent/.rvm/rubies/ruby-2.7.2/lib/ruby/gems/2.7.0/gems/cocoapods-mPaaS-0.9.5/lib/cocoapods-mPaaS/mPaaSPodfile.rb:39:in `run'
/Users/vinsent/.rvm/rubies/ruby-2.7.2/lib/ruby/gems/2.7.0/gems/cocoapods-mPaaS-0.9.5/lib/cocoapods-mPaaS/baselineTools.rb:149:in `getBaselineMatched'
/Users/vinsent/.rvm/rubies/ruby-2.7.2/lib/ruby/gems/2.7.0/gems/cocoapods-mPaaS-0.9.5/lib/cocoapods-mPaaS/cocoapods_hook.rb:747:in `block (2 levels) in <module:CocoapodsMpaasHooks>'
/Users/vinsent/.rvm/rubies/ruby-2.7.2/lib/ruby/gems/2.7.0/gems/cocoapods-mPaaS-0.9.5/lib/cocoapods-mPaaS/cocoapods_hook.rb:744:in `each'
/Users/vinsent/.rvm/rubies/ruby-2.7.2/lib/ruby/gems/2.7.0/gems/cocoapods-mPaaS-0.9.5/lib/cocoapods-mPaaS/cocoapods_hook.rb:744:in `block in <module:CocoapodsMpaasHooks>'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/hooks_manager.rb:122:in `block (3 levels) in run'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/user_interface.rb:145:in `message'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/hooks_manager.rb:116:in `block (2 levels) in run'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/hooks_manager.rb:115:in `each'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/hooks_manager.rb:115:in `block in run'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/user_interface.rb:145:in `message'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/hooks_manager.rb:114:in `run'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/installer.rb:638:in `run_plugins_post_install_hooks'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/installer.rb:612:in `perform_post_install_actions'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/installer.rb:169:in `install!'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/command/install.rb:52:in `run'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/claide-1.0.3/lib/claide/command.rb:334:in `run'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/lib/cocoapods/command.rb:52:in `run'
/Users/vinsent/.rvm/gems/ruby-2.7.2/gems/cocoapods-1.10.1/bin/pod:55:in `<top (required)>'
/Users/vinsent/.rvm/gems/ruby-2.7.2/bin/pod:23:in `load'
/Users/vinsent/.rvm/gems/ruby-2.7.2/bin/pod:23:in `<main>'
/Users/vinsent/.rvm/gems/ruby-2.7.2/bin/ruby_executable_hooks:22:in `eval'
/Users/vinsent/.rvm/gems/ruby-2.7.2/bin/ruby_executable_hooks:22:in `<main>'
...
TEMPLATE END
```

## Solution

1. Modify the podfile file and save it.

```
14 project 'Runner', {
15   'Debug' => :debug,
16   'Profile' => :release,
17   'Release' => :release,
18 }
19
20 def flutter_root
21   generated_xcode_build_settings_path = File.expand_path(File.join('..', 'Flutter', 'Generated.xcconfig'), __FILE__)
22   unless File.exist?(generated_xcode_build_settings_path)
23     raise "#{generated_xcode_build_settings_path} must exist. If you're running pod install manually, make sure flutter pub get is executed first"
24   end
25
26   File.foreach(generated_xcode_build_settings_path) do |line|
27     matches = line.match(/FLUTTER_ROOT=(.*)/)
28     return matches[1].strip if matches
29   end
30   raise "FLUTTER_ROOT not found in #{generated_xcode_build_settings_path}. Try deleting Generated.xcconfig, then run flutter pub get"
31 end
32
33 require File.expand_path(File.join('packages', 'flutter_tools', 'bin', 'podhelper'), flutter_root)
34
35 flutter_ios_podfile_setup
36
37 target 'Runner' do
38
39   flutter_install_all_ios_pods __dir__ #File.dirname(File.realpath(__FILE__))
40   remove_pod 'mPaaS_UIDID'
41   mPaaS_pod "mPaaS_Config"
42   # mPaaS_pod "mPaaS_LBS"
43 end
44
45 post_install do |installer|
46   installer.pods_project.targets.each do |target|
47     flutter_additional_ios_build_settings(target)
48   end
49 end
50
```

2. Run the pod install command again.

## 8. FAQ of mPaaS framework

This section describes the mPaaS framework FAQ. Click a question to view its answer.

- The error `ERROR: Failed to build gem native extension.d` occurred when I upgraded RubyGems.
- The error `Library not loaded` occurred when I installed the RVM.
- The error `Lazy symbol binding failed` occurred when I installed the RVM.
- How to use my UIApplication delegate class?
- How to exit all micro applications and return to the Launcher?
- If Application B exists on top of Application A, how can Application B restart Application A and pass arguments?
- After the base class is inherited from DTViewController, the VC created using the XIB shows a white screen after it is opened.

### The error `ERROR: Failed to build gem native extension.d` occurred when I upgraded RubyGems.

If the error `ERROR: Failed to build gem native extension.` occurs when you upgrade RubyGems, install the Xcode command line tool and try again.

```
xcode-select --install
```

### The error `Library not loaded` occurred when I installed the RVM.

If the error `For dyld: Library not loaded: /usr/local/lib/libgmp.10.dylib` occurred when you install Ruby 2.2.4 using the RVM, run the following command and try again.

```
brew update && brew install gmp
```

### The error `Lazy symbol binding failed` occurred when I installed the RVM.

If the error `dyld: lazy symbol binding failed: Symbol not found: _clock_gettime` occurs when you install Ruby 2.2.4 using the RVM, install the Xcode command line tool and try again.

```
xcode-select --install
```

### How to use my UIApplication delegate class?

If you do not use the mPaaS framework, you can override `DFClientDelegate` in the main method with your own class.

### How to exit all micro applications and return to the Launcher?

```
[DTContextGet() startApplication:@"app ID of the launcher" params:nil  
animated:kDTMicroApplicationLaunchModePushNoAnimation];
```

### If Application B exists on top of Application A, how can Application B restart Application A and pass arguments?

Suppose Application A is started, and Application B on its top is also started. In this case, when Application A is restarted, Application B (and all applications on top of Application A) will exit.

```
[DTContextGet() startApplication:@"name of A" params:@{@"arg": @"something"}
launchMode:kDTMicroApplicationLaunchModePushWithAnimation];
```

Meanwhile, `DTMicroApplicationDelegate` of Application A will receive the following event, with arguments carried in `options`.

```
- (void)application:(DTMicroApplication *)application willResumeWithOptions:
(NSDictionary *)options
{
}
}
```

**After the base class is inherited from `DTViewController`, the VC created using the XIB shows a white screen after it is opened.**

Rewrite the `loadView` method in category of `DTViewController`, as shown in the following code.

```
@interface DTViewController (NibSupport)
@end

@implementation DTViewController (NibSupport)

- (void)loadView
{
    [super loadView];
}

@end
```