

Ant Technology

Access iOS
User Guide

Document Version: 20230209

Legal disclaimer

Ant Group all rights reserved©2022.

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

Trademark statement



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

Disclaimer

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

Document conventions

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

Table of Contents

1.Integration method introduction	06
2.Create an application	07
3.Integrate by using CocoaPods based on the existing project	12
4.Advanced guide	19
4.1. mPaaS directory structure	19
4.2. mPaaS iOS framework	20
4.3. mPaaS Micro Applications and Services	32
4.3.1. Create a micro application	32
4.3.2. Create a service	34
4.3.3. Manage the micro application and service	36
4.3.4. Code sample of micro application	38
4.4. iOS language settings	41
4.5. Customize the city selection	43
5.Adaption to iOS	45
5.1. Upgrade guide of mPaaS 10.1.68	45
5.2. Privacy permission	48
5.3. mPaaS 10.1.60 baseline upgrade guide	53
5.4. Adapt to WKWebView	57
5.5. mPaaS 10.1.68 baseline adapt to Xcode 13	63
5.6. mPaaS 10.1.68 baseline adapt to iOS 15	64
5.7. mPaaS 10.1.68 baseline adapt to iOS 14	65
5.8. mPaaS 10.1.60 baseline adapt to iOS 13	66
5.9. mPaaS 10.1.32 baseline adapt to iOS 13	70
6.Reference	74
6.1. Customize the iOS navigation bar	74
6.2. Handle iOS conflict	83

6.3. Switch iOS environment	85
7.FAQ of mPaaS framework	91

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](#).

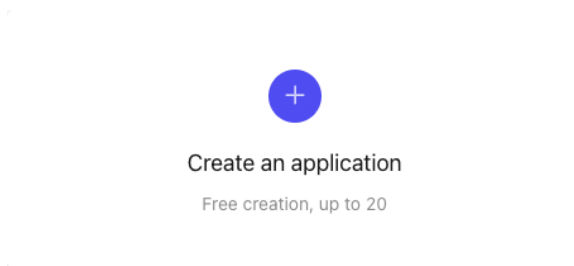
Create an mPaaS app


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

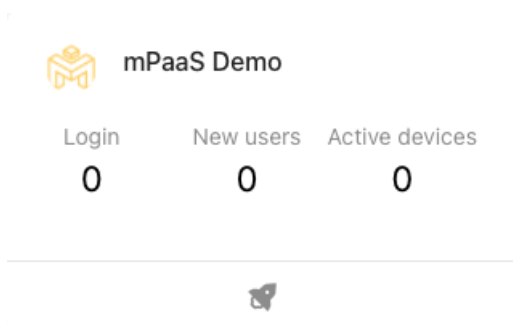
Note

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.



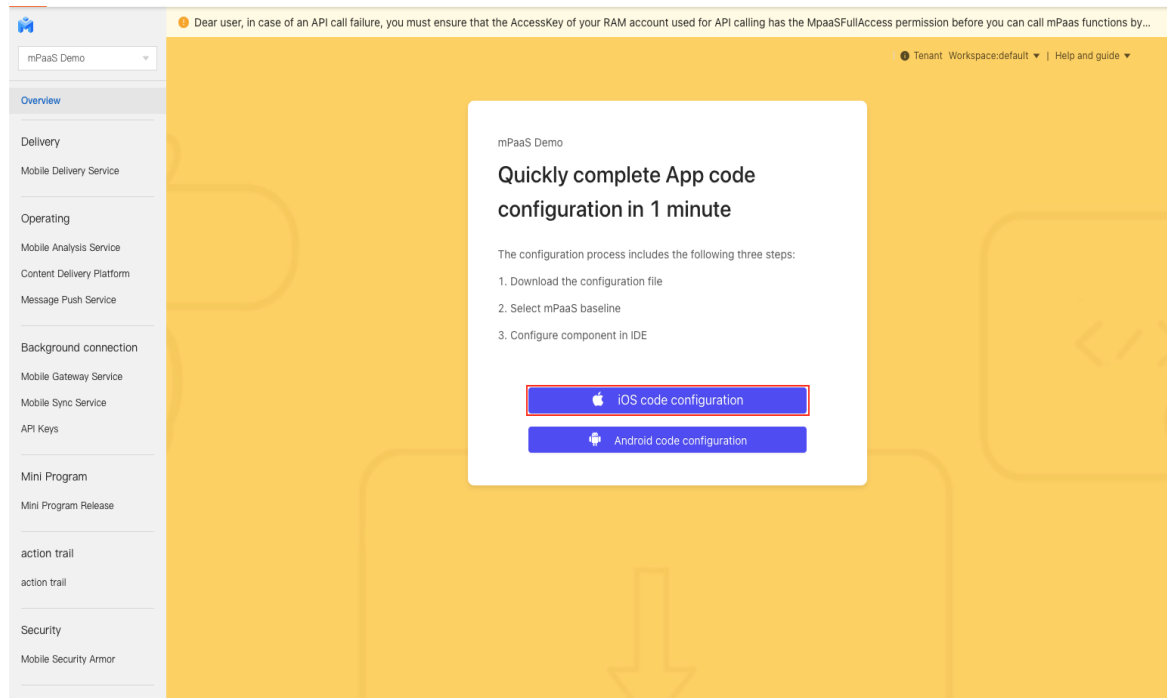
3. Complete the app information.
 - i. Enter the app name. Example: mPaaS Demo.
 - ii. Click  to upload the app icon. You can skip this step, and the app will use the default icon in this case.
4. Click **OK** to finish creating the app. The application you just created is displayed.



Download the configuration file

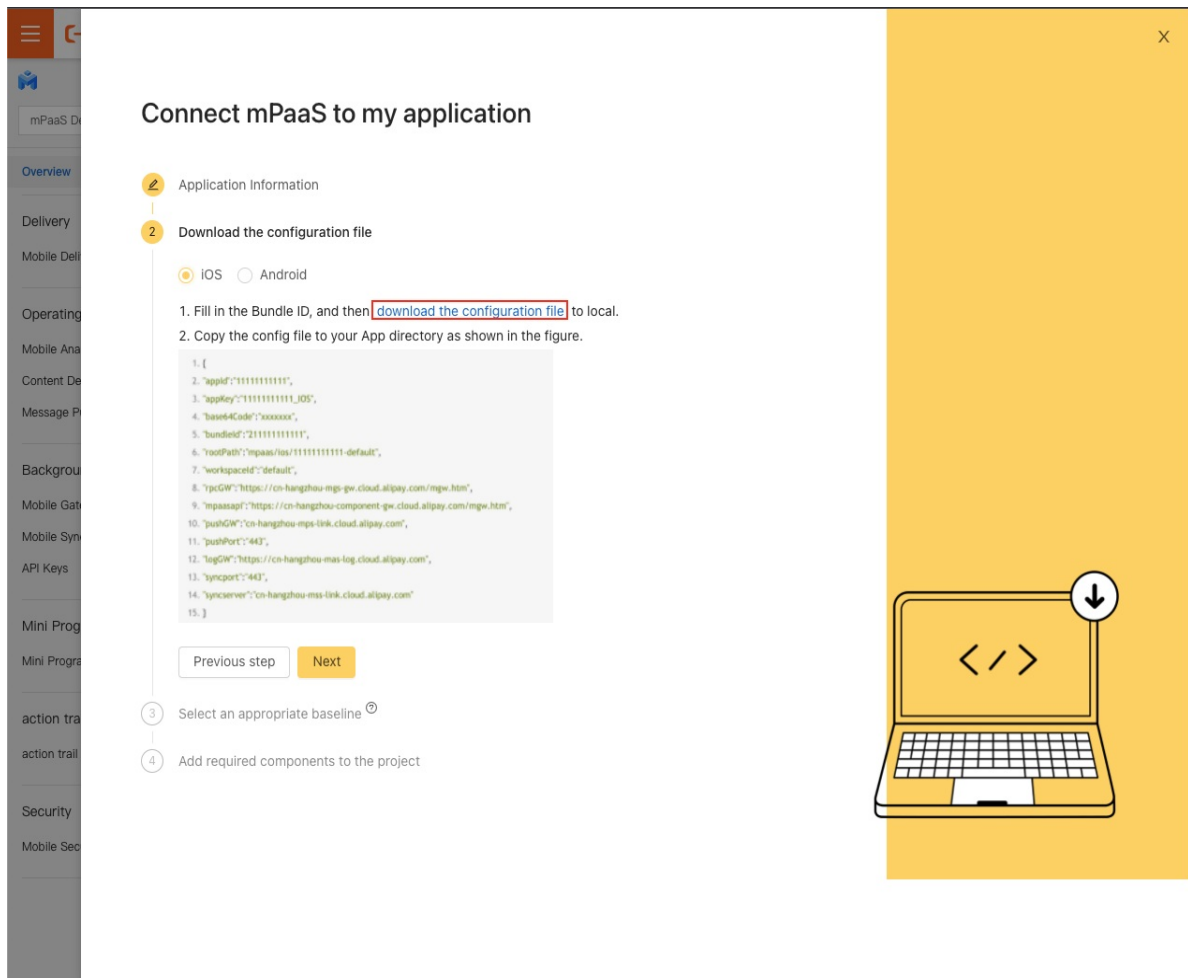
1. On the app list page, click the name of the created app (such as the mPaaS Demo app created in the previous step), and the following page appears:
 - The app name appears at the top left, where you can switch apps.

- The left side of the page displays a list of component services provided by mPaaS.

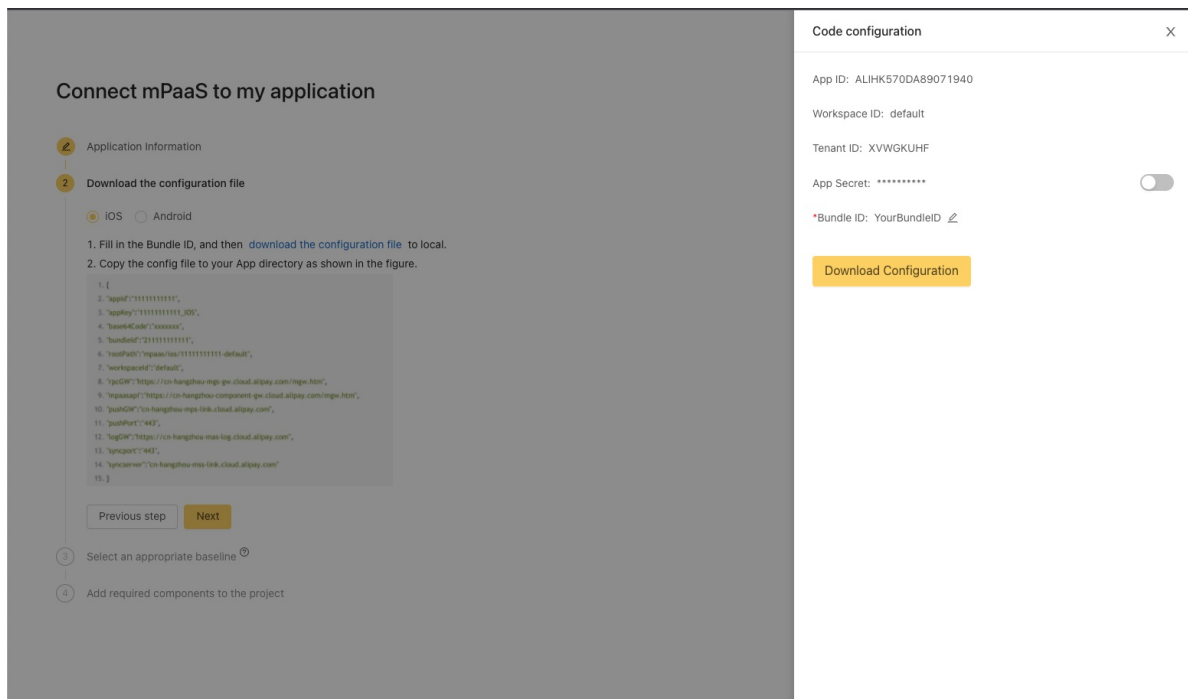


2. Click **iOS code configuration** button to go to the **Connect mPaaS to my application** page.

- On the **Connect mPaaS to my application** page, click **download the configuration file** to go to the **Code Configuration** page.



- 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:

[illegible]

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://code.aliyun.com/mpaas-public/podspecs.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 '10.1.32'`, `10.1.32` 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 "mPaaS_LocalLog"	10.1.32+	Local log.
mPaaS_pod "mPaaS_Log"	10.1.32+	Mobile analysis: behavior log, automation log, Crash log, and performance log analysis.
mPaaS_pod "mPaaS_Diagnosis"	10.1.32+	Diagnosis: Client diagnosis and analysis.
mPaaS_pod "mPaaS_RPC"	10.1.32+	Mobile gateway: provides download, upload, RPC call and other functions.
mPaaS_pod "mPaaS_Sync"	10.1.32+	Mobile synchronization: long connection service.
mPaaS_pod "mPaaS_Push"	10.1.32+	Message push.
mPaaS_pod "mPaaS_Config"	10.1.32+	Switch configuration: pull the corresponding value from the server based on the key to dynamically control the client logic.
mPaaS_pod "mPaaS_Hotpatch"	10.1.32+	Hotpatch: used to fix online bugs without releasing a new version.
mPaaS_pod "mPaaS_Upgrade"	10.1.32+	Upgrade release: provides convenient service to proactively detect and upgrade, which can be used for daily phased release and online new version update reminder.
mPaaS_pod "mPaaS_Share"	10.1.32+	Sharing: supports sharing text and pictures to well-known channels such as Weibo, DingTalk, and Alipay friends.

mPaaS_pod "mPaaS_Nebula"	10.1.32+	HTML5 container and offline package: Nebula container, which supports interaction between the front end and the native.
mPaaS_pod "mPaaS_UTDID"	10.1.32+	Device ID: easily and quickly obtained for the application to find a specific device safely and effectively.
mPaaS_pod "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 "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 "mPaaS_CommonUI"	10.1.32+	Universal UI: universal UI component library that provides various UI components.
mPaaS_pod "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 "mPaaS_AlipaySDK"	10.1.32+	Quick Pay function of Alipay: quick payment platform of Alipay.
mPaaS_pod "mPaaS_Multimedia"	10.1.32+	Multimedia component: supports image download, upload, cache and other functions.

mPaaS_pod "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 "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.

See the following example of complete Podfile:

```

1 # mPaaS Pods Begin
2 plugin "cocoapods-mPaaS"
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 Pods End
6 # -----
7
8 platform :ios, '9.0'
9
10 target 'DemoTemp' do
11
12     mPaaS_pod "mPaaS_Nebula"
13     mPaaS_pod "mPaaS_Hotpatch"
14
15 end
16

```

- Execute `pod mpaas update x.x.x` in the command line, `x.x.x` is the version number of configured baseline, such as `10.1.68`.
- 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`.

- 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

- 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 `10.1.32`.

```
[TT-MAC:demo19072501 tangtian$ pod mpaas update 10.1.32
1. update 10.1.32 baseline file ...
updateTime : 2019-08-16 17:34:19 +0800
The baseline has no change ...
update 10.1.32 baseline file Done

2. update mPaaS repo ...
Updating spec repo 'mPaaS_iOS_specs'
$ /usr/bin/git -C /Users/tangtian/.cocoapods/repos/mPaaS_iOS_specs fetch origin --progress
$ /usr/bin/git -C /Users/tangtian/.cocoapods/repos/mPaaS_iOS_specs rev-parse --abbrev-ref HEAD
master
$ /usr/bin/git -C /Users/tangtian/.cocoapods/repos/mPaaS_iOS_specs reset --hard origin/master
HEAD is now at 139e9f0 PodSpec Update at 2019-08-19 21:39:33
update mPaaS repo Done
```

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

Upgrade the baseline

- 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 `10.1.32` to `10.1.60`. The baseline number supports the standard or custom baseline.

```
1 # mPaaS Pods Begin
2 plugin "cocoapods-mPaaS"
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 Pods End
6 # -----
7
8 platform :ios, '9.0'
9
10 target 'DemoTemp' do
11
12   mPaaS_pod "mPaaS_Nebula"
13   mPaaS_pod "mPaaS_Hotpatch"
14
15 end
```

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

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", :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 2 # This line is maintained by MPaaS plugin automatically. Please don't modify.
6 # mPaaS Pods End
7 # -----
8
```

Parameters	Function	Applicable versions
<code>:only_frameworks => true</code>	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
<code>:check_repo => false</code>	In some cases (such as using an intranet proxy), the default added repo is not automatically checked.	≥ V0.9.5.0.0.2

Command list

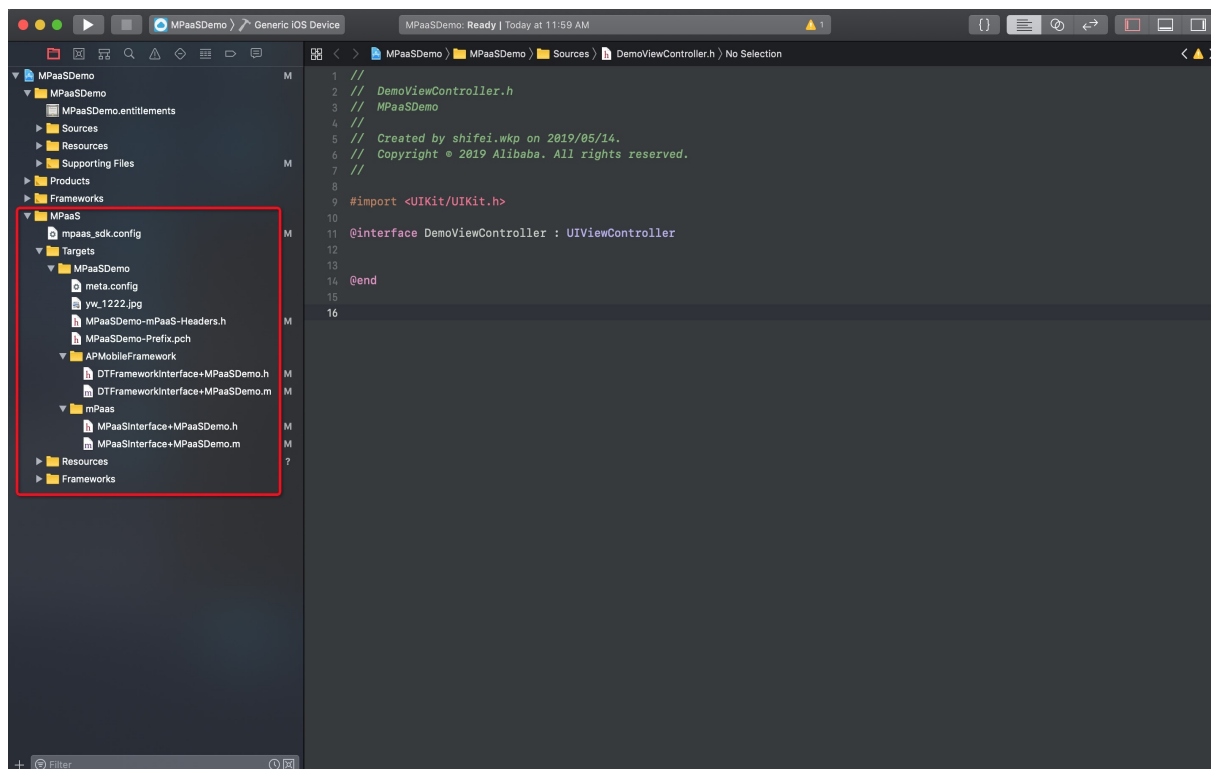
After installing the cocoapods-mPaaS plug-in, you can use command line tools to assist your development.

Command	Function
<code>pod mpaas init</code>	In Podfile , add plugin , source and mPaaS_baseline .
<code>pod mpaas update <VERSION></code>	Upgrade the baseline. The parameter <VERSION> is the specific baseline number such as 10.1.32. Then upgrade the podspec library.
<code>pod mpaas update --all</code>	In the official version of the plug-in, this command will upgrade the plug-in, and run the installation script again. In the beta version of the plug-in, this command will implement features of the official version, and upgrade the local baseline.
<code>pod mpaas info</code>	Show the complete information of the baseline and the corresponding component.
<code>pod mpaas info <NAME> <VERSION></code> , in which <VERSION> is optional.	Filter the information about a module name. For example, <code>pod mpaas info hotpatch 10.1.32</code> .
<code>pod mpaas info --only-mPaaS</code>	Show some default baseline information, which is easy to be pasted to Podfile.
<code>pod mpaas open</code>	Directly open the .xcworkspace file from the command line.
<code>pod mpaas version</code>	Show the complete baseline used in the current project.
<code>pod mpaas version --plugin</code>	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:(NSDictionary *)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 message 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 message
```

```
e 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 didReceiveLocalNotifi
cation:(UINotification *)notification;

/**
* The framework calls back this method in advance, allowing the accessed application to p
rocess the notification message in advance.
* When DTFrameworkCallbackResultContinue is returned, the framework broadcasts the messag
e to global listeners through UIApplicationDidReceiveLocalNotification, and calls completio
nHandler().
* 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 handleActionWithIdent
ifier:(NSString *)identifier forLocalNotification:(UINotification *)notification compl
etionHandler:(void (^)(void))completionHandler;

/**
* The framework calls back this method in advance, allowing the accessed application to g
et deviceToken.
* When DTFrameworkCallbackResultContinue is returned, the framework broadcasts the messag
e 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 didRegisterForRemoteN
otificationsWithDeviceToken:(NSData *)deviceToken;

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

/**
* The framework notifies sharing component (if there is, and shouldAutoactivateShareKit r
eturns YES) in advance, if the sharing component cannot process it, this method is called b
ack to allow the accessed application to process openURL.
* When DTFrameworkCallbackResultReturnYES or DTFrameworkCallbackResultReturnNO is returne
d, the framework directly returns to the system, without executing the subsequent logic.
* When DTFrameworkCallbackResultContinue is returned, the framework continues to process
the URL, and distribute it to SchemeHandler and other classes for further processing.
*
* Comparing with the system method, this method has an additional newURL parameter, allow
ing the application to return a different URL after processing. If the function returns DTF
rameworkCallbackResultContinue and assign value to newURL, the framework will use the new U
RL for subsequent processing.
*/
```

```
^ /
- (DTFrameworkCallbackResult)application:(UIApplication *)application openURL:(NSURL *)url
newURL:(NSURL **)newURL sourceApplication:(NSString *)sourceApplication annotation:(id)anno
tation;

/**
 * The framework calls back this method in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework continues to execute,
no other logic currently.
 * When DTFrameworkCallbackResultReturn is returned, the framework stops executing the sub
sequent 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 execute,
no other logic currently.
 * When DTFrameworkCallbackResultReturn is returned, the framework stops executing the sub
sequent 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 execute,
no other logic currently.
 * When DTFrameworkCallbackResultReturn is returned, the framework stops executing the sub
sequent 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 execute
and give event to sharing component (if there is, and shouldAutoactivateShareKit returns YE
S). If the entire application is not loaded, BootLoader is called.
 * When DTFrameworkCallbackResultReturn is returned, the framework stops executing the sub
sequent 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 execute,
no other logic currently.
 * When DTFrameworkCallbackResultReturn is returned, the framework stops executing the sub
sequent 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 execute,
no other logic currently.
```

```
no other logic currently.
* When DTFrameworkCallbackResultReturn is returned, the framework stops executing the sub
sequent logic, no other logic currently.
*/
- (DTFrameworkCallbackResult)applicationDidReceiveMemoryWarning:(UIApplication *)applicatio
n;

/**
 * The framework calls back this method in advance, allowing the accessed application to p
rocess the Watch message in advance.
 * When DTFrameworkCallbackResultContinue is returned, the framework broadcasts the Watch
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 handleWatchKitExtensi
onRequest:(NSDictionary *)userInfo reply:(void (^)(NSDictionary *replyInfo))reply;

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

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

/**
 * Background Fetch mechanism callback
 * completionHandler must be called back in 30 seconds, otherwise the process will be term
inated.
 * To enable this mechanism, you need to configure the fetch option of Background Modes, a
nd then call the following method in didFinishLaunching. See documentation for more informa
tion.
 * [application setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalMini
mum];
 * 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**.

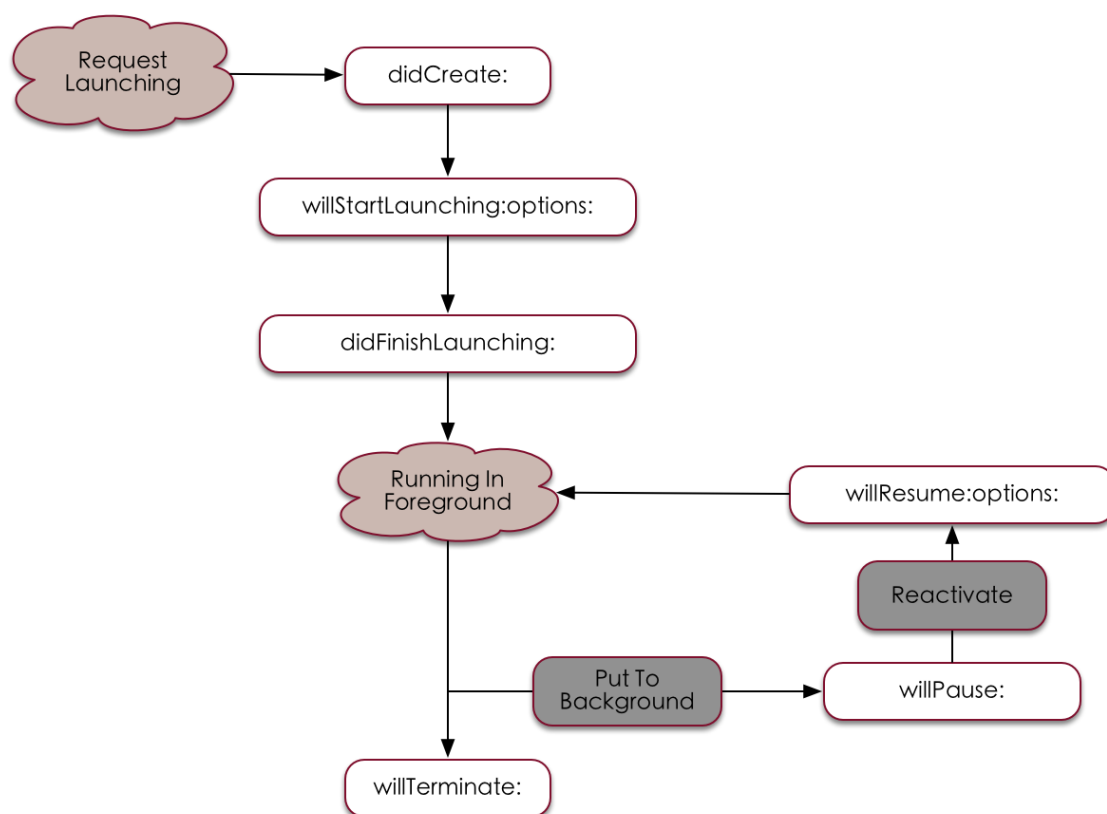
Terminology	Definition
MicroApplication	A micro application with UI on client at runtime
Service	Lightweight abstracted service provided by client at runtime
Context	Context of client micro-component at runtime

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 background.
 *
 * @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 version.
 */
- (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**, the 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:

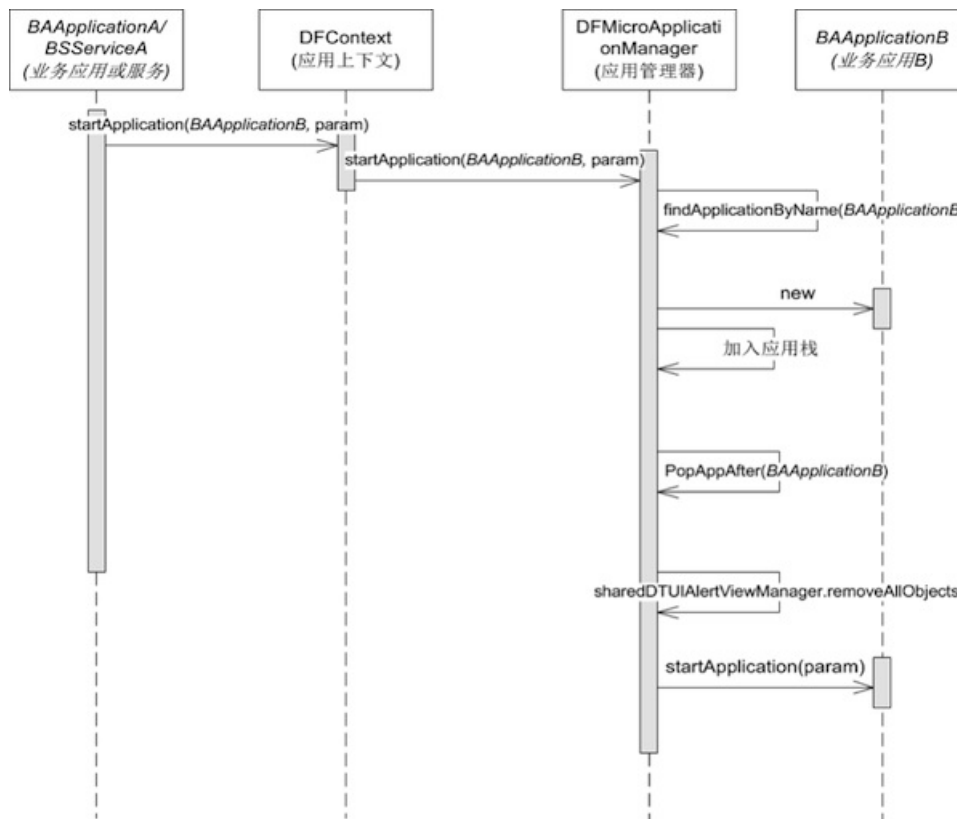
- Provide an interface for starting micro application. Users can quickly find, close and manage the jumps of the micro application through using name;
- Provide an interface for starting service, managing the registration, discovery and deregistration of services.

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 applica
tion 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)anim
ated;
/**
 * 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 applica
tion 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:(DTMicro
ApplicationLaunchMode)launchMode;
/**
 * Find the specified application.
 *
 * @param name: Name of the application to find.
 *
 * @return: Return corresponding application object if the specified application is in the a
pplication stack, otherwise nil.
 */
- (DTMicroApplication *)findApplicationByName:(NSString *)name;
/**
 * Return the application which is on the top of the stack currently, namely the application
visible to users.
 *
 * @return: Current visible applications.
 */
- (DTMicroApplication *)currentApplication;
```

- Micro application starting process:



Service management

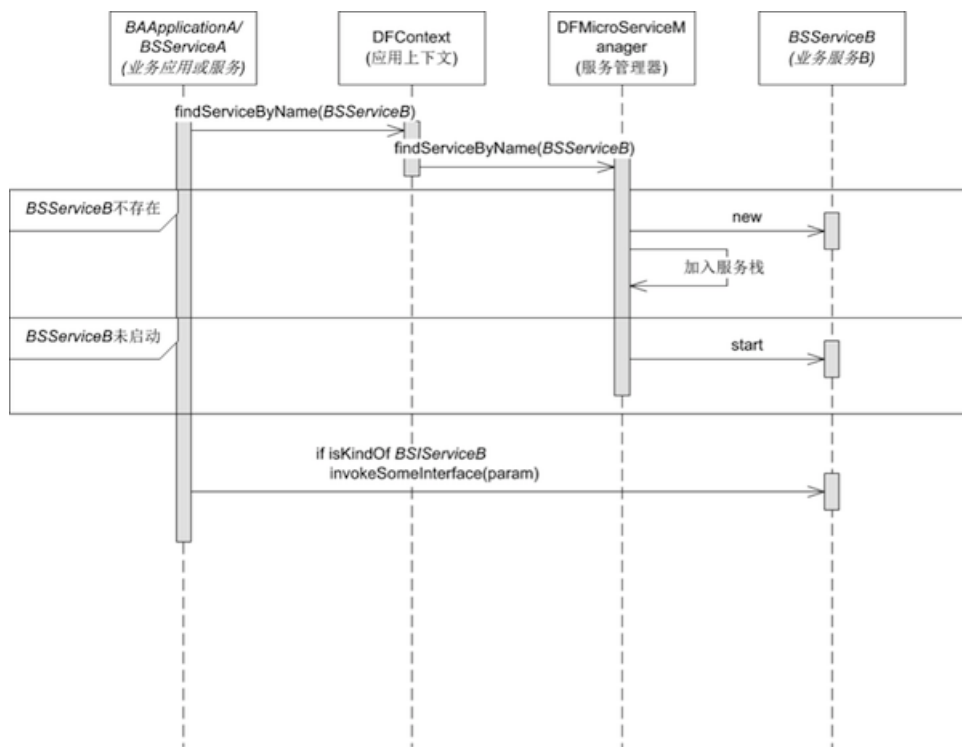
- Relevant interfaces of service management (see `DTContext.h` file for more information):

```
/**
 * 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;

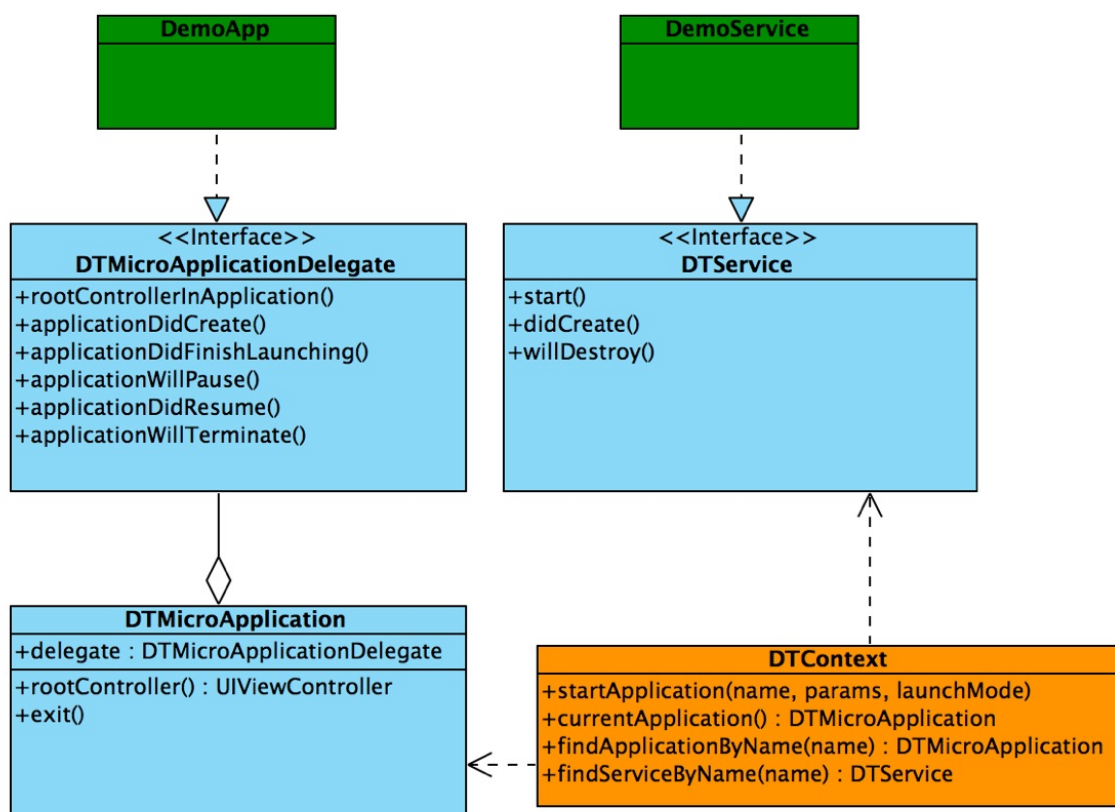
/**
 * Register a service.
 *
 * @param name: Service name
 */
- (BOOL)registerService:(id)service forName:(NSString *)name;

/**
 * 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 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.

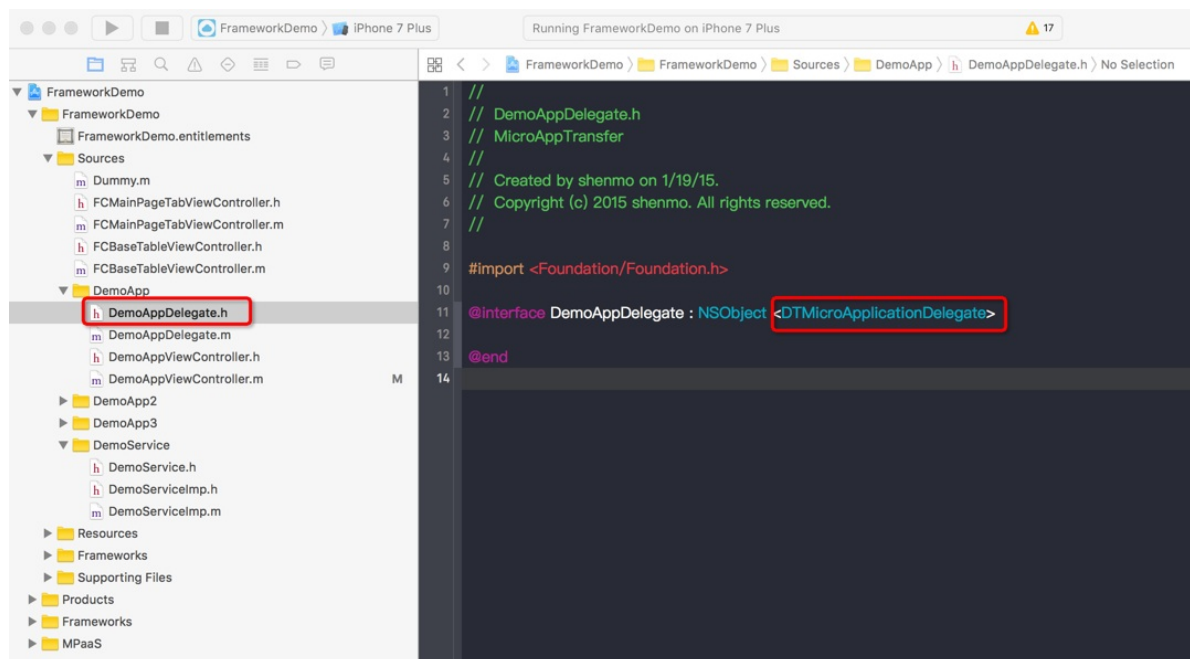
Sample code

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

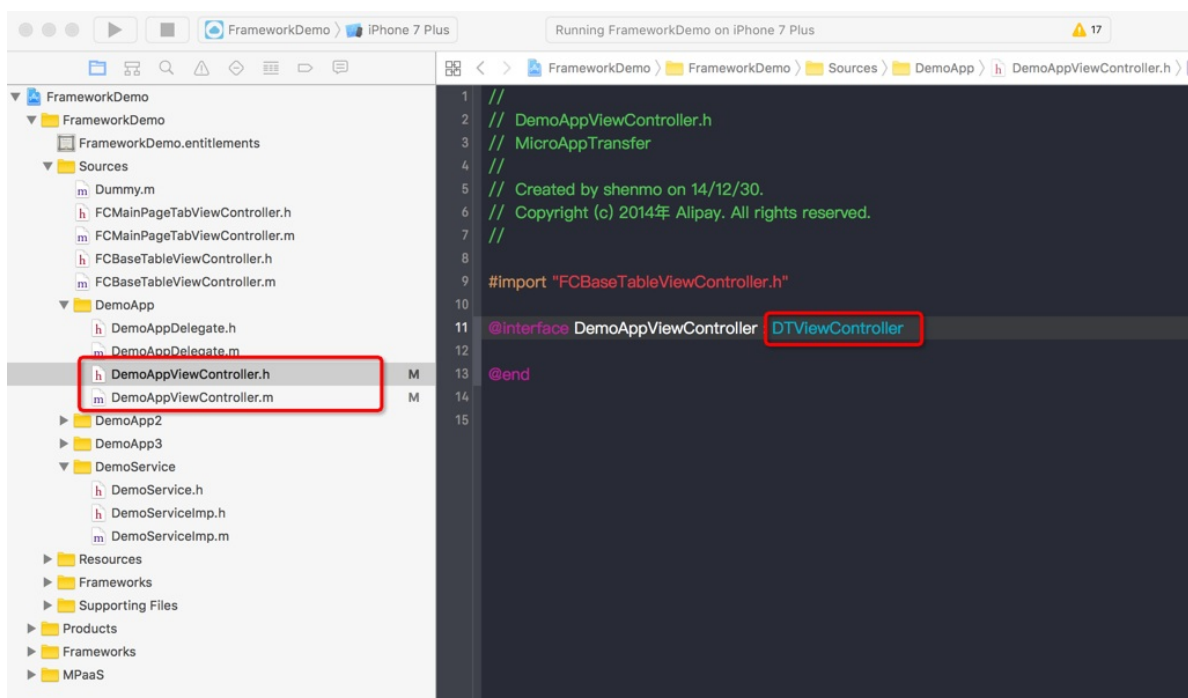
Procedure

1. Add micro-application template code.

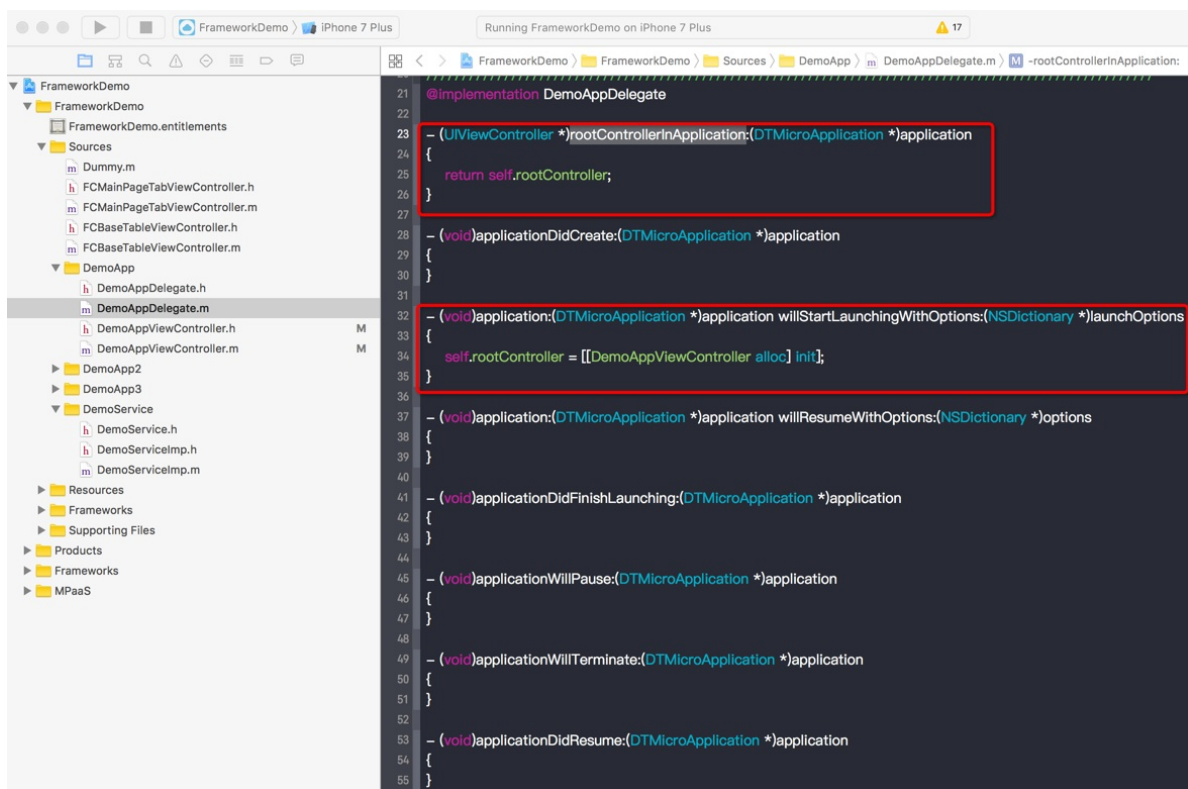
1. Create a micro-application delegation class and implement the delegation method of the micro-application manager of the mPaaS iOS framework, `DTMicroApplicationDelegate`.



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

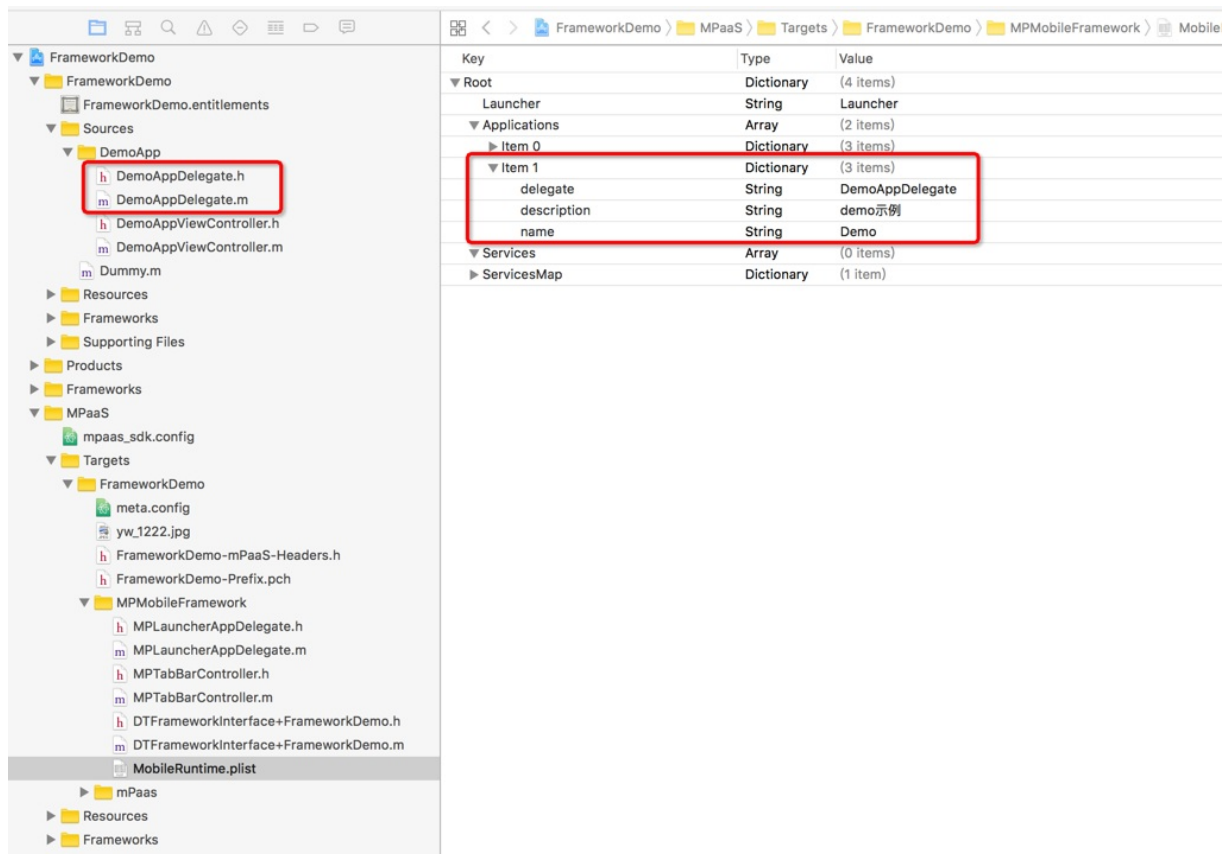


3. Specify `rootViewController` for the micro application. You can perform service actions in the lifecycle of the micro application by using its delegation method.



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.

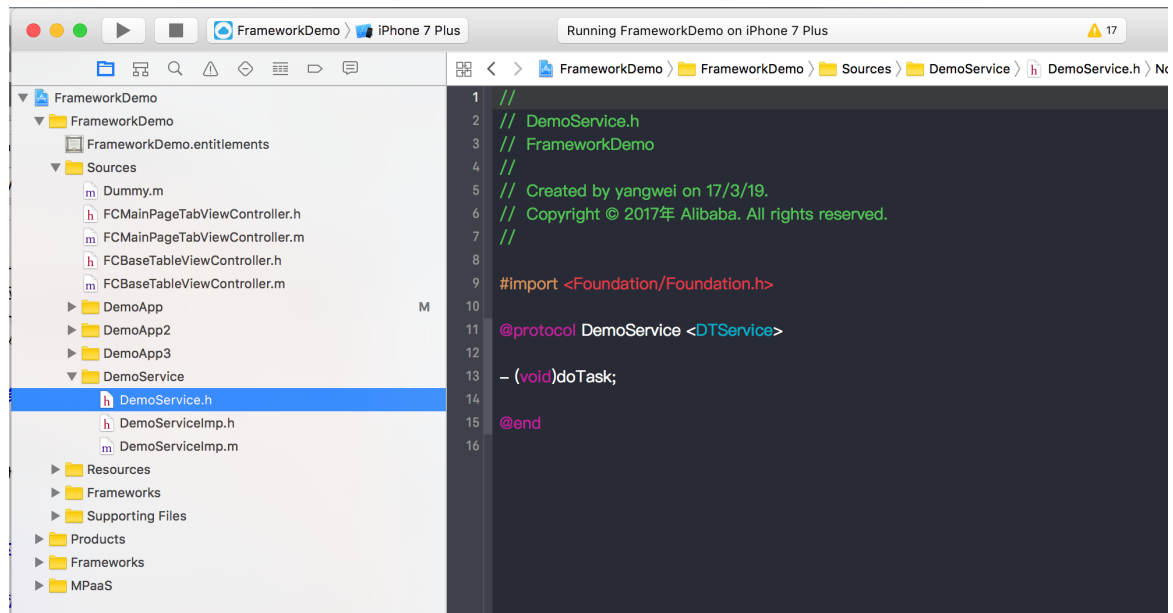
4.3.2. Create a service

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.

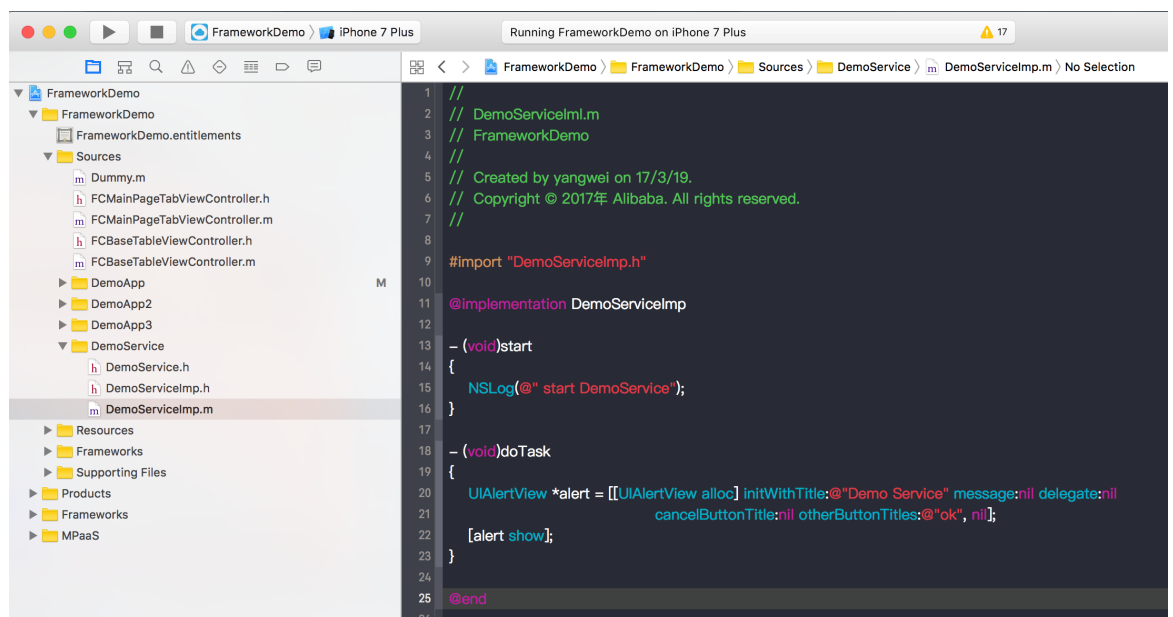
Procedure

1. Add service template codes.

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

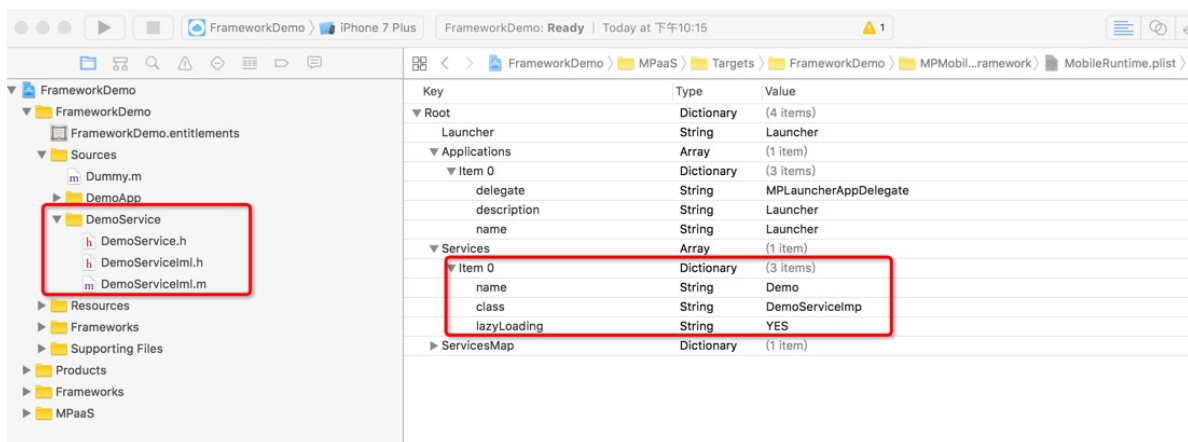


ii. Define the class to implement the service interface method.



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`.



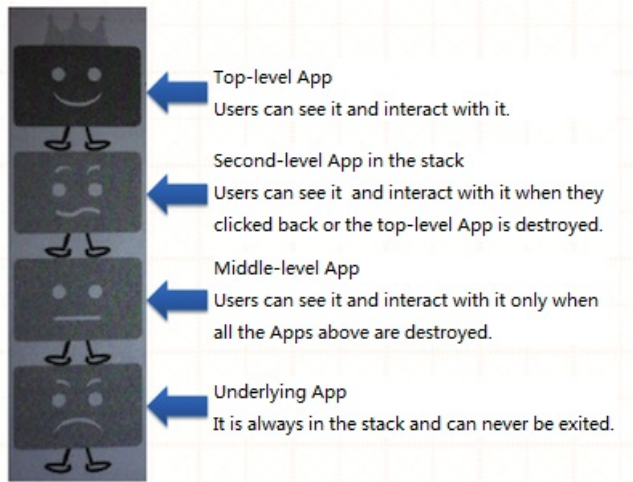
Field	Description
Name	The unique identifier of a service.
Class	The implementation class of service. When creating a service, the framework utilizes a run-time reflection mechanism to create the instances of the service implementation class.
lazyLoading	Whether to delay loading. If Yes, the service 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:kDTMicroApplicat
ionLaunchModePushWithAnimation];
}
```

- 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 upper-
level applications and back to Launcher
    [DTContextGet() startApplication:@"Launcher" params:nil animated:kDTMicroApplicatio
nLaunchModePushNoAnimation];
}
```

- 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 its w
indows 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](#).

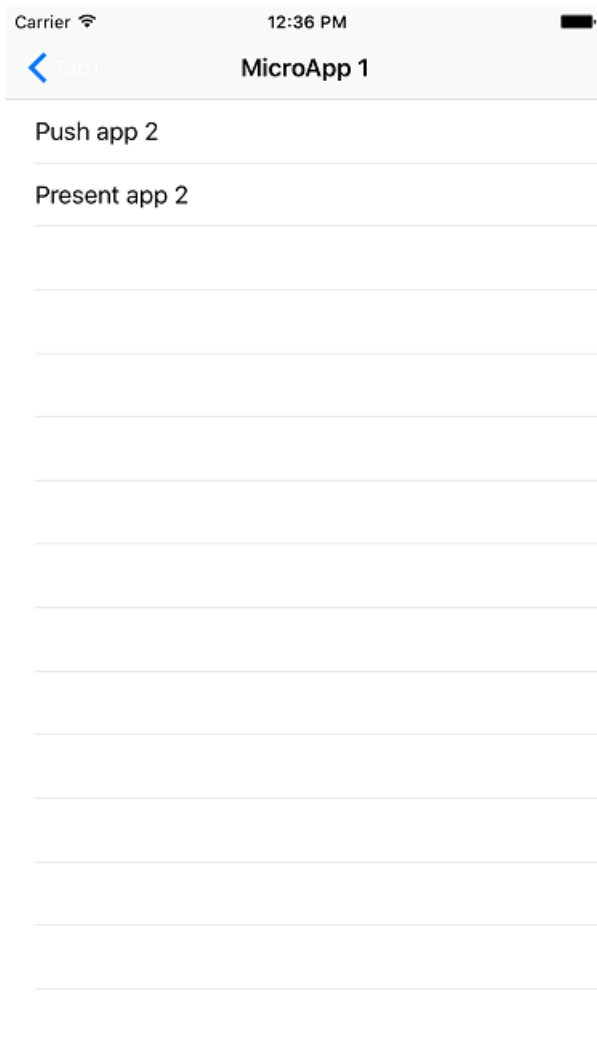
Download codes

Sign in [console](#) and download the Demo codes locally.

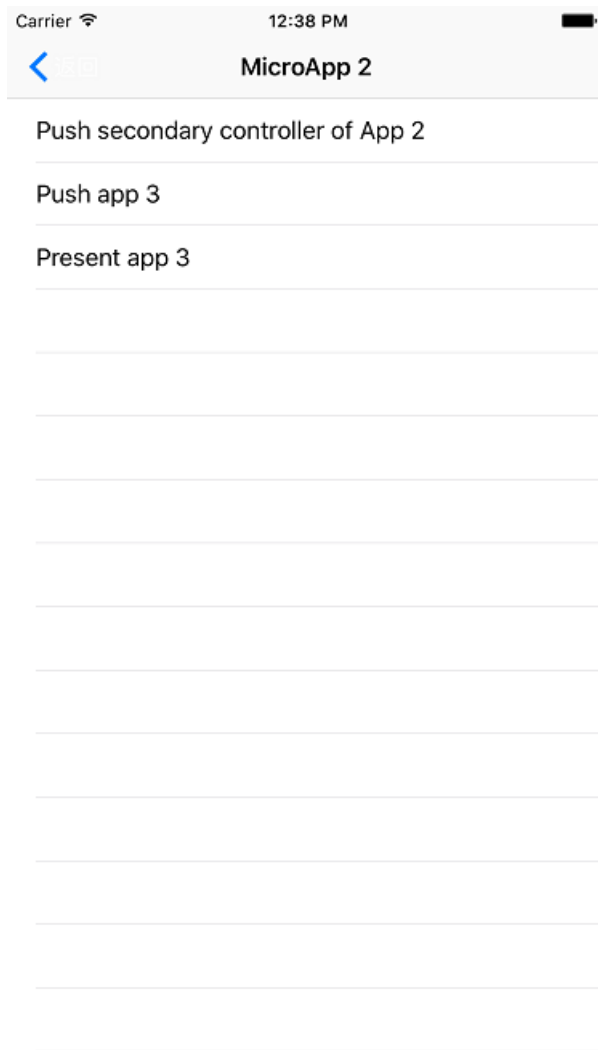
```
git clone git@git.cloud.alipay.com:mPaaS-Demos/FrameworkDemo.git
```

Micro application hierarchy demonstration

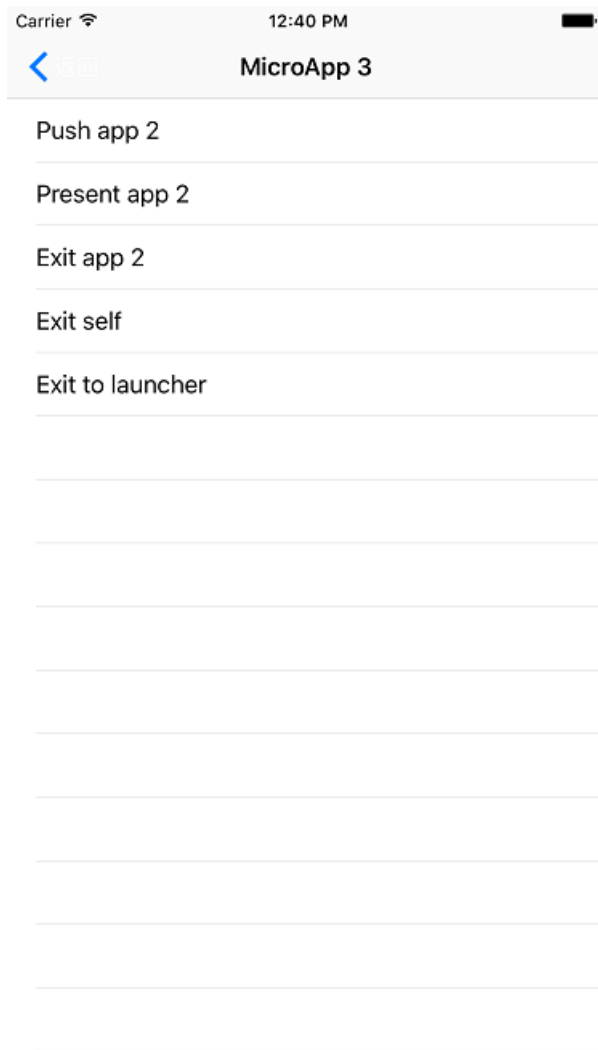
- Click **Push app** to start MicroApp 1;



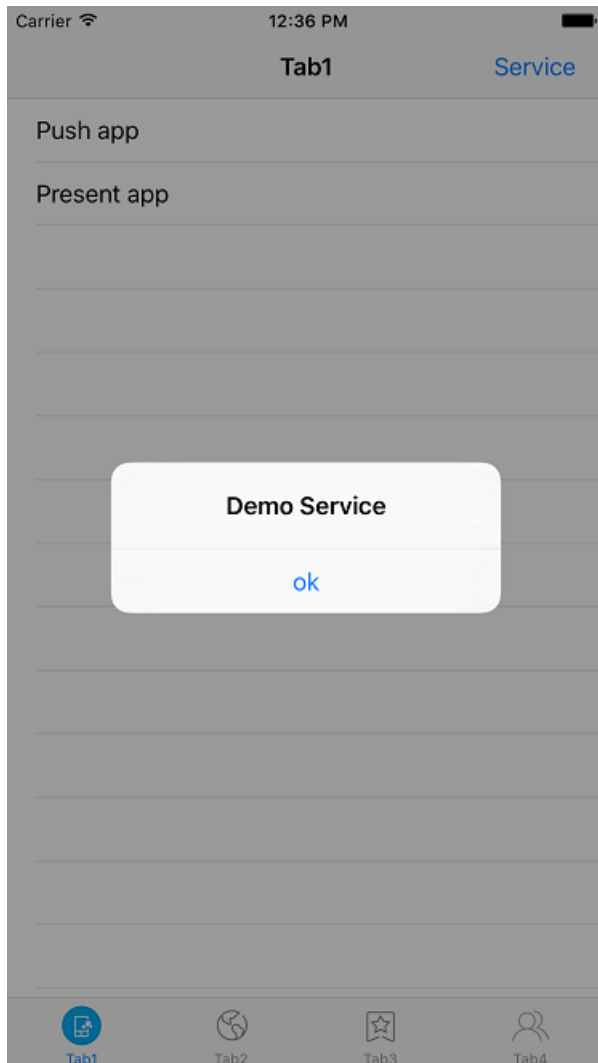
- Click **Push app 2** to start MicroApp 2;



- 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 introduces 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];
```

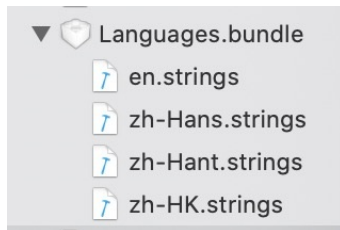
Obtain the current app language

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:



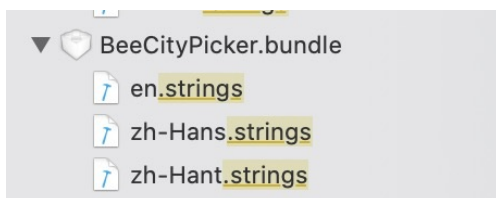
You can modify the current app language by using the following command:

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

Support multilingual copywriting

1. Add multilingual bundle files.

- Add the strings files corresponding to the languages currently supported by the app.



- 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.

Portal > Pods > Resources > BeeCityPicker.bundle > zh-Hans.strings > No Selection

```
1 "BeeCityPicker:地区" = "地区";
2 "BeeCityPicker:定位到的位置" = "定位到的位置";
3 "BeeCityPicker:返回" = "返回";
4 "BeeCityPicker:全部" = "全部";
5 "BeeCityPicker:无法获取当前地理位置" = "无法获取当前地理位置";
6 "BeeCityPicker:已选地区" = "已选地区";
7 // "BeeCityPicker:输入城市名、拼音或首字母查询" = "输入城市名、拼音或首字母查询";
8 "BeeCityPicker:城市选择" = "城市选择";
9 // "BeeCityPicker:你所在城市" = "你所在城市";
10 // "BeeCityPicker:当前" = "当前";
11 "BeeCityPicker:热门城市" = "热门城市";
12 // "BeeCityPicker:热门" = "热门";
13 // "BeeCityPicker:历史访问城市" = "历史访问城市";
```

- 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:城市选择".

Portal > Pods > Resources > BeeCityPicker.bundle > en.strings > No Selection

```
1 //Written by 多语言翻译管理系统
2 //Thu Jun 16 2016 08:15:53 GMT+0800 (CST)
3 //如有问题请联系去疾
4 "BeeCityPicker:全部" = "All";
5 "BeeCityPicker:地区" = "Region";
6 "BeeCityPicker:城市选择" = "Select";
7 "BeeCityPicker:定位到的位置" = "Position located";
8 "BeeCityPicker:已选地区" = "Selected";
9 "BeeCityPicker:无法获取当前地理位置" = "Unable to find your location";
10 "BeeCityPicker:返回" = "Back";
11 "BeeCityPicker:热门城市" = "Popular Cities";
12 "BeeCityPicker:重新定位" = "Click to try again";
13
14
```

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

- 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.text`. The SDK will automatically read the file name:

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

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.text";
```

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. 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. For details, see [Apple's official announcement](#). Upgrade to the 10.1.68 baseline as soon as possible to adapt to WKWebView.

News and Updates

[News](#) [Releases](#)

Updating Apps that Use Web Views

December 23, 2019

If your app still embeds web content using the deprecated UIWebView API, we strongly encourage you to update to WKWebView as soon as possible for improved security and reliability. WKWebView ensures that compromised web content doesn't affect the rest of an app by limiting web processing to the app's web view. And it's supported in iOS and macOS, and by Mac Catalyst.

The App Store will no longer accept new apps using UIWebView as of April 2020 and app updates using UIWebView as of December 2020.

[Learn more about 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.
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 Pod file.
```

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](#)
- [Use mPaaS Plug-in](#)

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 `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](#).

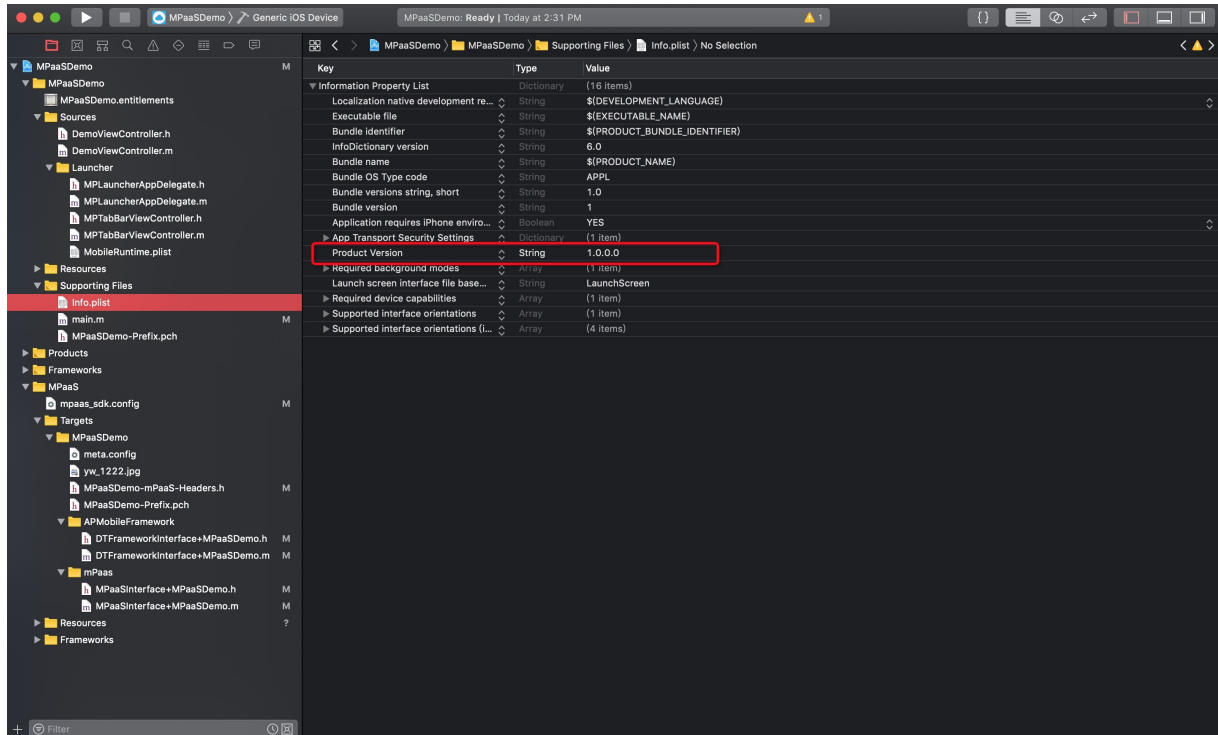
Change of Info.plist

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 10.1.68 baseline incorporate customization requirements. However, if your 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.

5.2. 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 MPAA  
S 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`.



****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: <
116
117 /**
118 * 需要显示隐私权限弹框时，框架会自动回调该方法禁止mPaaS启动流程
119 * @param completionHandler 隐私权限弹框后的继续启动mPaaS框架的回调，
120 *
121 * @return 是继续让框架执行，还是直接给系统返回YES或NO
122 */
123 - (DTFrameworkCallbackResult)application:(UIApplication *)application privacyAuthDidFinishLaunchingWithOptions:(NSDictionary
 *)launchOptions completionHandler:(void (^)(void))completionHandler;
124
```

## Sample code

```
``objective-c
- (DTFrameworkCallbackResult)application:(UIApplication *)application privacyAuthDidFinishL
aunchingWithOptions:(NSDictionary *)launchOptions completionHandler:(void (^)(void))complet
ionHandler
{
 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 didFinishLaunchingWithOptions:(NSDictionary *)launchOptions` method. The code sample is as follows:

```
- (void)application:(UIApplication *)application didFinishLaunchingWithOptions:(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.



```
67 /**
68  * 是否由mPaas用户控制隐私权限弹框处理逻辑。
69  *
70  * @return YES 允许，否则不允许。默认返回NO，即mPaas业务方不处理
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 afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions` method. The code sample is as follows:

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

    ...
}
```

5.3. 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-in
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](#).

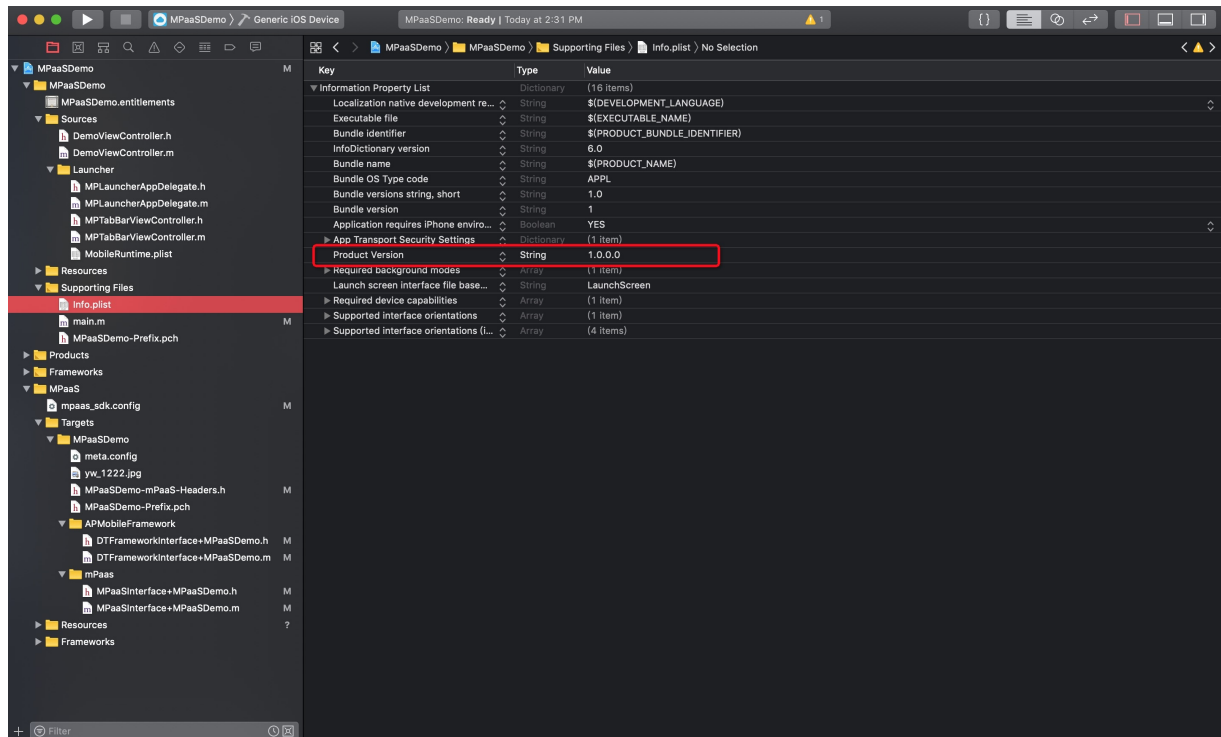
Change of Info.plist

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.

5.4. 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.

We identified one or more issues with a recent delivery for your app. Your delivery was successful, but you may wish to correct the following issues in your next delivery:

ITMS-90809: Deprecated API Usage - Apple will stop accepting submissions of apps that use UIWebView APIs . See <https://developer.apple.com/documentation/uikit/uiwebview> for more information.

After you've corrected the issues, you can use Xcode or Application Loader to upload a new binary to App Store Connect.

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. For details, See [Apple's official announcement](#).

News and Updates

News Releases



Updating Apps that Use Web Views

December 23, 2019

If your app still embeds web content using the deprecated UIWebView API, we strongly encourage you to update to WKWebView as soon as possible for improved security and reliability. WKWebView ensures that compromised web content doesn't affect the rest of an app by limiting web processing to the app's web view. And it's supported in iOS and macOS, and by Mac Catalyst.

The App Store will no longer accept new apps using UIWebView as of April 2020 and app updates using UIWebView as of December 2020.

[Learn more about WKWebView >](#)

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

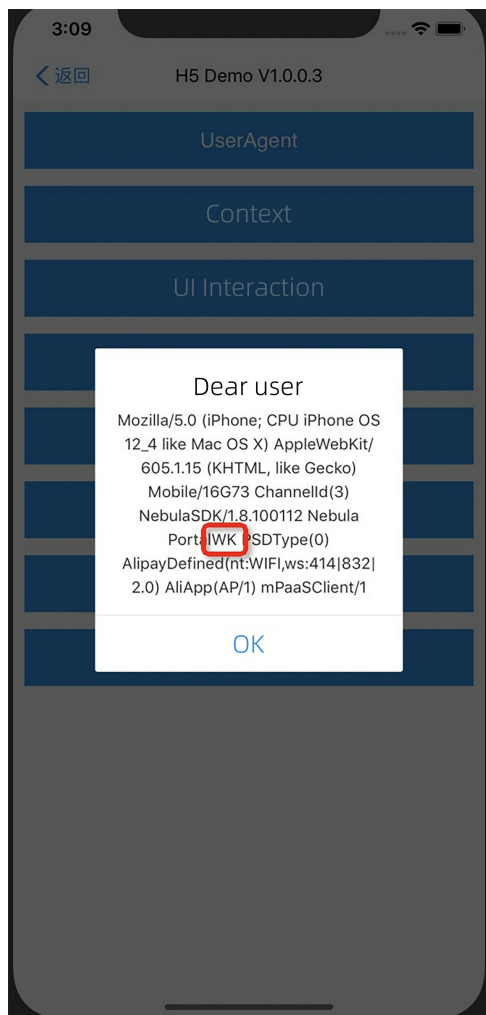
mPaaS containers use UIWebView to load HTML5 pages by default. The mPaaS framework supports two WKWebView enablement methods: **global enablement** and **canary deployment enablement**.

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

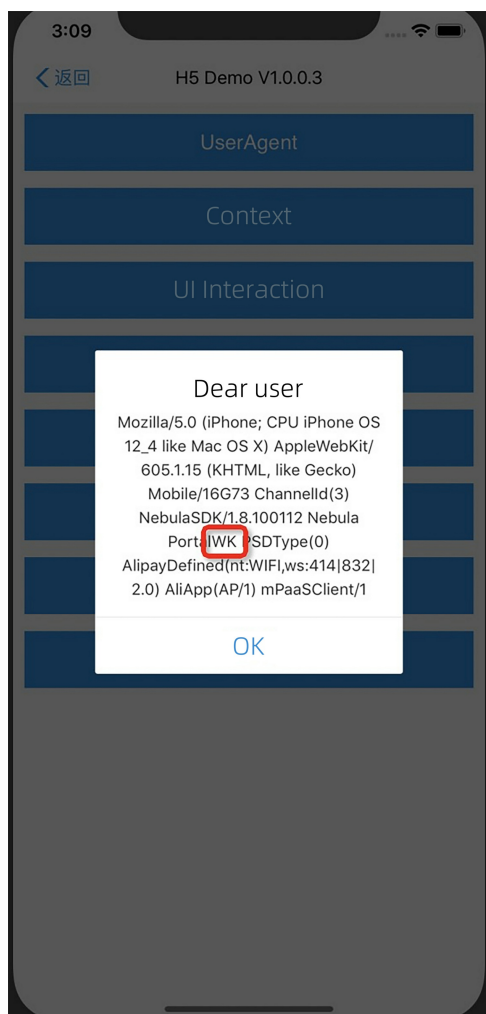
After WKWebView is globally enabled, in order to ensure the functional stability of online HTML5 pages, the mPaaS framework provides **online stop-loss** capabilities to help you quickly switch WKWebView to UIWebView. The procedure is as follows.

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

| | | | |
|---------------|-------|--|---|
| 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 | appld | WKWebView is not used for all HTML5 pages that match the appld 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.



5.5. 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

5.6. 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 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 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.

5.7. 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.

5.8. 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](#).

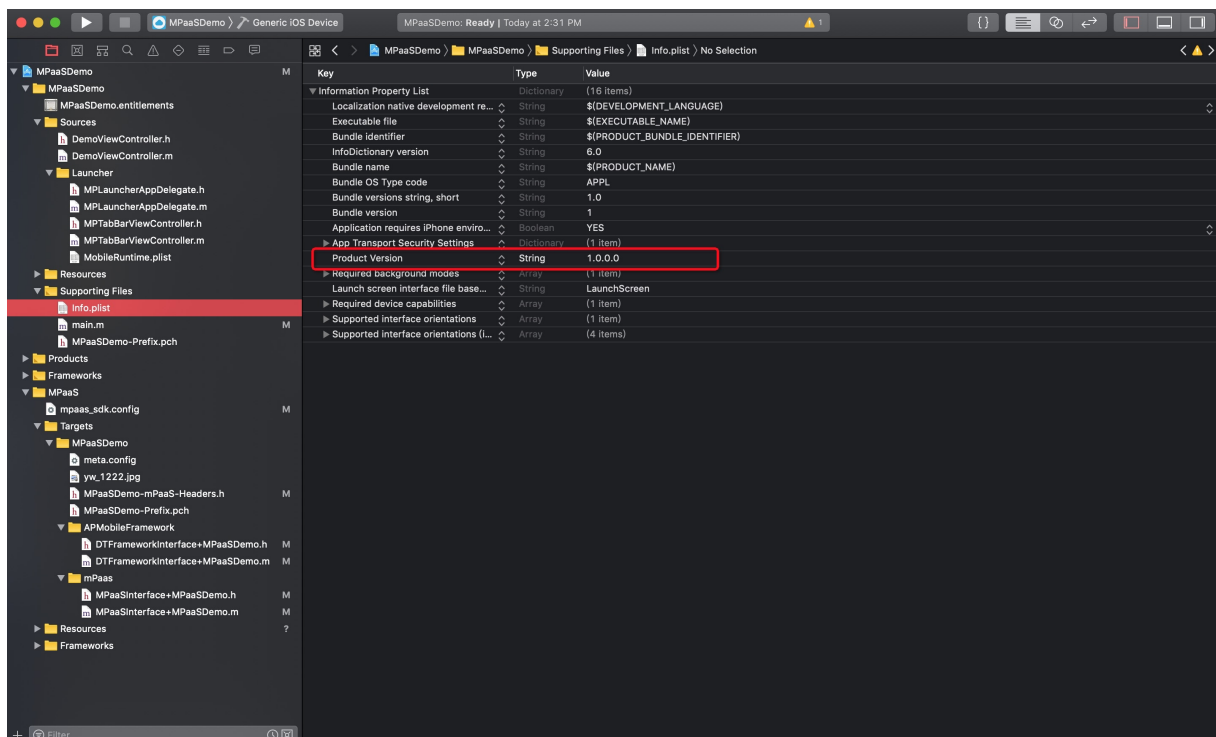
Change of Info.plist

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

5.9. 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](#).

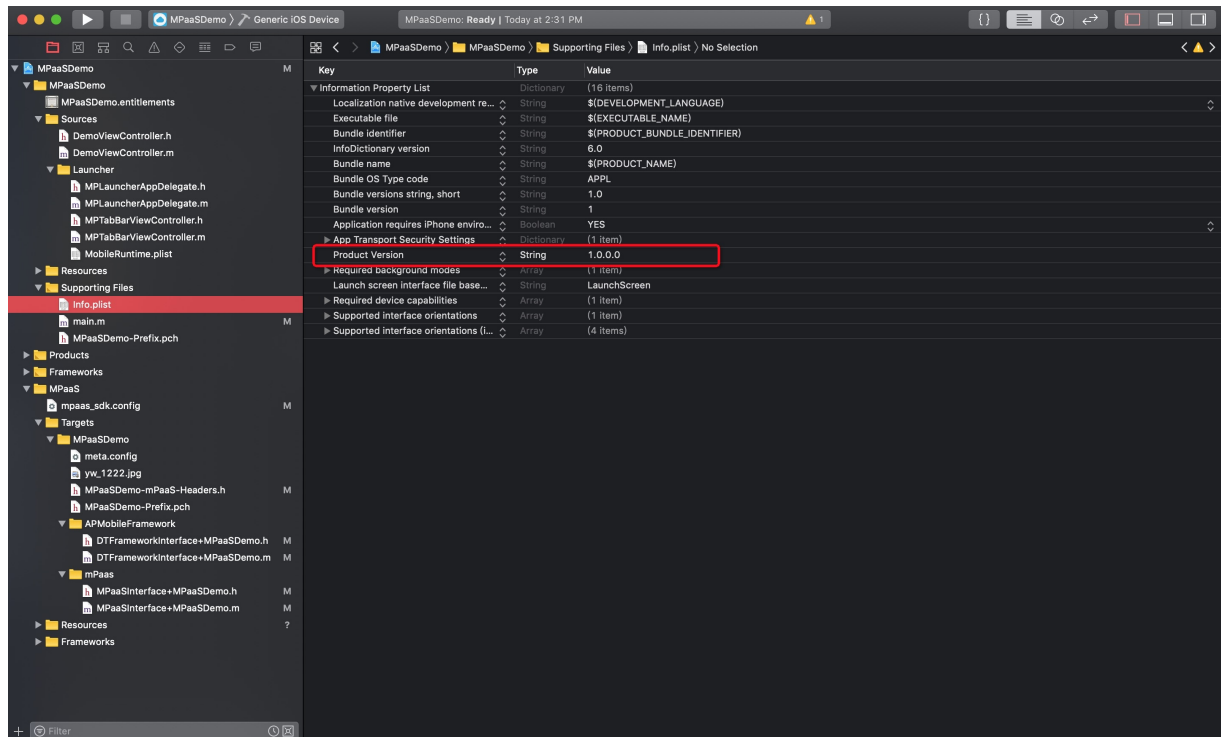
Change of Info.plist

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
- Ant UI
- NebulaSDK
- TinyAppCommon
- MPPromotion

6.Reference

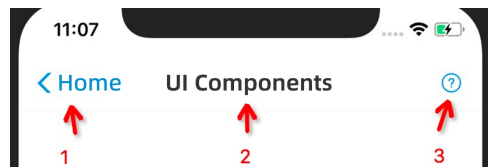
6.1. Customize the iOS navigation bar

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

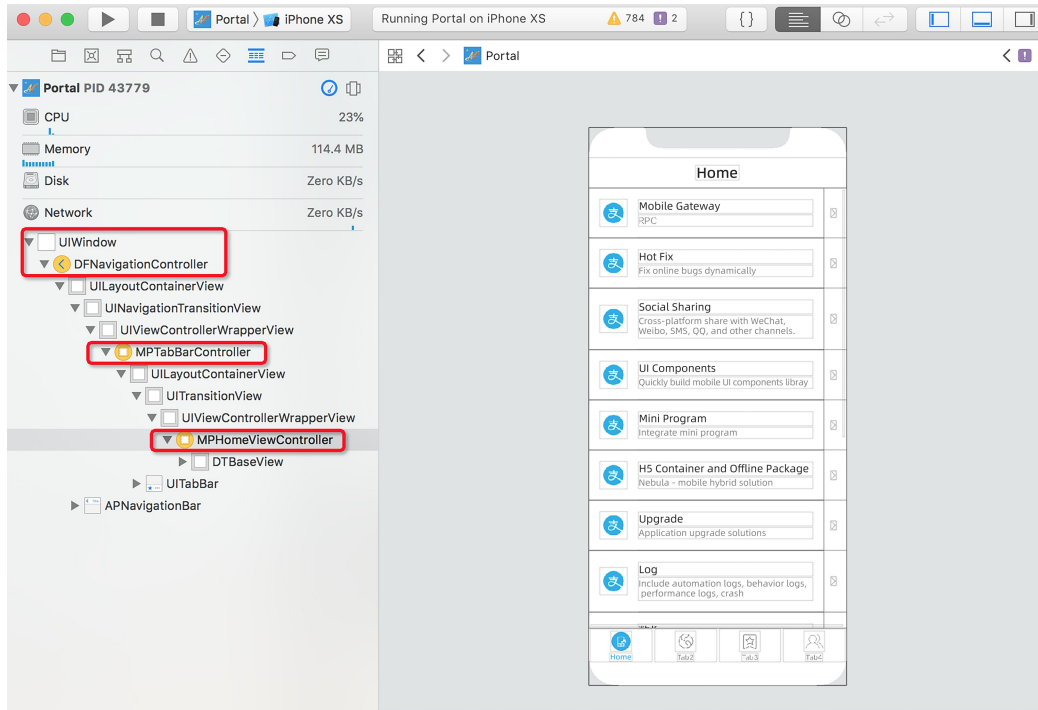
Navigation bar elements are mainly distributed in three areas. Generally, navigation bar customization involves modifying these areas.



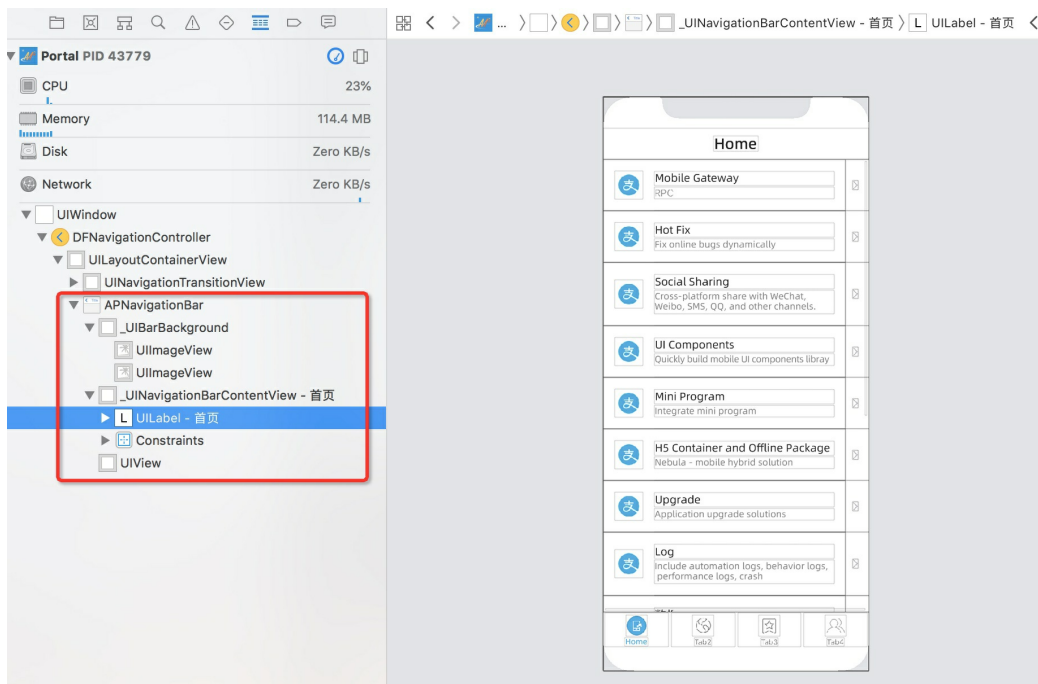
1. Back: The Back button control area that is created by the base class of the mPaaS page. The default format is back arrow + "Back" .
2. Title/Subtitle: The title bar control area, which is not displayed by default. To display this area, call the system method to set the title of the current page.
3. 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 `DTViewControoler`, 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(AUExtendingInfo)
/* 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 theme.
 * 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 color
    dict[TITLEBAR_TITLE_TEXTCOLOR] = [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 sepa
    ration 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 s
    ize 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 ap
ps.
+ (UIImage *)au_default_backButtonlmg;

@end
```

- Sample code

```
@implementation AUBarButtonItem (CGBBarButtonItem)

+ (UIImage *)au_default_backButtonImg
{
    // Customize the Back icon
    return APCommonUILoadImage(@"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.
```

- 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 APCommonUILoadImage(@"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 setNavigationBarTitleAttributesWithTextColor:[UIColor blackColor];}
- (void)gotoTwoTitle{ // Modify the title style: title and subtitle self.navigationController.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.view.width-100, 64); self.navigationController.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 = [UIBarButtonItem barButtonItemWithImageType:
    UIBarButtonItemGroupChat target:self action:@selector(onClickRightItem)];
}

- (void)gotoSetTwoOptionsMenu
{
    // Set two buttons on the right side
    UIBarButtonItem *item1 = [UIBarButtonItem barButtonItemWithImageType: UIBarButtonItemGroupChat target:self action:@selector(onClickRightItem)];
    UIBarButtonItem *item2 = [UIBarButtonItem barButtonItemWithImageType: UIBarButtonItemHelp 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:[AUBarButto
nItem class]]) {
            AUBarButtonItem *backItem = leftBarButtonItems[0];
            backItem.titleColor = [UIColor blackColor];
        }
    }
}

- (void)gotoUpdateTitleColor
{
    // Modify the title color
    [self.navigationController.navigationBar setNavigationBarTitleTextAttributesWithT
extColor:[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. |
| OpenSSL | APOpenSSL | The mPaaS version, which optimizes the original national secret algorithm, must be used. For more details, please refer to How to solve the OpenSSL conflict in iOS projects . |
| 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 |
| | |

| Components supported by remove_pod | Open-source libraries contained |
|------------------------------------|---------------------------------|
| mPaaS_Security | SecurityGuard
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 10.1.68 baseline.

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

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

```
→ ~ pod mpaas version --plugin
0.9.5.0.0.5-beta
→ ~
```

2. Run the `pod mpaas update 10.1.68` 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:

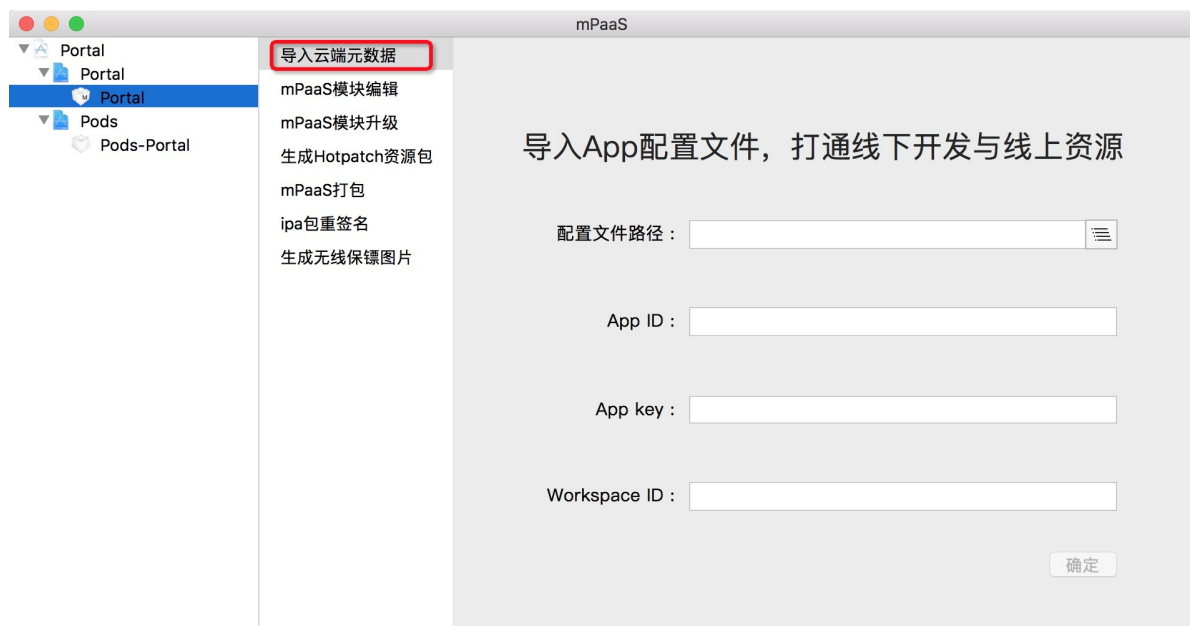
- Static workspace switching
- Dynamic workspace switching

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

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

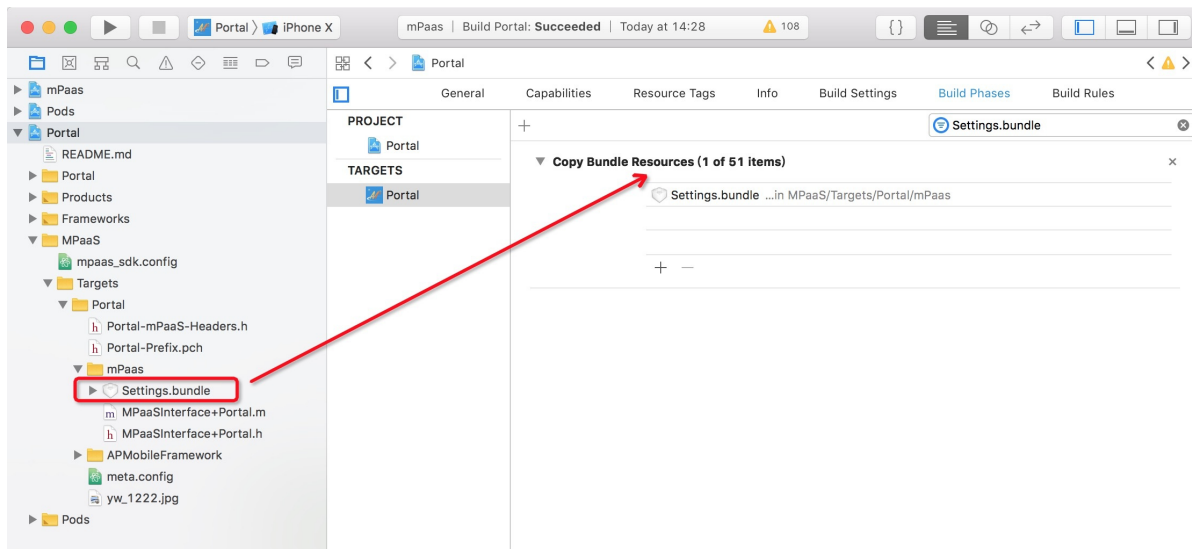
1. Enable dynamic workspace switching. Specifically, rewrite the `enableSettingService` method in category of `MPaaSInterface` and return YES.

```
@implementation MPaaSInterface (Portal)

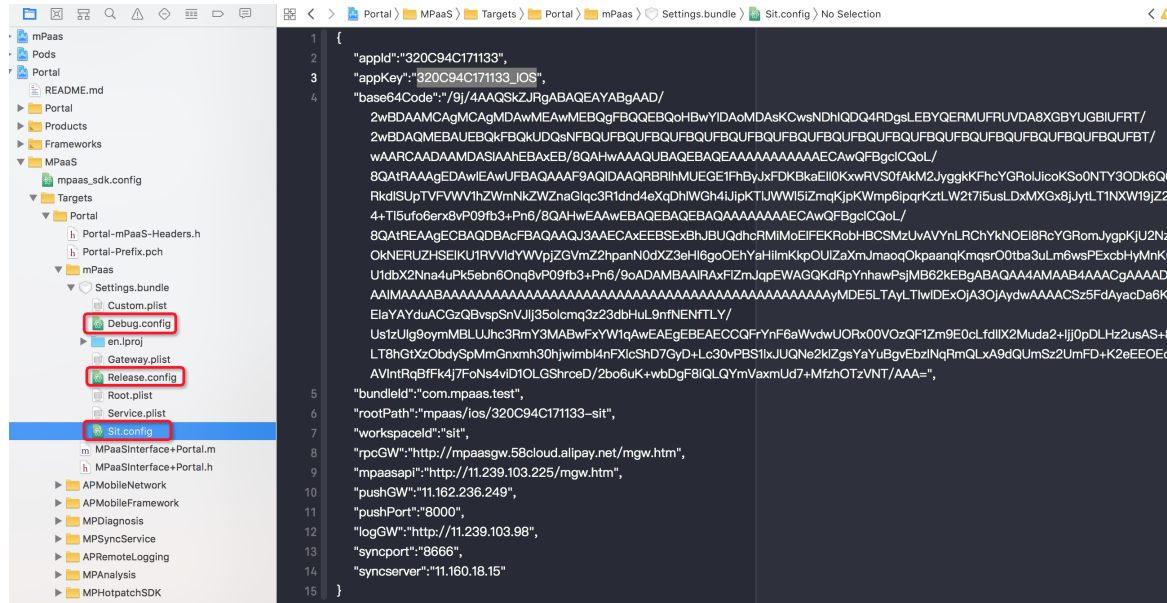
- (BOOL)enableSettingService
{
    return YES;
}

@end
```

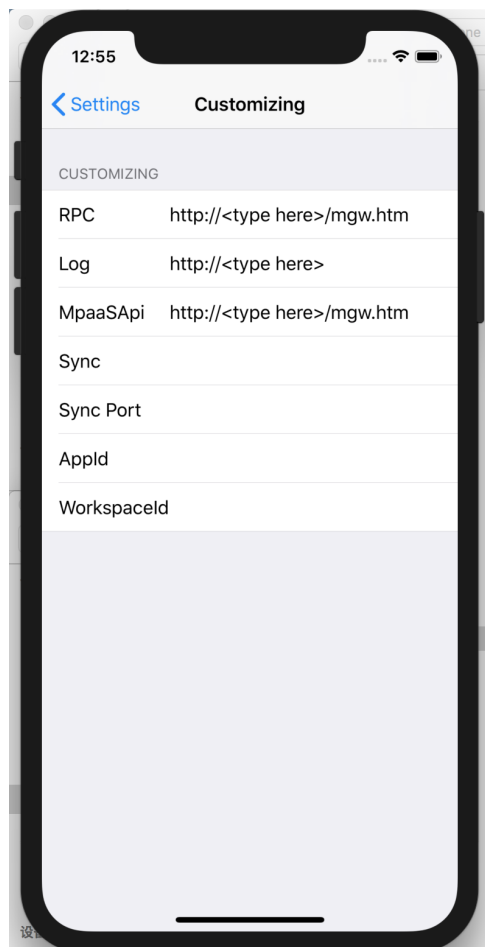
2. Download the workspace configuration file [Settings.bundle.zip](#), and then add it to the project.



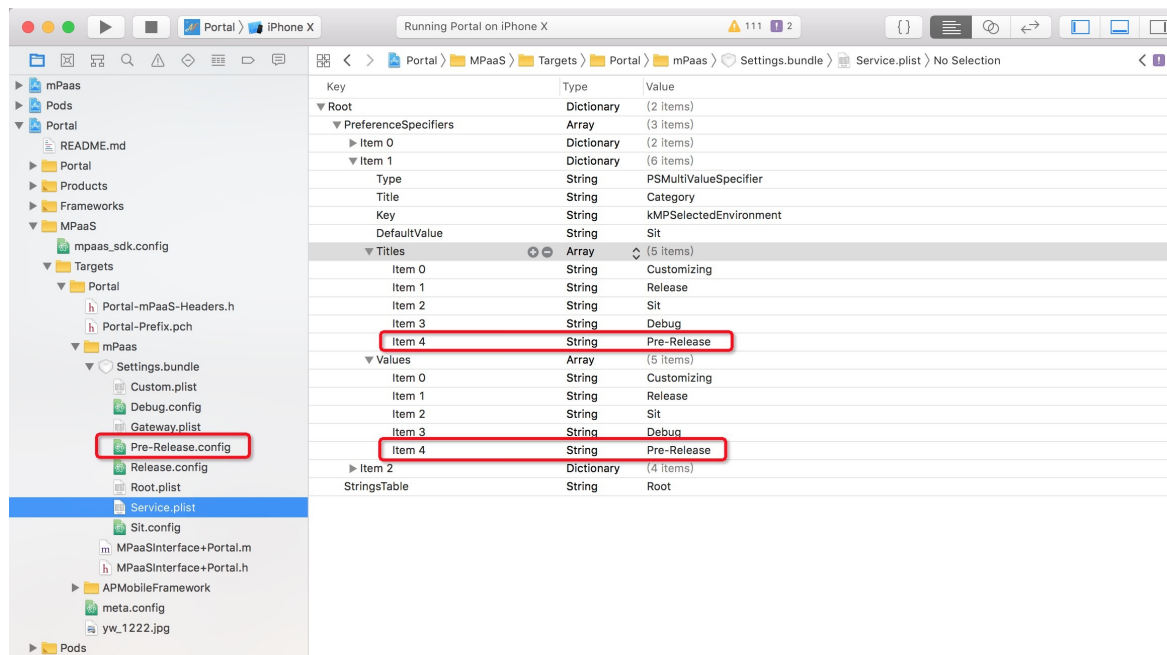
3. Understand the workspace configuration information in `Setting.bundle`.
 - `Setting.bundle` provides four workspaces by default, corresponding to the 4 configuration files: Debug, Sit, Release, and Custom respectively.
 - Among them, Debug, Sit and Release are preset workspaces. You need to download 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`.



- The Custom workspace applies to a scenario where repackaging is not performed on the client and app workspace information is directly configured in mobile phone settings. The configuration path on a mobile phone is Settings > Current app > Settings > Customizing. Enter the corresponding values based on the downloaded configuration information.

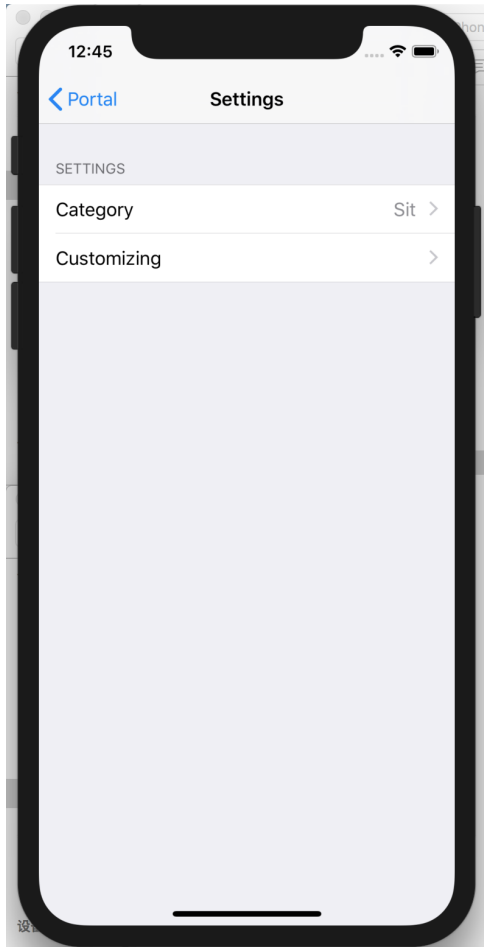


- To preset more workspaces in addition to the default Debug, Sit, Release, and Custom workspaces, add a configuration item in `Setting.bundle/Service.plist` and add the corresponding configuration file. The format is as follows:



Dynamic workspace switching

- After the Settings.bundle workspace configuration file is added, the app configuration information will overwrite the default `meta.config` file in the project, and the workspace selected in `Settings.bundle` prevails. The path for viewing the currently selected workspace is: **Mobile phone Settings > mPaaS app > Settings > Category**. The Sit workspace is selected by default.



- 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.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
```