

Table of Contents

Introduction	0
Guide of Access	1
Installing mPaaS SDK	1.1
Configuring Xcode Project	1.2
Implementing mPaasAppInterface	1.3
Using Setting Service	1.4
Building Complete Demo	1.5
Appendix	1.6
Client framework	2
Overview	2.1
Basic Terminologies	2.2
Quick Start	2.3
Guide to Advanced Features	2.4
FAQs	2.5
Release History	2.6
Appendix	2.7
Client monitoring	3
Overview	3.1
Basic Terminologies	3.2
Quick Start	3.3
Guide to Advanced Features	3.4
FAQs	3.5
Release History	3.6
Appendix	3.7

Basic Communication	4
RPC	4.1
Overview	4.1.1
Basic Terminologies	4.1.2
Quick Start	4.1.3
Guide to Advanced Features	4.1.4
FAQs	4.1.5
Release History	4.1.6
Appendix	4.1.7
APNS	4.2
Generic Service	5
Hotpatch	5.1
Configuration Service	5.2
Badge	5.3
User Feedback	5.4
File Upload and Download	5.5
Checking for Updates	5.6
Tool Component Set	6
Data Center	6.1
Overview	6.1.1
Function Module	6.1.2
APDataCenter	6.1.2.1
Key-Value storage	6.1.2.2
DAO	6.1.2.3
Overview	6.1.2.3.1
Keywords	6.1.2.3.2
Reference	6.1.2.3.3

DAO Proxy	6.1.2.3.4
Transaction	6.1.2.3.5
Function overloading	6.1.2.3.6
Concurrent SELECT operations	6.1.2.3.7
High-level syntax	6.1.2.3.8
LRU	6.1.2.4
APLRUMemoryCache	6.1.2.4.1
APLRUDiskCache	6.1.2.4.2
Custom storage	6.1.2.5
APCustomStorage	6.1.2.5.1
APAsyncFileArrayService	6.1.2.5.2
APObjectArrayService	6.1.2.5.3
FAQs	6.1.3
Audio Recording and Playback	6.2
Sharing Component	6.3
Handling of Chinese Pinyin	6.4
iOS APSecurityUtility	6.5
H5 Container	6.6
Common control	6.7
UI Controls	6.7.1
UI Style Description File	6.7.2
Description File	6.7.2.1
Custom Theme	6.7.2.2
APNavigationBar	6.7.2.3
IconFont	6.7.3
SonicWaveNFC Component	6.8
Multilingual Component	6.9

Code Scan Component	6.10
Developer Tools	7
 Overview	7.1
 Xcode Project Template	7.2
 Xcode File Template	7.3
 Xcode Plug-in	7.4
 Command Line Tools	7.5
 Release History	7.6
Appendix	8

Table of Contents |

@cnName 1 Installing mPaaS SDK @priority 1

1 Installing mPaaS SDK

1.1 Install developer tools

Execute the following command in your Terminal. For details, please refer to [Developer Tools-Overview](#). Please quit Xcode completedly before installing.

```
sh
```

1.2 Install SDK packages

Execute the following command in your Terminal.

```
mpaas sdk update
```

@cnName 2 Configuring Xcode Project @priority 2

2 Configuring Xcode Project

[TOC]

You can configure a project based on mPaas.framework manually, but it is usually not necessary, as you can generate a template project using developer tools.

This chapter can be skipped, but we recommend you read it through for better understanding in mPaaS-based Xcode projects.

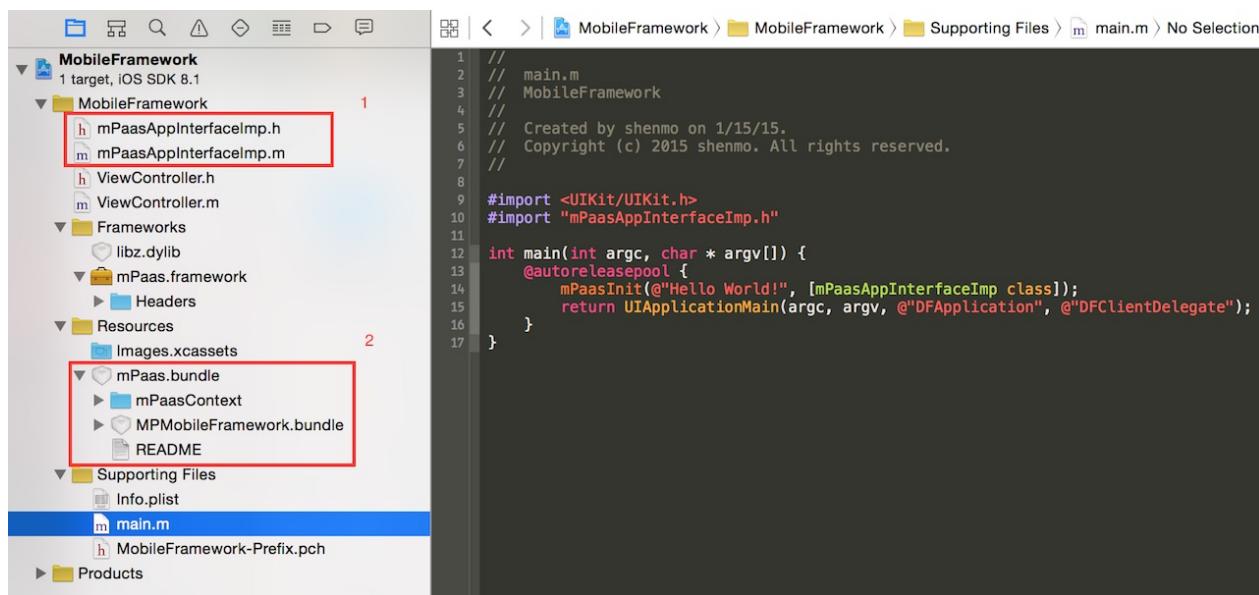
2.1 Adding mPaas.framework and resources to a project

Add the ready mPaas.framework and resource bundles to the Xcode project. You only need to reference the mPaas.h file. It is recommended that you create a public PCH file so that you don't need to reference the header file separately for all the files in the project.

```
#ifdef __OBJC__
#import <UIKit/UIKit.h>
#import <mPaas/mPaas.h>
#endif
```

The directory structure of a project is as follows:

Table of Contents |



2.2 Adding library

After mPaas.framework is added, the project compiling may fail and the following libraries can be added as appropriate:

- | Name |
|-----------------------|
| libstdc++.6.dylib |
| libc++.dylib |
| MediaPlayer.framework |
| libxml2.dylib |
| libz.dylib |

The header file search paths that may be added:

`${SDK_ROOT}/usr/include/libxml2`

The screenshot shows the Xcode settings for Header Search Paths.

Header Search Paths Configuration:

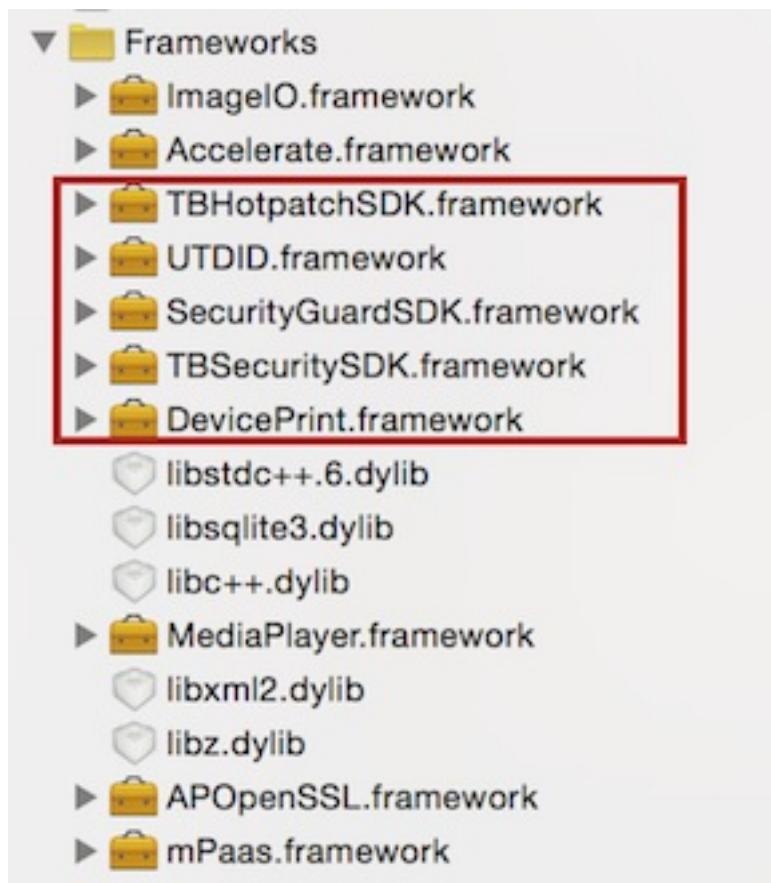
- Always Search User Paths
- Framework Search Paths
- Header Search Paths** (selected)
 - Library Search Paths
 - Rez Search Paths
 - Sub-Directories to Exclude in Recursive Searches
 - Sub-Directories to Include in Recursive Searches
 - User Header Search Paths

Header Search Paths Value:

```
$(inherited)
/Applications/Xcode.app/Contents/Developer/Toolchains
${SDK_ROOT}/usr/include/libxml2
```

2.3 Adding third-party dependency

Some components contained in mPaas.framework may depend on some third-party frameworks and such third-party frameworks should also be added to the project.



Add -ObjC to the link option of the application

Other Librarian Flags ► Other Linker Flags ▼ Path to Link Map File <ul style="list-style-type: none"> Debug Release 	-ObjC <small><Multiple values></small> <input type="text" value="-ObjC"/>
---	--

2.4 Accessing mPaaS application framework

Modify the application initialization method as follows and delete the main storyboard in the plist file of the application. The modification will not be necessary if the mPaaS application framework is not used, and only mPaaS

public components and services are used.

```
UIApplicationMain(argc, argv, @"DFApplication", @"DFClientDelegate");
```

The application can custom a class inherited from the DFClientDelegate, and use the self-defined class in the last argument of UIApplicationMain function, which is not recommended though. The appDelegateEvent in mPaaSAppInterface class is enough for use. Please see [Environment Variable](#)

```
- (id)appDelegateEvent:(mPaaSAppEventType)event arguments:(NSDictionary*)arguments;
```

2.5 Initializing mPaaS by calling mPaaSInit method

This is the initialization method of mPaaS and is recommended to be called prior to the UIApplicationMain method of the main function. If the mPaaS application framework is not applied, the application:didFinishLaunching method of AppDelegate can be used.

```
/**<br/>
 * <#Description#><br/>
 * @param appKey The AppKey of the application<br/>
 * @param appInterfaceClass The interface class of the application environment<br/>
 * @return Successful or not<br/>
 */<br/>
BOOL mPaaSInit(NSString* appKey, Class appInterfaceClass);
```

@cnName 3 Implementing mPaasApplInterface @priority 3

3 Implementing mPaasApplInterface

[TOC]

3.1 Overview

The '@protocol mPaasApplInterface' is implemented by developers of the access party and passes the interface class to mPaaS in the mPaasInit method, for example, the 'mPaasApplInterfaceImp' class below.

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

int main(int argc, char * argv[]) {
    @autoreleasepool {
        mPaasInit(@"productId", [mPaasApplInterfaceImp class]);
        return UIApplicationMain(argc, argv, @"DFApplication", @"DFC1");
    }
}
```

mPaaS acquires the data, usually some configurations, provided by the access application with interfaces of this class.

3.2 Description of mPaasApplInterface

```
//
//  mPaasApplInterface.h
//  mPaasContext
//
//  Created by shenmo on 1/14/15.
```

```
// Copyright (c) 2015 Alipay. All rights reserved.  
//  
  
#import <Foundation/Foundation.h>  
  
// When the mPaaS client framework is used, the inner class DFClientDe  
// DFClientDelegate will notify the access application of events throu  
// Below are some defined event names and their respective triggering  
typedef NS_ENUM (NSInteger, mPaasAppEventType)  
{  
    mPaasAppEventBeforeDidFinishLaunching, // Called at the entry of (i  
    mPaasAppEventAfterDidFinishLaunching, // Called at the end of @sele  
    mPaasAppEventBeforeStartLoader, // Called before initiating the lo  
    mPaasAppEventDidReceiveRemoteNotification, // Called at the entry  
    mPaasAppEventDidReceiveRemoteNotificationFetchCompletion, // @selec  
    mPaasAppEventDidFailToRegisterForRemoteNotifications, // @selecto  
    mPaasAppEventDidReceiveLocalNotification, // Called at the entry o  
    mPaasAppEventQueryOpenURL, // Called by @selector(application:ope  
    mPaasAppEventWillResignActive, // Called by @selector(application:  
    mPaasAppEventDidEnterBackground, // Called by @selector(application:  
    mPaasAppEventWillEnterForeground, // Called by @selector(application:  
    mPaasAppEventDidBecomeActive, // Called by @selector(applicationD  
    mPaasAppEventWillTerminate, // Called by @selector(applicationWil  
    mPaasAppEventHandleWatch, // Called by @selector(application:hand  
};  
  
/**  
 * This will be implemented by the application accessed to the mPaaS  
 */  
@protocol mPaasAppInterface <NSObject>  
  
@optional  
  
/**  
 * The short name of the application. For example, the wallet is named  
 */  
- (NSString*)appBriefName;
```

```
/**  
 *  The BundleIdentifier of the application. Not used for the moment  
 */  
- (NSString*)appBundleIdentifier;  
  
/**  
 *  The BundleIdentifier of the RC version of the application. It must  
 *  The Alipay wallet has an RC version as the stable output during th  
 *  If the application connected to mPaas also has an RC version avail  
 */  
- (NSString*)appRCBundleIdentifier;  
  
#pragma mark Setting Service  
  
/**  
 * Whether to use mPaas Setting Service.  
 * If yes, the configurations should be written in GatewayConfig.plist.  
 * For non-released versions, a switch for selecting the environment  
 * The method will return whether to use Setting Service. If it retur  
 *  
 * The access party can modify configurations by defining the Setting  
 * With the Setting Service enabled, if mPaas fails to retrieve the l  
 * The default structure of GatewayConfig.plist is as follows:  
 * Root  
 *   |- Debug  
 *   |- Pre-release  
 *   |- Release  
 *     |- mPaasPushAppId          The application ID used for ac  
 *     |- mPaasLogServerGateway    The URL of the log server (such  
 *     |- mPaasLogProductId       The log application ID. It is  
 *     |- mPaasRpcGateway         The RPC gateway URL (such as '  
 *     |- mPaasRpcETagURL        The RPC ETag URL (such as "htt  
 */  
- (BOOL)appUseSettingService;  
  
#pragma mark Hotpatch  
  
/**
```

```

    * The key name configured in Security Guard for encrypting the Hotpatch
    *
    * @return The key name configured in Security Guard
    */
- (NSString*)appHotpatchScriptEncryptionKey;

#pragma mark Log

/**
 * URL of the remote log server. When the Setting Service is not used,
 */
- (NSString*)appRemoteLogServerURL;

/**
 * Name of the log service application. Some arguments may be required.
 */
- (NSString*)appRemoteLogProductId;

/**
 * The set of log types to be uploaded by default as defined by the APLogTypeBehavior.
 * If it is not defined, the application will upload three types of logs.
 *
 * @return @{@"APLogTypeBehavior"}, @{@"APLogTypeCrash}]
 */
- (NSArray*)appDefaultUploadLogTypes;

/**
 * The log service supports the client report-active feature. After the application
 * When the application is switched from the background to the foreground
 * This will not affect the cold start. The client will report active when
 *
 * @return Number of seconds
 */
- (NSInteger)appReportActiveMinIntervalSeconds;

#pragma mark Framework callback

/**

```

Table of Contents |

```
* When the client framework is used, DFClientDelegate will take over
* In usual cases, you only need to implement the callback function :
*/
- (id)appDelegateEvent:(mPaasEventType)event arguments:(NSDictionary*)arguments;

#pragma mark Login status

/***
 * It returns the userId of the current user logged in. If the user is not
 */
- (NSString*)currentUserId;

/***
 * It returns the current sessionId. If the user is not logged in, it returns
 */
- (NSString*)currentSessionId;

/***
 * Names of notifications for successful logins and logouts. Persistent
 */
- (NSString*)loginSuccessNotificationName;
- (NSString*)logoutNotificationName;

#pragma mark Scheme

/***
 * For a third-party navigation scheme whose RC version is different
 * If this field is absent in the configuration file, the framework will
 */
- (NSString*)appSchemePatternName;

/***
 * It returns the scheme handler class to be added. The returned handler
 */
- (NSArray*)appSchemeHandlerClasses;

#pragma mark ShareKit
```

Table of Contents |

```
/**  
 * It configures arguments of sharing channels, mainly including the  
 *  
 * @return The configuration dictionary of the desired sharing channel.  
 * The dictionary format is: @{@"laiwang" : @{@"key" : @"your_key", (br/> * @"weixin" : {}, @"weibo" : {}, @"qq" : {}};  
 * A decimal APPID can be passed as the QQ key value.  
 */  
- (NSDictionary*)appShareKitConfig;  
  
#pragma mark PushService  
  
/**  
 * The default URL of the Push provider server, used when the Push se  
 */  
- (NSString*)appPushProviderServerURL;  
  
/**  
 * The AppId of the Push service. The application ID used for access  
 */  
- (NSString*)appPushAppId;  
  
/**  
 * The application receives the remote push notification. Not used -  
 */  
- (void)appDidReceiveRemoteNotification:(NSDictionary*)info;  
  
#pragma mark Network configurations  
  
/**  
 * The server address of the RPC service. If the configuration service  
 */  
- (NSString*)appRPCGatewayURL;  
  
/**  
 * The ETag server address of the RPC service. If the configuration :  
 */  
- (NSString*)appRPCETagURL;
```

```
/**  
 *  RPC service will adopt the timeouts defined by the application if  
 *  
 *  @return NSTimeInterval, in seconds.  
 */  
- (NSTimeInterval)appRPCTimeoutInterval;  
  
/**  
 *  The default RPC interceptor container class. If this method is avai  
 *  An application may have multiple RPC interceptors. You need to add  
 */  
- (NSString*)appRPCCommonInterceptorClassName;  
  
/**  
 *  It defines whether to activate the Sync service on the application.  
 */  
- (BOOL)appSyncServiceActive;  
  
/**  
 *  The AppName, URL and port of the Sync service. It is only valid wh  
 */  
- (NSString*)appSyncServiceAppName;  
- (NSString*)appSyncServiceConnectionURL;  
- (NSInteger)appSyncServiceConnectionPort;  
  
/**  
 *  It provides mPaaS with the domain name for direct access with the  
 */  
- (NSArray*)appHttpDNSHosts;  
  
/**  
 *  Mtop configurations. If this method is not implemented and the Mto  
 *  Mtop service is only used in the scenario of Alipay account login  
 *  
 *  @return The Mtop environment. There are three values: 0: online; 1:  
 */  
- (NSInteger)appMtopEnvironment;
```

```
#pragma mark Data center

/***
 *  The Data Center feature supports data encryption. The encryption :
 *
 *  Implement this method, and a 32-byte encryption key will be returned.
 *  If this method is not implemented and the Data Center function is
 *
 *  @return The 32-byte key in NSData.
 */
- (NSData*)appDataCenterDefaultCryptKey;

@end
```

@cnName 4 Using Setting Service @priority 4

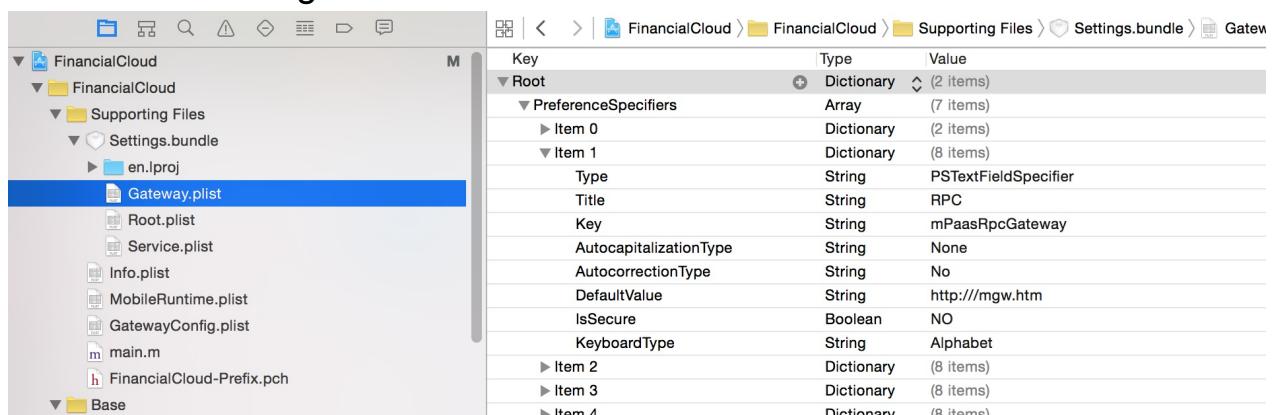
4 Using Setting Service

[TOC]

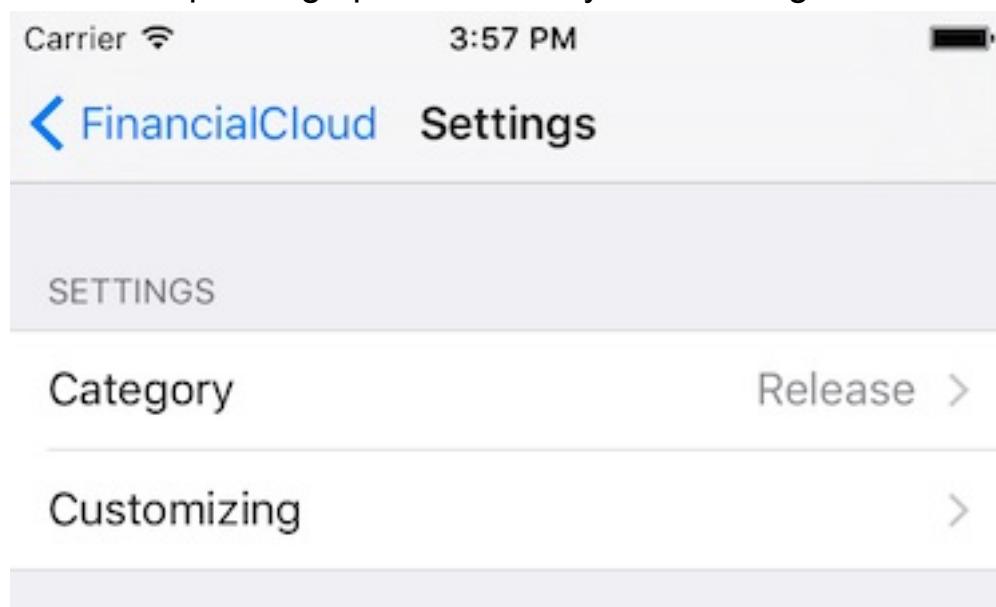
4.1 Overview

Developers can add the settings function for their applications in system settings. This function is usually used to configure the server address in the development phrase for the purpose of switching environment without changing the package.

After a template project is generated using developer tools, the project will contain the Settings.bundle.



The corresponding options in the system Settings are as follows:



In 'Category', the 'Customizing', 'Release', 'Pre-release', and 'Debug' options are provided. The specific environment address can be configured in 'Customizing'. The configurations will take effect after the application is restarted.

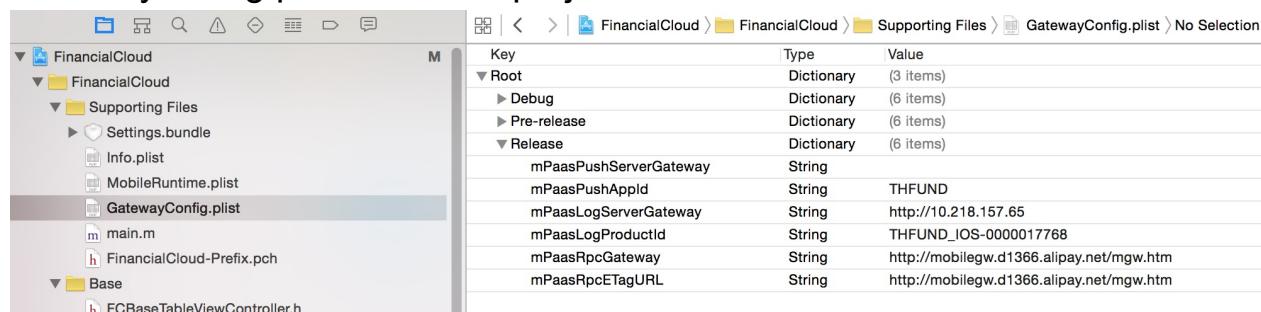
4.2 Configuring Setting Service

4.2.1 Enabling Setting Service

In the implementation class of mPaasAppInterface, when the appUseSettingService method returns YES, it indicates that the Setting Service has been enabled.

```
- (BOOL)appUseSettingService
{
    return YES;
}
```

The mPaaS will use environment variables configured in the `GatewayConfig.plist` file in the project.



The developer only needs to configure the desired URL in the corresponding environment in `GatewayConfig.plist`. The key value in the dictionary cannot be modified.

4.2.2 Preset settings

Name	Format Style	Me...
mPaasPushServerGateway	N/A	Not us... the mo... You c... ignore... remov...
mPaasPushAppId	THFUND	Appli... of the ... no pla... inform... which ... same ... in An... is diffe... from t... AppKe... mPaa...
mPaasLogServerGateway	http://mdap.alipay.com	URL c... log se... which ... serv... addre... the m... point,... report... crash...

		user diagnos- tisyste
mPaasLogProductId	THFUND_IOS-0000017768	ID of the service application is usually composed by the Application (with platform) and the Workspac
mPaasRpcGateway	http://mobilegw.d1366.alipay.net/mgw.htm	URL configuration of the RESTful service. When the application is available online, HTTP should be used, for development and debugging, HTTPS is used.
mPaasRpcETagURL	Ditto	Ditto

4.2.3 Not using Setting Service

When the Setting Service is not used, i.e., when the appUseSettingService method returns NO, mPaaS will request the environment address from the accessed application using callback methods of mPaasApplInterface. The specific methods involved are as follows:

Table of Contents |

```
/**  
 *  RPC server URL  
 */  
- (NSString*)appRPCGatewayURL;  
  
/**  
 *  ETag server URL of the RPC service  
 */  
- (NSString*)appRPCETagURL;  
  
/**  
 *  AppId of the Push service  
 */  
- (NSString*)appPushAppId;  
  
/**  
 *  URL of the remote log server  
 */  
- (NSString*)appRemoteLogServerURL;  
  
/**  
 *  Name of the log service application. Some arguments may be required.  
 */  
- (NSString*)appRemoteLogProductId;
```

@cnName 5 Building Complete Demo @priority 5

5 Building Complete Demo

[TOC]

This guide will instruct how to create an mPaaS-based application and how to configure the basic functions of client framework, RPC and logs. In Section 3, we will create a demo application from scratch, including calling RPC interfaces, accessing Apple Push service and reporting crashes. For more information about configuring and using modules, see the detailed documents of respective modules.

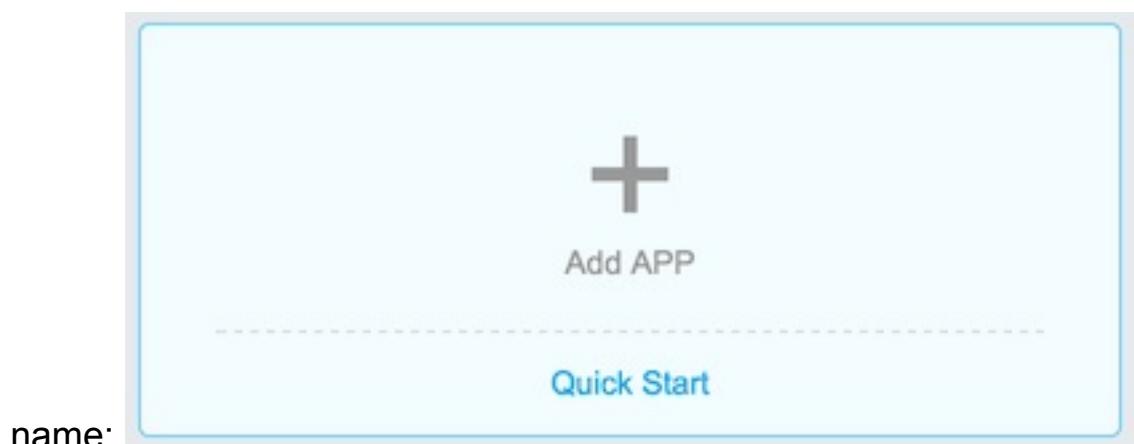
5.1 Using mPaaS website

5.1.1 Creating your first application

After logging into the primary site with a developer account, the developer can see all the applications under the account.



Click "Add Application" to create a new application, and input the application



name:

Back

Create Mobile APP

1 Basic Info

2 Create Version

* CN Name:

* AppID

APP Icon:



Next

Select the mPaaS modules the application is dependent on (This step can be skipped. We recommend using mPaaS Xcode plugin to maintain modules.):

Table of Contents |

Back

Create Mobile APP

Basic Info 2 Create Version

* Select Platform Type: iOS Android

* Select Extension Component:

<input type="checkbox"/>	Component Name	Component Description
<input type="checkbox"/>	MPAlipayCashierKit	Alipay cashier kit for use in mainland of China.
<input type="checkbox"/>	MPAnimationDirector	iOS animation script using CoreAnimation. Supporting time-line, key-frame, function invocation.
<input type="checkbox"/>	MPAudioKit	Recording and replaying of AMR format audio stream.
<input type="checkbox"/>	MPBadgeService	Badge notification component for client. Supporting many styles of badge notification such as red-dot, number, New. Automatically managing tree-structure relationship.
<input type="checkbox"/>	MPCommonUI	User interface components based on Alipay client.

* Select Code Repo: New Self-owned

SVN

Support SVN only now

Complete

After the setup is complete, the application you just created can be viewed on the Management Console. Click on the application and the application details will be displayed:

The screenshot shows the Alipay Management Console interface. On the left, there's a sidebar with options like 'Mobile', 'APP Management' (which is selected), 'Build & Release', 'Service Management', 'Statistics Analysis', and 'Message Management'. In the main area, under 'APP Management / Settings', it shows an app named 'ecool'. The app's details include its creator ('alipayAdmin'), creation time ('2015-11-02 16:10:32'), and identifiers ('AppID: ECOOL', 'WorkspaceID: 0000000001'). Below this, there's a 'Basic Information' section with tabs for 'iOS' and 'Android'. It lists 'AppSecret' (with a copy link), 'Last Package' (status: 'Packaging succeeded Download'), and 'Code Repository' (link: 'Create Code Repository'). At the bottom, there's a 'Component Info' section with tabs for 'Mobile Gateway Service', 'Mobile Push Service', and 'Mobile Analysis Service'. It shows the 'Gateway Address' as '10.218.157.79' with a 'Copy' link.

5.1.2 Generating mPaas.framework

We recommend using mPaaS Xcode plugin to maintain modules. You do not need to re-compile mPaaS.framework anymore.

The developer can select mPaaS modules on the primary site to generate their own mPaaS.framework. Click "iOS Details" and select "Repackaging" on the pull-down page. The optional modules will be displayed. Check the desired modules, and click "OK".

Components	Component Description
<input type="checkbox"/> MPAlipayCashierKit	Alipay cashier kit for use in mainland of China.
<input type="checkbox"/> MPAnimationDirector	iOS animation script using CoreAnimation. Supporting time-line, key-frame, function invocation.
<input type="checkbox"/> MPAudioKit	Recording and replaying of AMR format audio stream.
<input checked="" type="checkbox"/> MPBadgeService	Badge notification component for client. Supporting many styles of badge notification such as nesting tree-structure relationship.

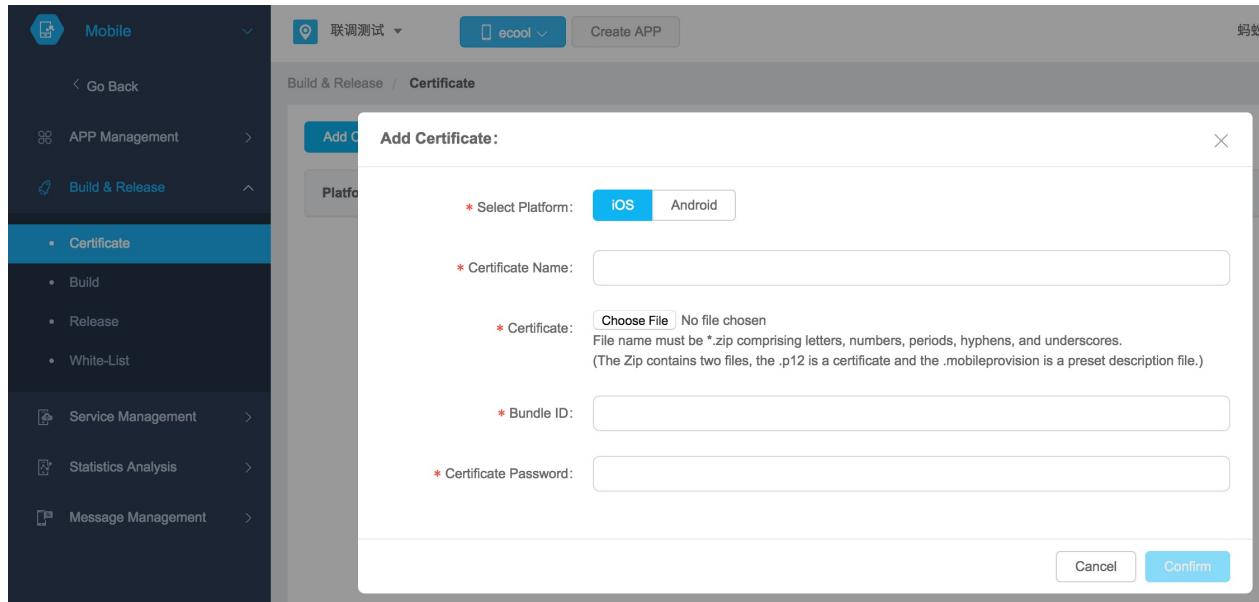
After packaging is complete, the developer can download the package, extract the mPaaS.framework from the package and use it to replace the .framework file in the project:

Creation Time	Creator	Select Component	Operation
2015-11-30 14:45:32	alipayAdmin@alipay.net	MPAlipayCashierKit MPBadgeService	Download

5.1.3 Building applications

The developer can select to use the online compiling feature provided by the primary site. First, the developer needs to upload the iOS packaging certificate, which is maintained by the developer, of the application. The certificate can be added or deleted.

The developer can select to use the online compiling feature provided by the primary site. First, the developer needs to upload the iOS packaging certificate, which is maintained by the developer, of the application. The certificate can be added or deleted.



Place the certificate (.p12) and the provisioning profile (.mobileprovision) to the same directory and generate a zip package. In APP IDS field, fill in the corresponding Bundle Identifier for using the certificate. Packaging scripts will modify the Bundle ID of the application to this value. In Certificate Password field, fill in the certificate key which will be used by the packaging tool for installing certificates automatically.

After the build is complete, download the .ipa package of the application, or scan the QR code online to install the application.

5.1.4 Viewing online status of application

The screenshot shows the mobile statistics analysis interface. On the left, there's a sidebar with navigation items like 'Mobile', 'APP Management', 'Build & Release', 'Service Management', 'Statistics Analysis' (which is selected), and 'Crash'. The main area has tabs for 'STATISTICS', 'TREND', and 'ANALYSIS'. It displays a table of crash data with columns for 'Time', 'Crash', and 'Report Active'. The data shows 0 crashes for each minute from 14:48 to 14:42.

Time	Crash	Report Active
14:48	0	0
14:47	0	0
14:46	0	0
14:45	0	0
14:44	0	0
14:43	0	0
14:42	0	0

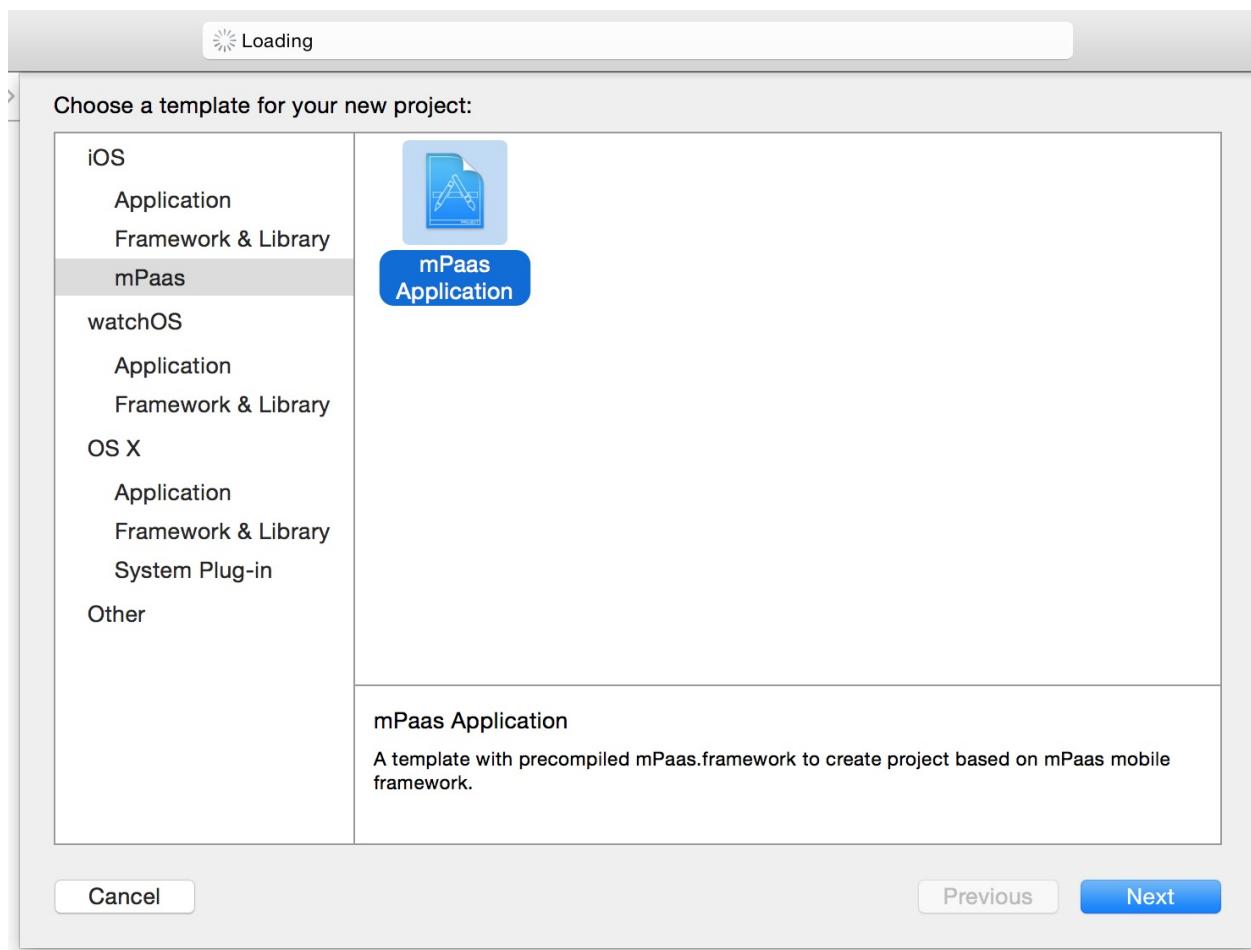
5.2 Configuring mPaaS-based project

5.2.1 Creating application using command line tools or Xcode templates

By installing a command line tool or Xcode templates, the developer can quickly create an mPaaS-based project, saving the effort for the complicated project configurations. (For details, see the '[Quick Start->Configuring Xcode Project](#)' chapter.)

```
sh <(curl -s http://code.taobao.org/svn/mpaaskit/trunk/install.sh)
```

After the installation is complete, the developer can use the command 'mpaas createapp XXXX' to create a new application, or they can use the Xcode templates alternatively.



5.2.2 Adding or removing module

The project created through the command line tool or Xcode templates is a simplest one, and it is integrated with only the most basic mPaaS modules by default. To add or remove a module in future development, the developer can refer to the [1.2 Generating mPaas.framework](#) chapter to generate the mPaas.framework on the primary site for replacing the existing .framework file in his/her own project.

Attention: After a module is added, there may be a newly added system library or third-party dependency, or there may be a newly added *.bundle resource directory under the new mPaas.framework. They should also be added to the project.

In the future, we will offer easier-to-use Xcode plug-ins to complete this job automatically.

5.2.3 Initializing mPaaS

The mPaasInit method will be called to initialize mPaaS. (The project generated by templates already incorporates this method.):

```
int main(int argc, char * argv[]) {
    @autoreleasepool {
        mPaasInit(@"TEST_IOS", [mPaasAppInterfaceImp class]);
        return UIApplicationMain(argc, argv, @"DFApplication", @"DFC1");
    }
}
```

The first argument is the 'mPaas appKey' of the application. It can be seen in the "iOS Details" on the primary site.

test 创建于: 2015-08-05 11:46:27

基础信息

组件

AppKey: TEST_IOS

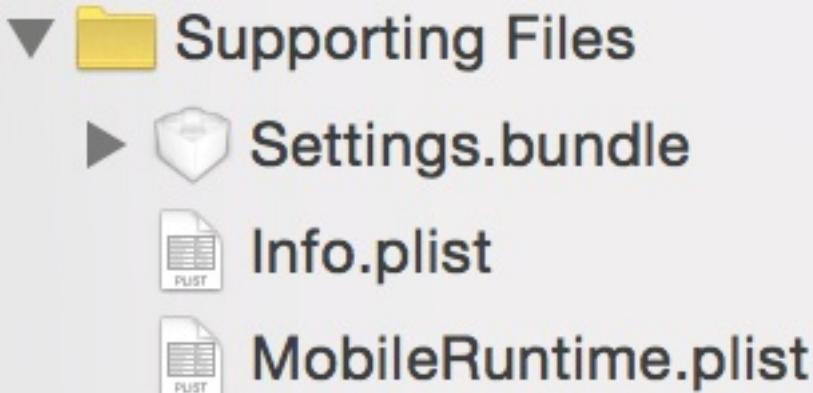
iOS 详情

This AppKey and its secret key should be configured in Security Guard for signing RPC requests. (If the access party does not require the RPC feature, this AppKey can be skipped.)

The second argument is the class name of the '@protocol mPaasAppInterface' implemented by the application. Some arguments of mPaaS need to be provided by the application. The mPaaS module will call back methods of this interface. For details, see '[Quick Start->Environment Variable and Configuration Service](#)' chapters.

5.2.4 Configuring applications and services of framework

The 'micro applications' and 'services' managed by the mPaaS client framework are configured in the MobileRuntime.plist file, as shown in the figure:



For specific how-tos, see '[Client Framework->Micro Application->Modifying Application Configuration](#)' and '[Client Framework->Service->Modifying Service Configuration](#)' chapters.

5.2.5 Configuring log module and crash reporting feature

The log module requires two arguments:

'Log Server Gateway': Log server URL of the application.

'Log Product Id': Log service product ID of the application.

With the two callback methods under the '@protocol mPaaSAppInterface', the two arguments can be returned to the mPaaS log module:

- (NSString*)appRemoteLogServerURL;
- (NSString*)appRemoteLogProductId;

To enable crash reporting, the developer only needs to call the 'enable_crash_reporter_service()' method in the main function. When the application uses Xcode for debugging on the simulator or a real machine, no crash logs will be reported.

If your application uses the Setting Service provided by mPaaS, see '[Quick Start->Environment Variable and Configuration Service](#)' chapters to configure the log module.

5.2.6 Configuring RPC

With the callback method under the '@protocol mPaasApplInterface', the RPC gateway URL is returned to the mPaaS RPC module:

```
- (NSString*)appRPCGatewayURL;
```

RPC interceptor is a way to achieve control on RPC behaviors on the client. The application should use a Common Interceptor as the container class of all the interceptors. This Common Interceptor also implements '@protocol DTRpcInterceptor'. All the self-defined interceptors should be added to the Common Interceptor. ['Common Interceptor template code'](#)

With the callback method under the '@protocol mPaasApplInterface', the Common Interceptor class name is returned to the mPaaS RPC module:

```
- (NSString*)appRPCCommonInterceptorClassName
```

See the '[Basic Communication->RPC](#)' chapter for more information.

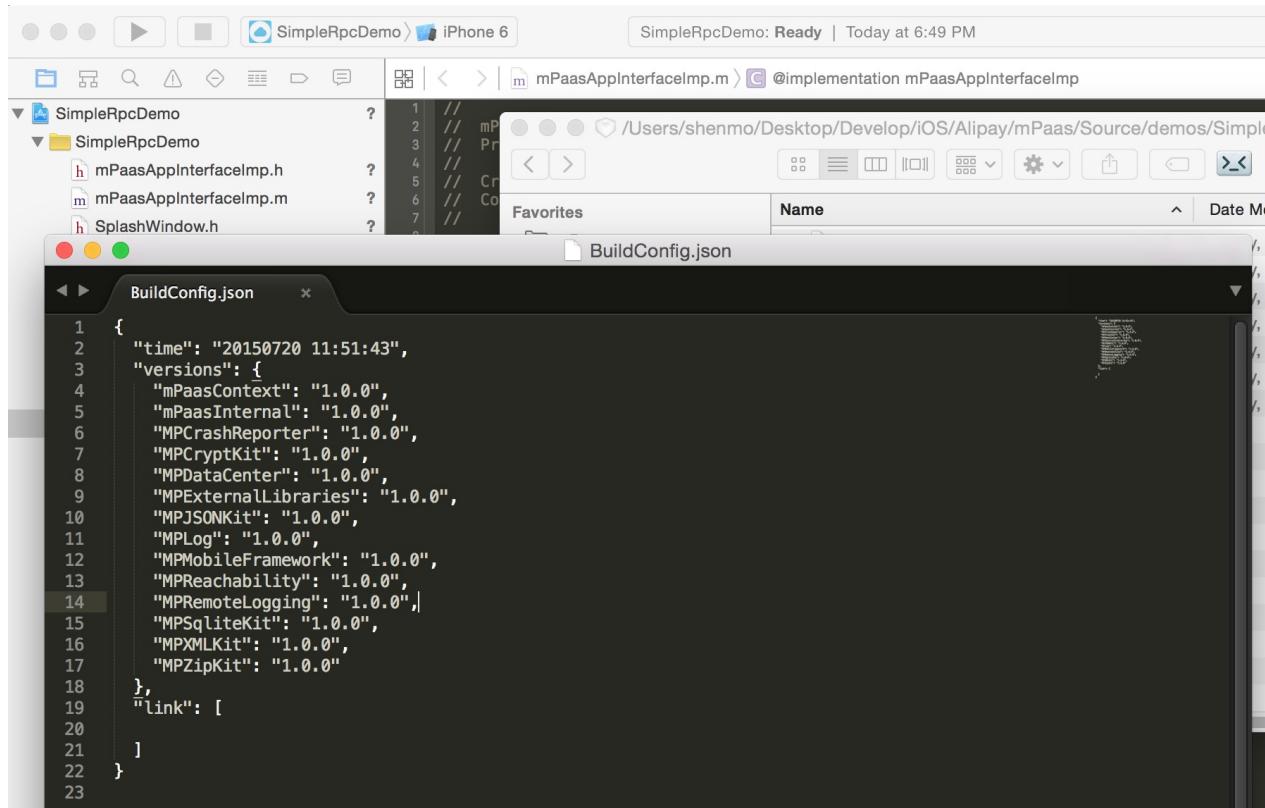
If your application uses the Setting Service provided by mPaaS, see the '[Quick Start->Environment Variable](#)' chapter to configure the RPC module.

5.3 Building demo application

5.3.1 Creating project

With the introduction above, now we will start to build a demo application with login, RPC request sending, APNS notification receiving, and crash reporting features.

First, we will create a project with the command line tool. Let's name the project SimpleDemoRpc, and create the application on the primary site.



The demo project will integrate the RPC and the login window component of CommonUI, but no RPC or CommonUI modules can be found in the 'BuildConfig.json' file in mPaas.framework of the demo project. So we need to go to the primary site, check MPHttClient and MPCommonUI to repackage an mPaas.framework, and use it to replace the current mPaas.framework in the demo project.

'Note: after the framework file is replaced, you also need to add the APCommonUI.bundle into the project. '

Run the project, but many symbols are not found. This is because after the RPC module is introduced, we need to add some third-party dependencies.

▼ ⓘ Link /Users/shenmo/Library/Developer/Xcode/DerivedData/SimpleRpcDemo-dxchv
① "_OBJC_CLASS_\$_OpenSecurityGuardManager", referenced from:
Objc-class-ref in mPaaS
① "_OBJC_CLASS_\$_OpenSecurityGuardParamContext", referenced from:
Objc-class-ref in mPaaS
① "_OPEN_ENUM_SIGN_COMMON_MD5", referenced from:
-[DTRpcOperation signRequestBody] in mPaaS
① "_OPEN_KEY_SIGN_INPUT", referenced from:

Add 'SecurityGuardSDK' to the project. The compiling succeeded.

'Download Third-party Library'

Submit the codes to SVN and the repository URL can be found in the Basic Information area on the website:



5.3.2 Configuring mPaaS

Our demo project uses Setting Service for managing the environment. So we should make the 'appUseSettingService' method return YES:

```
- (BOOL)appUseSettingService
{
    return YES;
}
```

The testing gateway URL: <http://10.218.157.194/mgw.htm>

the testing log server URL: <http://10.218.157.65>

log application ID: SIMPLEDEMORPC_IOS-0000017768

and Push application ID: SIMPLEDEMORPC

are all configured into 'GatewayConfig.plist'. (The values are used by the demo project, and access parties will use their own configurations), as shown in the figure:

SimpleRpcDemo > SimpleRpcDemo > Supporting Files > GatewayConfig.plist > No Selection		
Key	Type	Value
Root	Dictionary	(3 items)
▶ Debug	Dictionary	(5 items)
▶ Pre-release	Dictionary	(5 items)
▼ Release	Dictionary	(5 items)
mPaaSPushAppId	String	SIMPLEDEMORPC
mPaaSLogServerGateway	String	http://10.218.157.65
mPaaSLogProductId	String	SIMPLEDEMORPC_IOS-0000017768
mPaaSRpcGateway	String	http://10.218.157.194/mgw.htm
mPaaSRpcETagURL	String	http://10.218.157.194/mgw.htm

5.3.3 Adding login and APNS Token accessing features

The login feature is abstracted into the micro application 'LoginAppDelegate' and login service 'LoginService', and added to 'MobileRuntime.plist':

SimpleRpcDemo > SimpleRpcDemo > Supporting Files > MobileRuntime.plist > No Selection		
Key	Type	Value
Root	Dictionary	(3 items)
Launcher	String	20000001
▼ Services	Array	(2 items)
▶ Item 0	Dictionary	(3 items)
▼ Item 1	Dictionary	(3 items)
class	String	LoginServiceImpl
lazyLoading	Boolean	NO
name	String	LoginService
▼ Applications	Array	(3 items)
▼ Item 0	Dictionary	(3 items)
delegate	String	LoginAppDelegate
description	String	Login
name	String	20000003

'LoginService' is in charge of obtaining the APNS Token of the application, open account and password login interface (this interface is exclusive to the demo application), as well as recording the sessionId of successful logins:

```
@interface LoginServiceImpl ()
{
    NSString* _sessionId;
    NSString* _pushToken;
```

```
}

@end

@implementation LoginServiceImpl

@synthesize sessionId = _sessionId;
@synthesize pushToken = _pushToken;

- (void)start
{
    NSLog(@"LoginServiceImpl started");

    // Push notification registration
#ifdef __IPHONE_8_0      //Push notification registration of iOS8
    if ([[UIDevice currentDevice] systemVersion] floatValue] >= 8.0f)
        UIUserNotificationSettings *settings = [UIUserNotificationSettings
            [[UIApplication sharedApplication] registerUserNotificationSettings:settings
        }
        else
#endif
    {
        [[UIApplication sharedApplication] registerForRemoteNotifications];
    }

    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(applicationDidRegisterForRemoteNotifications:)
                                               name:UIApplicationDidRegisterForRemoteNotificationsNotification
                                         object:nil];
}

- (void)applicationDidRegisterForRemoteNotifications:(NSNotification *)notification
{
    NSString* token = [[notification.object description] stringByTrimmingCharactersInSet:[NSCharacterSet whitespaceAndNewlineCharacterSet]];
    _pushToken = [token stringByReplacingOccurrencesOfString:@" " withString:@""];
    NSLog(@"%@", token);
}

- (NSString*)loginWithUserName:(NSString*)userName password:(NSString*)password
{
}
```

```

{
    id<MPSettingService> settings = [mPaaS() serviceWithName:@"Setting"];
    NSString* appCode = settings.settings[@"mPaaSPushAppId"];
    appCode = appCode ?: @"";
}

MPAASGWLoginDemoServiceFacade* facade = [[MPAASGWLoginDemoServiceFacade alloc] init];
_sessionId = [facade login:@{@"userId":userName, @"password":password} completion:^(NSError * _Nullable error) {
    NSLog(@"sessionId:%@", _sessionId);
    return _sessionId;
}];

@end

```

When the login service is started, 'LoginService' requests the APNS Token from the system, and monitors the framework notification of 'UIApplicationDidRegisterForRemoteNotifications'. The obtained APNS Token will be logged into `_pushToken`.

The 'loginWithUserName:password:' interface receives the account and password, and invokes RPC login requests. The 'appCode' indicates the Push service ID which is available in the Setting Service of mPaaS. The 'osType' is 2, indicating that it is an iOS client. Upload "`_pushToken`" and bind the token to the user name in the background.

Add codes in the login window to call the LoginService interface. After successful login, synchronize the user's login status to the log service and Data Center. For details, see [User-mode Processing](#).

```
__block BOOL succeed = YES;
__weak DMLoginBoxViewController* weakSelf = self;
NSString* userName = _loginBox.usernameField.text;
NSString* password = _loginBox.passwordField.text;
_loginCaller = [self callAsyncBlock:^{
    @try
    {
        id<LoginService> service = [DTContextGet() findServiceByName:([
            service loginWithUserName:userName password:password];
        ]
        @catch (DTRpcException *exception)
        {
            succeed = NO;
        }
    } completion:^{
        DMLoginBoxViewController* strongSelf = weakSelf;
        strongSelf->_loginCaller = nil;
        if (succeed)
        {
            [[[APDataCenter defaultCenter] setCurrentUserId:userName];
             [[APMonitorPointManager sharedInstance] setAuthenticated:YES];
             [[APMonitorPointManager sharedInstance] setUserId:userName];
            [[[APRemoteLogger writeLogWithActionId:KActionID_Event extParam:
                [[[NSNotificationCenter defaultCenter] postNotificationName:LOG_IN_SUCCESS
                [[DTContextGet() findApplicationByName:@"20000003"] exitAnimation];
            }
            else
            {
                [strongSelf showAlert:[NSString stringWithFormat:@"Login failed"]];
            }
        }];
    }
}
```

5.3.4 Adding RPC interceptor to simulate the action of sending RPC requests for login status

After login, we can send the RPC request requiring the login status. To this end, we make a convention that the sessionId obtained after the login should be placed in the RPC request header. (In usual cases, the client software is implemented through HTTP Cookie. Here we make it simpler.)

Add a self-defined RPC interceptor 'FCHeaderSessionRpcInterceptor'. This interceptor will intercept all the RPC requests. If it is not a login request, the sessionId field will be added into the header.

```
@interface FCHeaderSessionRpcInterceptor : NSObject <DTRpcInterceptor>
@end
```

```
#import "LoginService.h"
@implementation FCHeaderSessionRpcInterceptor

- (DTRpcOperation *)beforeRpcOperation:(DTRpcOperation *)operation
{
    if (![operation.rpcOperationType isEqualToString:@"alipay.mpaasgw"])
    {
        // Place the session into the header.
        id<LoginService> service = [DTContextGet() findServiceByName:@""];
        if (service.sessionId)
        {
            NSMutableURLRequest* request = (NSMutableURLRequest*)operation.request;
            [request setValue:[service.sessionId copy] forHTTPHeaderField:@"sessionId"];
        }
    }
    return operation;
}
@end
```

Add 'FCHeaderSessionRpcInterceptor' to the RPC interceptor container class.

```
@implementation DTRpcCommonInterceptor

- (id)init
{
    self = [super init];
    if (self)
    {
        NSMutableArray *interceptorList = [[NSMutableArray alloc] init];
        FCHeaderSessionRpcInterceptor *hsInterceptor = [[FCHeaderSessionRpcInterceptor alloc] init];
        [interceptorList addObject:hsInterceptor];
        self.interceptorArray = interceptorList;
    }
    return self;
}

@end
```

Notify RPC server of the interceptor container class name in mPaasAppInterfaceImp callback.

```
- (NSString*)appRPCCommonInterceptorClassName
{
    return @"DTRpcCommonInterceptor";
}
```

As gzip is not enabled on the testing gateway, we should disable gzip compression on RPC. This piece of code can be placed under the 'appDelegateEvent:arguments:' method of mPaasAppInterfaceImp, and implemented in 'mPaasAppEventAfterDidFinishLaunching' events.

```
// The testing gateway does not support gzip compression.  
DTRpcConfig* demoRpcConfig = [[[DTRpcClient defaultClient] configForScope:kDTRpcConfigScopeDefault] mutableCopy];  
demoRpcConfig.requestGZip = NO;  
[[[DTRpcClient defaultClient] setConfig:demoRpcConfig forScope:kDTRpcConfigScopeDefault] release];
```

Next, we will add codes to simulate the action of sending a RPC request.

```
_block NSString* result = nil;  
[self callAsyncBlock:^{  
    @try  
    {  
        DTRpcMethod *method = [[DTRpcMethod alloc] init];  
        method.operationType = @"alipay.mpaasgw.needlogintest";  
        method.checkLogin = NO;  
        method.returnType = @":\"NSString\"";  
        method.signCheck = YES;  
  
        result = [[DTRpcClient defaultClient] executeMethod:method parameters:nil completion:^(  
            NSLog(@"%@", operation);  
        )];  
    }  
    @catch(DTRpcException *exception)  
    {  
        result = exception.reason;  
    }  
} completion:^{  
    [self showAlert:result];  
}];
```

5.3.5 Developing interfaces and adding codes simulating crashes

First, enable the crash reporting feature in the main function.

```

int main(int argc, char * argv[]) {
    enable_crash_reporter_service();
    @autoreleasepool {
        mPaasInit(@"SIMPLEDEMORPC_IOS", [mPaasAppInterfaceImp class]);
        return UIApplicationMain(argc, argv, @"DFApplication", @"DFC1");
    }
}

```

Then, add a piece of code simulating the crash.

```

- (void)raiseOutOfRangeException
{
    NSArray * array = @[@"Hello", @"World"];
    NSString * str = [array objectAtIndex:2];
    NSLog(@"String value:%@", str);
}

```

OK. Our demo project has been complete. Now upload the codes to SVN.

5.3.6 Testing demo application

First, upload the certificate used for the packaging (a zip package containing the certificate (.p12) and the provisioning profile (.mobileprovision) files).

The screenshot shows a user interface for managing certificates. At the top, there is a blue button labeled "Add Certificate". To the right, there are three tabs: "All" (which is selected), "iOS", and "Android". Below the tabs is a table with the following data:

Platform	Name	Creation Time	Operation
ios	ios01	2015-11-20 15:15:51	Details Delete
android	Android	2015-11-20 15:16:37	Details Delete

Upload the certificate (.p12) for Apple Push service.

The screenshot shows the 'Message Management' section of the Mpaas test interface. On the left, there's a sidebar with options like 'Mobile', 'APP Management', 'Build & Release', 'Service Management', 'Statistics Analysis', 'Message Management' (which is expanded), 'Pushing Message', 'iOS Push Certificate' (which is selected and highlighted in blue), and 'Template Management'. At the top right, there are buttons for 'Go Back', 'Create APP', and '蚂蚁金融' (Ant Financial). The main area is titled 'Message Management / iOS Push Certificate'. It displays a table of properties and their values:

Properties	Value
alias	mpaasdemo_distribution
Environment	Production Environment
Host	gateway.push.apple.com
Port	2195
Effect Time	2015-05-25T15:15:43+08:00
Expiration Time	2016-05-24T15:15:43+08:00
issuerDN	CN=Apple Worldwide Developer Relations Certification Authority, OU=Apple Worldwide Developer Relations, O=Apple Inc., C=US
subjectDN	C=US, OU=LQ38NAVXP6, CN=Apple Production IOS Push Services: com.alipay.mpaasdemo, UID=com.alipay.mpaasdemo

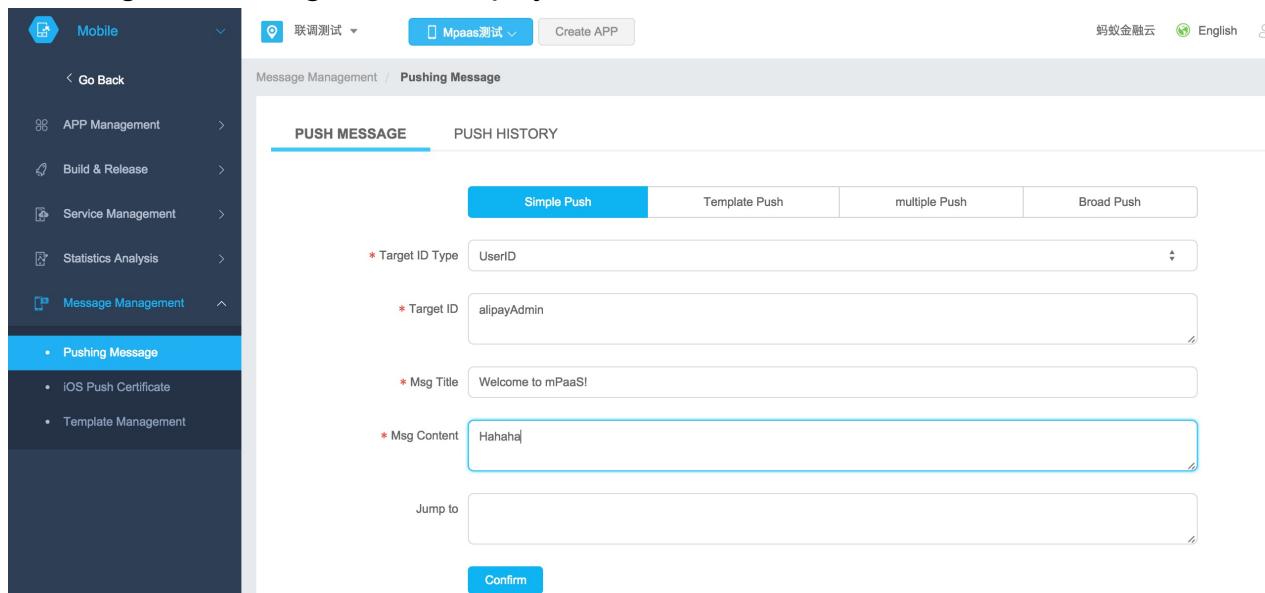
At the bottom left of this area is a 'Re-Upload' button.

After the build, install the demo application to a mobile phone and log in using the 'alipayAdmin' account (Password: ali123). After successful login, click to send a RPC request and you will see a prompt returned: "Welcome

to mPaaS".



On the 'Development on Cloud->Push->Push Notification' interface, push a message to the login user 'alipayAdmin'.



The message is received successfully.



Click "Make a Crash" to simulate a crash of the demo application. Restart the demo application to upload the crash report to the server. Two minutes later, we will see the crash log on the website.

[Download complete codes of demo application](#)

6 Appendix

[TOC]

6.1 User-mode Processing

Third-party applications of mPaaS have their own account login systems, and some components of mPaaS require the userId of the accessed application for normal operation. In case of changes in the login status of the accessed application, the application should notify mPaaS using the method below. (User sign-out events are also included. For logout events, pass nil.)

6.1.1 Notifying Data Center

When Data Center is used, the application should notify Data Center of changes in the login status as soon as possible for Data Center to switch user databases. Then the application can notify other service layers of the change.

```
[[APDataCenter defaultDataCenter] setCurrentUserId:userId];
```

When a user signs out, you don't have to call the setCurrentUserId method. Data Center will continue to open the database of last user, which will not cause problems.

6.1.2 Notifying monitor point

```
[[APMonitorPointManager sharedInstance] setAuthenticated:YES];
[[APMonitorPointManager sharedInstance] setUserId:userId];
```

6.1.3 Adding mPaaSAppInterface callback

The mPaaSAppInterface protocol contains the currentUserId method, which will be used by some modules of mPaaS for requesting the current userId of the accessed application.

```
- (NSString*)currentUserID
{
    return userId;
}
```

6.2 Blocking NSLog output

During the development phase we will allow the output of console logs of the developer's own applications. Yet after the application is distributed, we expect to offer quick ways to block all the log output. Add the codes below to your application and control it with macros. The log redirection method in compiling an App Store version is empty.

```
#include "stdio.h"

// In non-DEBUG mode, the log will be disabled.
#ifndef DEBUG
#define HIDE_ON_RELEASE 1
#endif

// Whether to convert the escape characters in the log to Chinese char
#ifdef DEBUG
#define CONVERT_UNICODE_ESCAPES
#endif
```

```

#ifndef HIDE_ON_RELEASE
void NSLogv (NSString *format, va_list args) { }
int printf (const char *format, ...) { return 0; }
#endif

#ifndef CONVERT_UNICODE_ESCAPES
// When NSLog is used for outputting NSArray or NSDictionary, the non-
static NSString* convertUnicodeCharEscapes(NSString* source)
{
    const NSInteger MAX_UNICHAR_LEN = 100;
    NSInteger lenSource = [source length];
    unichar unicharBuf[MAX_UNICHAR_LEN]; // A maximum of 100 unichar
    NSMutableString* dest = [[NSMutableString alloc] initWithCapacity:
    NSInteger i = 0, copyStart = 0, slashPos = -2, escapeIndex = -1, escapeCharValue = 0;
    NSInteger escapePow = 16 * 16 * 16;

    while (i < lenSource)
    {
        unichar c = [source characterAtIndex:i];
        if (escapeIndex >= 0)
        {
            if (escapeIndex == 3)
            {
                escapeCharValue = 0;
                escapePow = 16 * 16 * 16;
            }

            // Convert to the hexadecimal character.
            if (c >= '0' && c <= '9')
                escapeCharValue += escapePow * (c - '0');
            else if (c >= 'a' && c <= 'f')
                escapeCharValue += escapePow * (c - 'a' + 10);
            else
            {
                // something wrong
                if (escapeCharIndex > 0)
                    [dest appendString:[[NSString alloc] initWithCharacterAtIndexes:&unicharBuf[copyStart, 5 - escapeIndex]]];
                copyStart = i - (5 - escapeIndex);
                escapeIndex = -1;
            }
        }
        else
            dest = [dest stringByAppendingString:[source substringFromIndex:i]];
        i++;
    }
    return dest;
}
#endif

```

```
        escapeIndex = -1;
        i++;
        continue;
    }

    if (escapeIndex == 0)
    {
        unicharBuf[escapeCharIndex] = (unichar)escapeCharValue;
        copyStart = i + 1;
        // Judge whether \U still exists.
        if (escapeCharIndex < MAX_UNICHAR_LEN - 1 && i < lenSoFar)
        {
            escapeCharIndex++;
            escapeIndex = 3;
            i += 2;
        }
        else
        {
            escapeIndex = -1;
            [dest appendString:[[NSString alloc] initWithCharacters:&unicharBuf[copyStart] length:(lenSoFar - copyStart)]];
        }
    }
    else
    {
        escapeIndex--;
        escapePow /= 16;
    }
}
else if (c == '\\\\')
{
    slashPos = i;
}
else if (c == 'U')
{
    if (slashPos == i - 1)
    {
        escapeIndex = 3;
        escapeCharIndex = 0;
    }
}
```

```

        if (i - 2 - copyStart + 1 > 0)
        {
            [dest appendString:[source substringWithRange:NSMakeRange(i - 2 - copyStart + 1, 1)];
            copyStart = i - 1;
        }
    }
    i++;
}

if (copyStart == 0)
    return source;
else if (lenSource - copyStart > 0)
    [dest appendString:[source substringWithRange:NSMakeRange(copyStart, lenSource - copyStart)];
return dest;
}
#endif

static void _NSLog(id NIL, ...)
{
    va_list args;
    va_start(args, NIL);
    NSLogv(@"%@", args);
    va_end(args);
}

void NSLog (NSString *format, ...)
{
#if HIDE_ON_RELEASE
    // EMPTY REWRITE
#else
    va_list args;
    va_start(args, format);
    NSString *logString = [[NSString alloc] initWithFormat:format args];
    va_end(args);
#endif
    #ifdef CONVERT_UNICODE_ESCAPES
        logString = convertUnicodeCharEscapes(logString);
    #endif
}

```

```

    _NSLog(nil, logString); // This is necessary. Or else it is hard to
#endif
}

```

6.3 List of third-party library

Chinese Name	English Name	Function	Version	Download URL
无线保镖	SecurityGuardSDK	A black box storing the encrypted data and keys	1.5.32	Download
设备唯一标识	UTDID	To generate the unique ID of the device	1.0.2	Download
OpenSSL	APOpenSSL	OpenSSL lightweight encapsulation	1.0.0	Download
设备指纹	DevicePrint	Device fingerprint	1.2.0	Download

@cnName 1 Overview @priority 1

1 Overview

[TOC]

1.1 Design concept of client framework

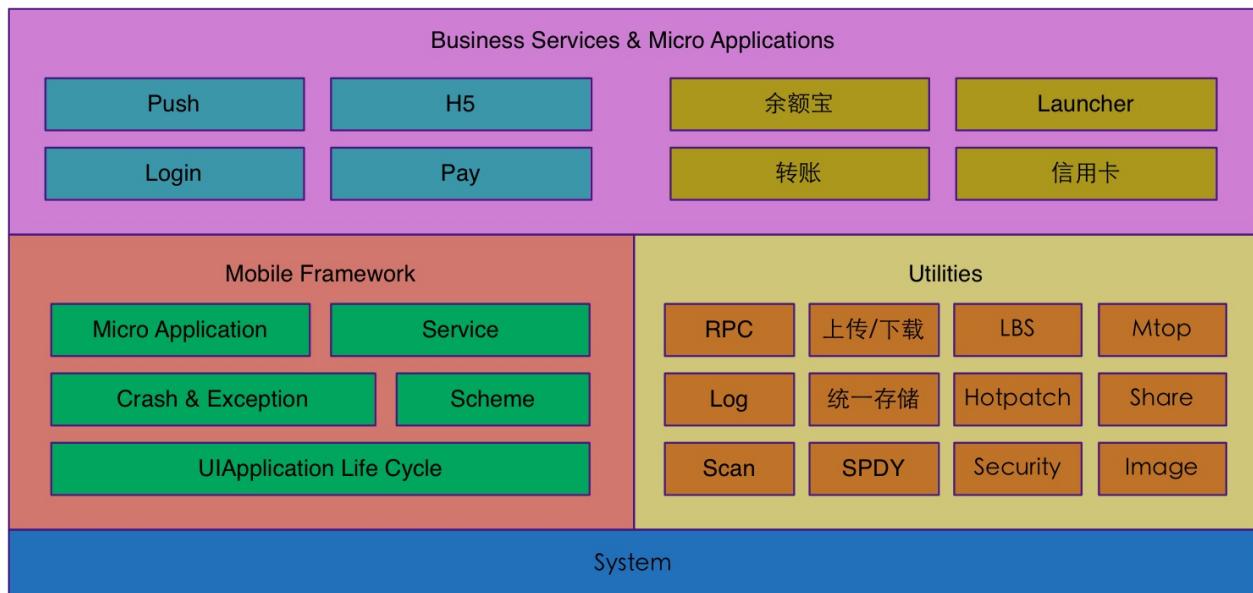
- The core designing concept of the framework is to separate the business into relatively independent modules to achieve high cohesion and loose coupling between modules.
- The framework divides modules with or without UI interfaces into the 'Micro Application' or 'Service' respectively, and defines their base classes, namely 'DTMicroApplication' and 'DTService', for life cycle management.
- Application is an independent business procedure, while Service offers universal services.
- Service has a status. Once it is initiated, it always exists throughout the life cycle of the client.
- Service is executed in the background and has no UI elements.

1.2 Roles of client framework

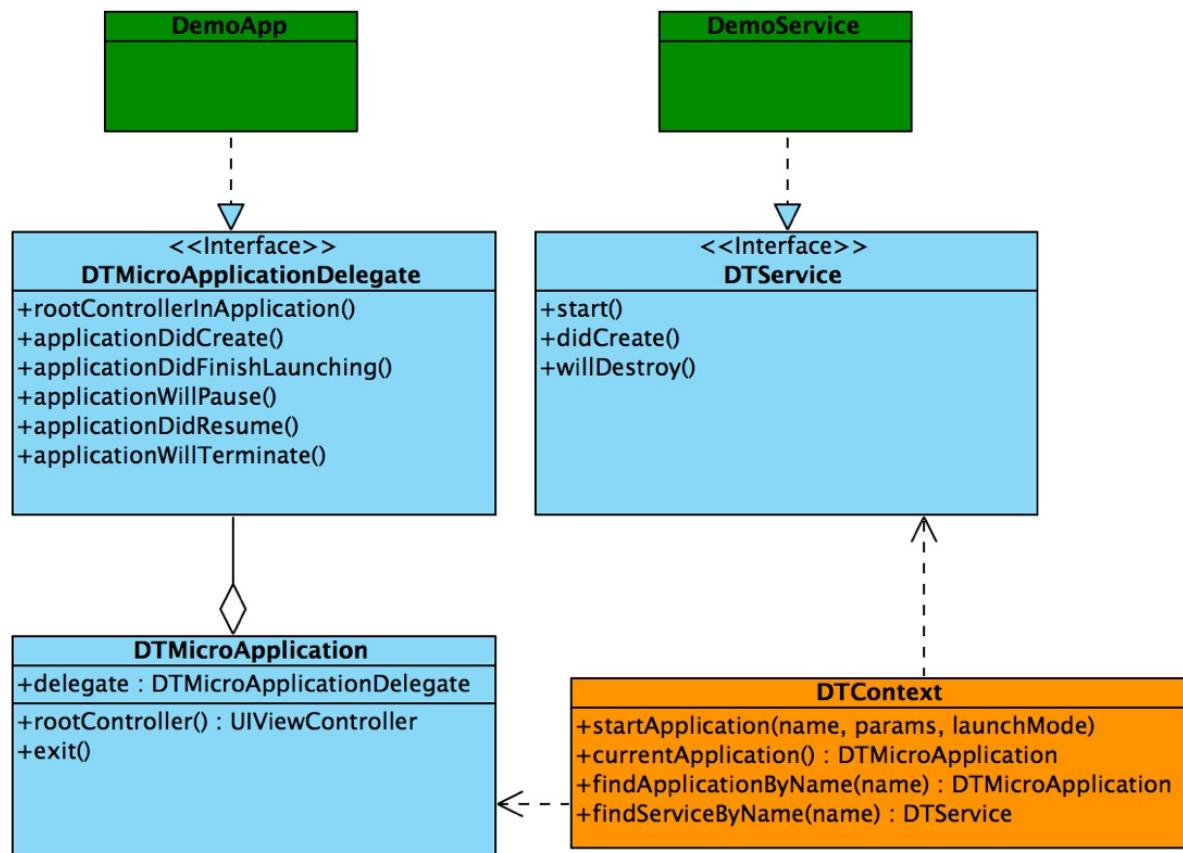
- Managing the application life cycle, and event processing and dispatching of the UIApplication delegates.
- Managing Micro Application and Service and handling application life cycle and navigation to other applications.
- Handling crash capture and reporting on the client.
- Providing DTViewController, DTSchemeHandler and other base classes.

1.3 Client architecture of wallet

In the figure, the Mobile Framework is the client framework.



1.4 UML diagram of Micro Application and Service management



@cnName 2 Basic Terminologies @priority 2

2 Basic Terminologies

Chinese Terminology	English Terminology	Description
微组件上下文	Micro Application Context	Runtime contexts of micro applications of the mPaaS client
微应用	Micro Application	Logical groupings of runtime user interfaces of the mPaaS client
微服务	Micro Service	Lightweight services provided by the mPaaS client during the runtime
多 Tab 应用	Multi-Tab Application	Applications such as Alipay and WeChat that enable switching between multiple tabs
客户端框架	Mobile Framework	Framework originating from the Alipay client and managing micro applications, services and life cycles

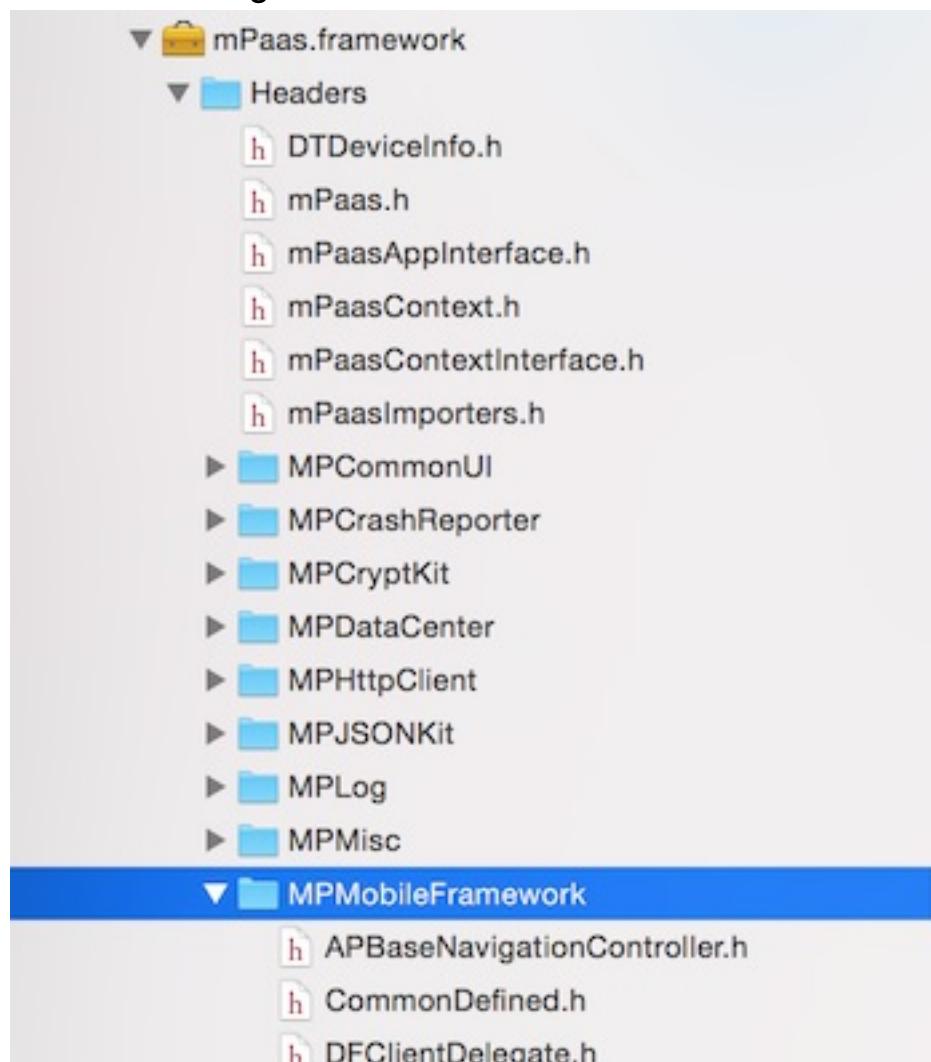
@cnName 3 Quick Start @priority 3

3 Quick Start

[TOC]

3.1 Module of client framework

The client framework module is named 'MPMobileFramework' in mPaaS, as shown in the figure:



You only need to reference the mPaas.h file instead of referencing all the header files separately. If you used our tools to generate the template project, the reference of the mPaas.h file is also done.

3.2 Using client framework

The core objective of accessing the mPaaS client framework is to manage the application life cycle with the framework. To achieve this, you only need to specify to use DFApplication and DFClientDelegate in the main.m file.

```
int main(int argc, char * argv[]) {
    enable_crash_reporter_service();
    @autoreleasepool {
        mPaasInit(@"BASKETBALL_IOS", [mPaasAppInterfaceImp class]);
        return UIApplicationMain(argc, argv, @"DFApplication", @"DFC1:");
    }
}
```

3.3 Framework context (DTContext)

- The control center of the framework. Every micro application interacts with the framework through the context.
- It provides interfaces for starting micro applications, searching for micro applications by name, disabling micro applications and managing navigation between micro applications.
- Registering, discovering and unregistering services.

3.3.1 Getting context interface

```
DTContext * DTContextGet();
```

3.3.2 Interface of micro application management

```
/**
 * Start an application with the specified name.
```

Table of Contents |

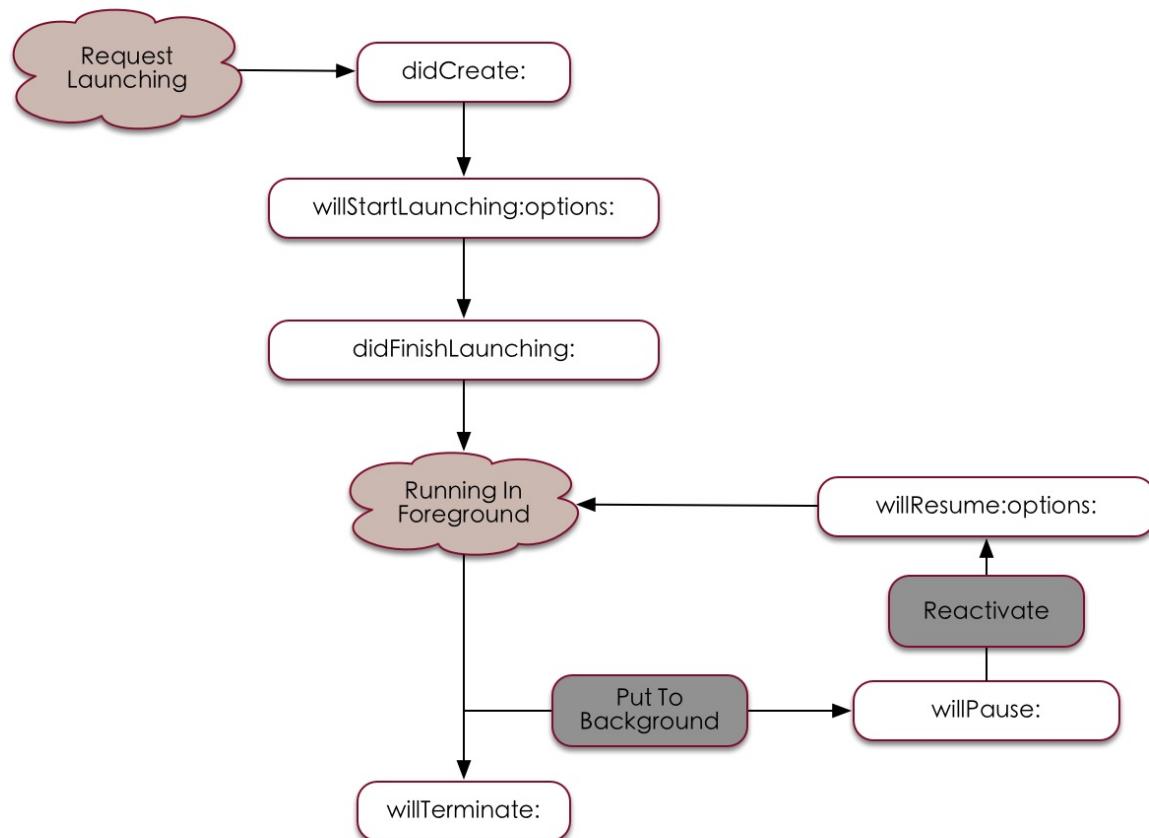
```
* @param name Name of the application to be started.  
* @param params Parameters to be passed to another application when it  
* @param animated Whether to display animations when the specified application  
*  
* @return YES is returned for successful application start; otherwise NO.  
*/  
- (BOOL)startApplication:(NSString *)name params:(NSDictionary *)params  
  
/**  
 * Start an application with the specified name.  
 *  
 * @param name Name of the application to be started.  
 * @param params Parameters to be passed to another application when it  
 * @param launchMode Specifies the startup mode of the application.  
 *  
 * @return YES is returned for successful application start; otherwise NO.  
*/  
- (BOOL)startApplication:(NSString *)name params:(NSDictionary *)params  
  
/**  
 * Look for the specified application.  
 *  
 * @param name Name of the application you are looking for.  
 *  
 * @return If the specified application is already in the application stack.  
 */  
- (DTMicroApplication *)findApplicationByName:(NSString *)name;  
  
/**  
 * Return applications on the top of the stack, i.e., applications visible.  
 *  
 * @return Currently visible applications.  
 */  
- (DTMicroApplication *)currentApplication;
```

3.3.3 Interfaces of service management

```
/**  
 * Look for a service with the specified name.  
 *  
 * @param name Service name.  
 *  
 * @return If the specified application is found, a service object wi.  
 */  
- (id)findServiceByName:(NSString *)name;  
  
/**  
 * Register a service.  
 *  
 * @param name Service name.  
 */  
- (BOOL)registerService:(id)service forName:(NSString *)name;  
  
/**  
 * Unregister an existing service.  
 *  
 * @param name Service name.  
 */  
- (void)unregisterServiceForName:(NSString *)name;
```

3.4 Micro application (DTMicroApplication)

3.4.1 Life cycle of micro application



3.4.2 DTMicroApplication

```

/**
 * Descriptions of the application.
 */
@property(nonatomic, strong) DTMicroApplicationDescriptor *descriptor.

/**
 * Delegate of the current application.
 */
@property(nonatomic, strong) id <DTMicroApplicationDelegate> delegate.

/**
 * Run mode of the application.
 */
@property(nonatomic, assign) DTMicroApplicationLaunchMode launchMode;

```

```
/**  
 * Get the root controller of the current application.  
 *  
 * @return The root controller object of the current application. This  
 */  
- (UIViewController *)rootController;  
  
/**  
 * Close the application.  
 *  
 * @param animated Whether to display animations when the specified ap  
 */  
- (void)exitAnimated:(BOOL)animated;  
  
/**  
 * Process Push messages.  
 *  
 * @param params Push data.  
 *  
 * @return YES is returned for successful processing. Otherwise, NO.  
 */  
- (BOOL)handleRemoteNotifications:(NSDictionary *)params;
```

2.3.4.3 DTMicroApplicationDelegate

```
@required  
/**  
 * Request the application object delegate to return the root view controller.  
 *  
 * @param application Application object.  
 *  
 * @return The root view controller of the application.  
 */  
- (UIViewController *)rootControllerInApplication:(DTMicroApplication *)application;  
  
@optional
```

Table of Contents |

```
/**  
 * Notify the application delegate that the application object has been  
 *  
 * @param application Application object.  
 */  
- (void)applicationDidCreate:(DTMicroApplication *)application;  
  
/**  
 * Notify the application delegate that the application is about to start  
 *  
 * @param application The started application object.  
 * @param options Operational parameters of the application.  
 */  
- (void)application:(DTMicroApplication *)application willStartLaunch:  
  
/**  
 * Notify the application delegate that the application has been started.  
 *  
 * @param application The started application object.  
 */  
- (void)applicationDidFinishLaunching:(DTMicroApplication *)application;  
  
/**  
 * Notify the application delegate that the application is about to run.  
 *  
 * @param application The started application object.  
 */  
- (void)applicationWillPause:(DTMicroApplication *)application;  
  
/**  
 * Notify the application delegate that the application is about to be activated.  
 *  
 * @param application The application object to be activated.  
 */  
- (void)application:(DTMicroApplication *)application willResumeWithOptions:  
  
/**
```

Table of Contents |

```
* 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, with the  
*/  
- (void)application:(DTMicroApplication *)application didResumeWithOptions:  
  
/**  
 * Notify the application delegate that the application is about to exit.  
*  
* @param application Application object.  
*/  
- (void)applicationWillTerminate:(DTMicroApplication *)application;  
  
/**  
 * Notify the application delegate that the application is about to exit.  
*  
* @param application Application object.  
* @param animated Whether to exit with animations.  
*/  
- (void)applicationWillTerminate:(DTMicroApplication *)application animated:  
  
/**  
 * Inquire the application delegate about whether the application is ready to exit.  
* Attention: Make sure to return NO only for special circumstances, as  
*  
* @param application Application object.  
*  
* @return Ready for the exit?  
*/  
- (BOOL)applicationShouldTerminate:(DTMicroApplication *)application;
```

3.5 Service (DTService)

- Services are slightly different from micro applications in that services run in the background.
- After a service is started, it usually exists throughout the life cycle of the client and can be accessed at any time.
- Methods can be called between services or between services and micro applications (for example, for executing a function or obtaining data).

3.5.1 DTService

```
@required
```

```
/**  
 * Start a service.  
 * Attention:  
 * The framework will call this method after it completes the initial:  
 * If a service wants to start an application, it cannot start other a  
 */  
- (void)start;
```

```
@optional
```

```
/**  
 * The service is created.  
 */  
- (void)didCreate;  
  
/**  
 * The service will be destroyed.  
 */  
- (void)willDestroy;
```

3.5.2 Creating service template

Define the service protocol.

```
@protocol MyService <DTService>

// Define the interface.
- (void)doTask;

@end
```

Implement the service.

```
@interface MyServiceImpl : MyService
@end

@implementation MyServiceImpl

- (void)doTask
{
    // TODO: Add your code here...
}

@end
```

3.6 Modifying micro application and service configuration

To use the client framework, the application should incorporate the MobileRuntime.plist configuration file (projects created from the template already have this file). The structure is as follows:

< > FinancialCloud > FinancialCloud > Supporting Files > MobileRuntime.plist > No Selection		
Key	Type	Value
▼ Root	Dictionary	(3 items)
Launcher	String	20000001
▼ Services	Array	+ (2 items)
► Item 0	Dictionary	(3 items)
▼ Item 1	Dictionary	(3 items)
class	String	LoginServiceImpl
lazyLoading	Boolean	NO
name	String	LoginService
▼ Applications	Array	(4 items)
▼ Item 0	Dictionary	(3 items)
delegate	String	AboutAppDelegate
description	String	About
name	String	20000004
► Item 1	Dictionary	(3 items)
► Item 2	Dictionary	(3 items)
► Item 3	Dictionary	(3 items)

3.6.1 Service configuration item

Field	Description
name	Unique ID of the service
class	The implementation class of services. When creating the service, the framework will utilize runtime reflection to create instances of the service implementation class
lazyLoading	Whether to enable lazy loading. If lazy loading is enabled, the service will not be instantiated at the framework startup. Instead, the service will be instantiated and started only when it is used. If lazy loading is not enabled, the service will be instantiated and started at the framework startup. By default, the value is NO.

3.6.2 Micro application configuration item

Field	Description
delegate	The class name of DTMicroApplicationDelegate implemented by the application
description	Descriptions of the application
name	Application name, usually in digit serial number. The startApplication method uses the name to start the application

4 Guide to Advanced Features

[TOC]

4.1 Handling scheme

4.1.1 DTService & DTServiceHandler

The framework opens up the DTService and the DTServiceHandler base class. Once a third-party application is accessed to the framework, the developer only needs to configure the DTService into the MobileRuntime.plist of the project.

▼ Services		Array	(3 items)
▼ Item 0		Dictionary	(3 items)
class		String	DTServiceServiceImpl
lazyLoading		Boolean	NO
name		String	DTService

The ways for handling different schemes can be implemented in different subclasses of the DTServiceHandler.

Table of Contents |

```
/*
 * \code DFSchemeHandler class is used for handling the calling mechanism.
 *
 */
@interface DTSSchemeHandler : NSObject

/**
 * Try to register a subclass of <code>DFSchemeHandler</code>, and enable
 *
 * @param handlerClass The subclass of the <code>DFSchemeHandler</code>
 *
 * @return YES is returned for successful registration; otherwise, NO
 */
+ (BOOL)registerClass:(Class)handlerClass;

/**
 * Unregister the handler class of open URL.
 *
 * @param handlerClass The handler class of open URL.
 */
+ (void)unregisterClass:(Class)handlerClass;

+ (BOOL)canHandleOpenURL:(NSURL *)aURL;

- (BOOL)handleOpenURL:(NSURL *)aURL userInfo:(NSDictionary *)userInfo;

@end
```

The application re-implements the methods below in the mPaasAppInterface protocol:

```
/**  
 * It returns the scheme handler class to be added. The returned hand  
 */  
- (NSArray*)appSchemeHandlerClasses;
```

Return the subclass name of the scheme handler in arrays. (The array element is NSString.)

4.1.2 Not using SchemeHandler

If you don't want to use SchemeHandler, you can use the appDelegateEvent method of mPaasAppInterface protocol for handling mPaasAppEventQueryOpenURL events.

```
- (id)appDelegateEvent:(mPaasEventType)event arguments:(NSDictionary*)  
{  
    if (event == mPaasAppEventQueryOpenURL) {  
  
    }  
    return @(YES);  
}
```

@cnName 4.2 Handling APNS @priority 2

4.2 Handling APNS

4.2.1 Getting device token

We recommend that the application create a PushService for handling the APNS device token. The application can register for the monitoring notification service of UIApplicationDidRegisterForRemoteNotifications so

that when the DFClientDelegate class gets the device token, it will return the token using the notification. The codes are as follows:

```
@protocol PushService <DTService>

@property (nonatomic, strong, readonly) NSString* pushToken;

@end

@interface PushServiceImpl : NSObject <PushService>
{
    NSString* _pushToken;
}
@end

@implementation PushServiceImpl

@synthesize pushToken = _pushToken;

- (void)start
{
    NSLog(@"PushServiceImpl started");

    // Push notification registration
#ifndef __IPHONE_8_0      //Push notification registration of iOS8
    if ([[UIDevice currentDevice] systemVersion] floatValue] >= 8.0f)
        UIUserNotificationSettings *settings = [UIUserNotificationSettings settingsForTypes:(UIUserNotificationTypeAlert | UIUserNotificationTypeSound | UIUserNotificationTypeBadge) categories:nil];
        [[UIApplication sharedApplication] registerUserNotificationSettings:settings];
#else
    [[UIApplication sharedApplication] registerForRemoteNotifications];
#endif
}

[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(applicationDidBecomeActive:)
                                         object:[UIApplication sharedApplication]];
```

```

    name: UIApplicationDid
}

- (void)applicationDidRegisterForRemoteNotifications:(NSNotification *
{
    NSString* token = [[notification.object description] stringByTrimmingCharactersInSet:[NSCharacterSet whitespaceAndNewlineCharacterSet]];
    _pushToken = [token stringByReplacingOccurrencesOfString:@" " withString:@""];
    NSLog(@"%@", token);
}

@end

```

4.2.2 Reporting device token

The developer of the application background should provide the RPC interface for the client for reporting the device token. If the login name (userId) is employed for pushing the notification, the login account of the application should also be bound to the device token for reporting. For details, contact the personnel of the application background.

4.2.3 Receiving Push notification

The DFClientDelegate class will throw two notifications as below when it receives Remote | Local Notifications, and returns the userInfo in the userInfo of the NSNotification.

```

NSString *const UIApplicationDidReceiveRemoteNotification = @"UIApplicationDidReceiveRemoteNotification";
NSString *const UIApplicationDidReceiveLocalNotification = @"UIApplicationDidReceiveLocalNotification";

```

The userInfo can also be obtained using the several events as follows defined by mPaasAppInterface, in which case the userInfo will be returned in the dictionary.

```
- (id)appDelegateEvent:(mPaaSEventType)event arguments:(NSDictionary*)arguments
```

```
mPaaSAppEventDidReceiveRemoteNotification, // Called at the entry of @selector(mPaaSAppEventDidReceiveRemoteNotification)
mPaaSAppEventDidReceiveRemoteNotificationFetchCompletion, // @selector(mPaaSAppEventDidReceiveRemoteNotificationFetchCompletion)
mPaaSAppEventDidReceiveLocalNotification, // Called at the entry of @selector(mPaaSAppEventDidReceiveLocalNotification)
```

4.3 Crash capture and reporting

The mPaaS client framework integrates the crash reporting feature, in which the crash monitoring encapsulates open-source PLCrashReporter. It has the following features:

- Fully automatic capture of crash logs, logging and log reporting. Users do not need to care about it.
- The crash log contents grow more detailed, with the window stack and gesture information available.
- The background supports the view of the reverse resolution logs, and log filtering and queries.

4.3.1 Enabling crash reporting feature

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

int main(int argc, char * argv[]) {
    enable_crash_reporter_service();
    @autoreleasepool {
        mPaasInit(@"THFUND_IOS", [mPaasAppInterfaceImp class]);
        return UIApplicationMain(argc, argv, @"DFApplication", @"DFClient");
    }
}
```

Call `enable_crash_reporter_service()` method in the main function to enable log reporting. (The log reporting is dependent on the monitoring system, so the monitoring gateway URL should be configured.)

4.4 Apple Watch

The `DFClientDelegate` class of the framework will handle delegate methods of `UIApplication`:

```
- (void)application:(UIApplication *)application handleWatchKitExtens:
```

Monitoring notification, the original arguments `userInfo` and `reply` will be placed in the dictionary and returned in `userInfo` of notifications.

```
NSString *const UIApplicationWatchKitExtensionRequestNotifications = (
```

Or the `appDelegateEvent` method defined by `mPaasAppInterface` can be used for handling the `mPaasAppEventHandleWatch` event.

```
- (id)appDelegateEvent:(mPaasEventType)event arguments:(NSDictionary*)arguments;
```

@cnName 5 FAQs @priority 5

5 FAQs

[TOC]

I. How to use your UIApplication delegate class?

You can declare that your class inherits from DFClientDelegate, overwrite the DFClientDelegate method, and start the application using your class in main.m.

II. How to exit all micro applications and back to the Launcher?

```
[DTContextGet() startApplication:@"AppId of Launcher" params:nil anima
```

III. 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:@"A 的 appid" params:@{@{"arg": @"some
```

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

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

@cnName 6 Release History @priority 6

6 Release History

@cnName 7 Appendix @priority 7

7 Appendix

@cnName 1 Overview @priority 1

1 Overview

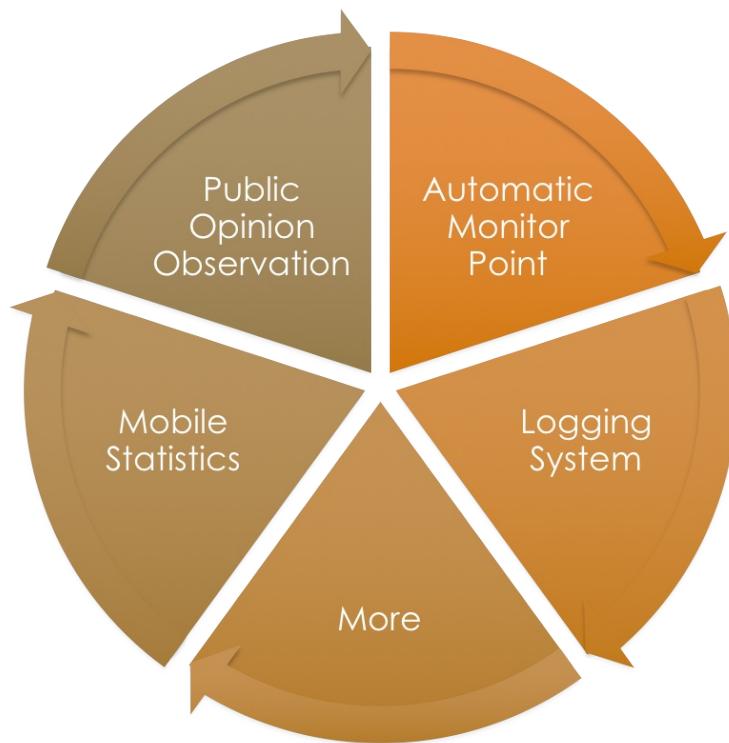
[TOC]

1.1 Introduction

The client monitoring and log reporting are an indispensable part of the best wireless applications. The client monitoring logs are always crucial for analyzing and solving issues.

The mPaaS client monitoring system offers a wide variety of monitoring means and log categories, covering crashes, networks, storage, threads, memory, user behaviors, etc. The matched background is capable of handling reporting of massive user data, and isolating valuable data from the massive data. The developer can have an intuitive view of such data on the primary site.

1.2 Components



1.2.1 Data statistics and trend analysis

It offers the statistics on the installed base, UV, PV, and version updates of applications, and analyzes user habits on the application in different regions and time periods, so as to provide reliable data support for developers.

1.2.2 Public opinion monitoring

It keeps a watchful eye on feedbacks to the application in media, micro blogs or other channels, and identifies issues as early as possible. (This feature is still under development.)

1.2.3 Monitor point and automatic monitor point

With manual or automatic monitor points by the developer, the run status of applications can be captured for reconstruction of the problematical scenes to the greatest extent. In addition to the most basic monitor point, monitor points for the application memory, threads, traffic, startup time, and other performance indicators will also be set automatically.

@cnName 2 Basic Terminologies @priority 2

2 Basic Terminologies

Chinese Terminology	English Terminology	Description
埋点	Monitor Point	The event triggered when the application runs to a specific status. The event will be logged.
自动化埋点	Automatic Monitor Point	Through Hook UI component, the automatic monitor point will monitor user gestures automatically, as well as the interface switching, among other events, of the application.

@cnName 3 Quick Start @priority 3

3 Quick Start

[TOC]

3.1 Monitor module

A complete monitor module consists of the following components:

Library	Function
MPRemoteLogging	Logging module of monitor points
MPCrashReporter	Log capturing module
MPMonitor	Automatic monitor point and automatic performance monitoring module
MPLog	User-action log module

You only need to reference the mPaas.h file instead of referencing all the header files of these libraries separately. If you used our tools to generate the template project, the reference of the mPaas.h file is also done.

3.2 Configuring monitoring gateway URL and application ID

When the system Setting Service is enabled, configure the monitoring gateway URL and application ID to `GatewayConfig.plist`.

Key	Type	Value
▼ Root	Dictionary	(3 items)
▶ Debug	Dictionary	(6 items)
▶ Pre-release	Dictionary	(6 items)
▼ Release	Dictionary	(6 items)
mPaasPushServerGateway	String	
mPaasPushAppId	String	THFUND
mPaasLogServerGateway	String	http://10.218.157.65
mPaasLogProductId	String	THFUND_IOS-0000017768
mPaasRpcGateway	String	http://mobilegw.d1366.alipay.net/mgw.htm
mPaasRpcETagURL	String	http://mobilegw.d1366.alipay.net/mgw.htm

If the system Setting Service is not enabled, return the log configuration information with the two methods below in the implementation class of `mPaasAppInterface`.

```
/**
 * URL of the remote log server
 */
- (NSString*)appRemoteLogServerURL;

/**
 * Name of the log service application. Some arguments may be required
 */
- (NSString*)appRemoteLogProductId;
```

For more information, see [Using Setting Service](#)

4 Guide to Advanced Features

[TOC]

4.1 User diagnosis log

4.1.1 Functions

The user diagnosis log is not uploaded automatically. It logs files by date and by hour separately. The developer can push remote commands to the target mobile phone on the primary site, and the mobile phone of the target user will upload the log of a specified time period.

The user diagnosis log is located in the MPLog module, and references the MPLog/APLog.h. Usually it only needs to reference mPaas.h.

The working procedure of the user diagnosis log is as follows:

- User diagnosis log writes data into local files by time period to form independent log files.
- The developer passes arguments to the configuration log of the primary site and adds tasks for collecting diagnosis logs.
- After receiving the tasks, the client packages and uploads log files according to the network type and logging periods. The developer downloads logs from the primary site for viewing.

4.1.2 Log interface

The interfaces here are all available, but uploading by log level is not supported yet.

```
#define APLogError(tag,fmt, ...) \
APLogToFile(tag, kAPLogLevelError, fmt, ##_VA_ARGS__)

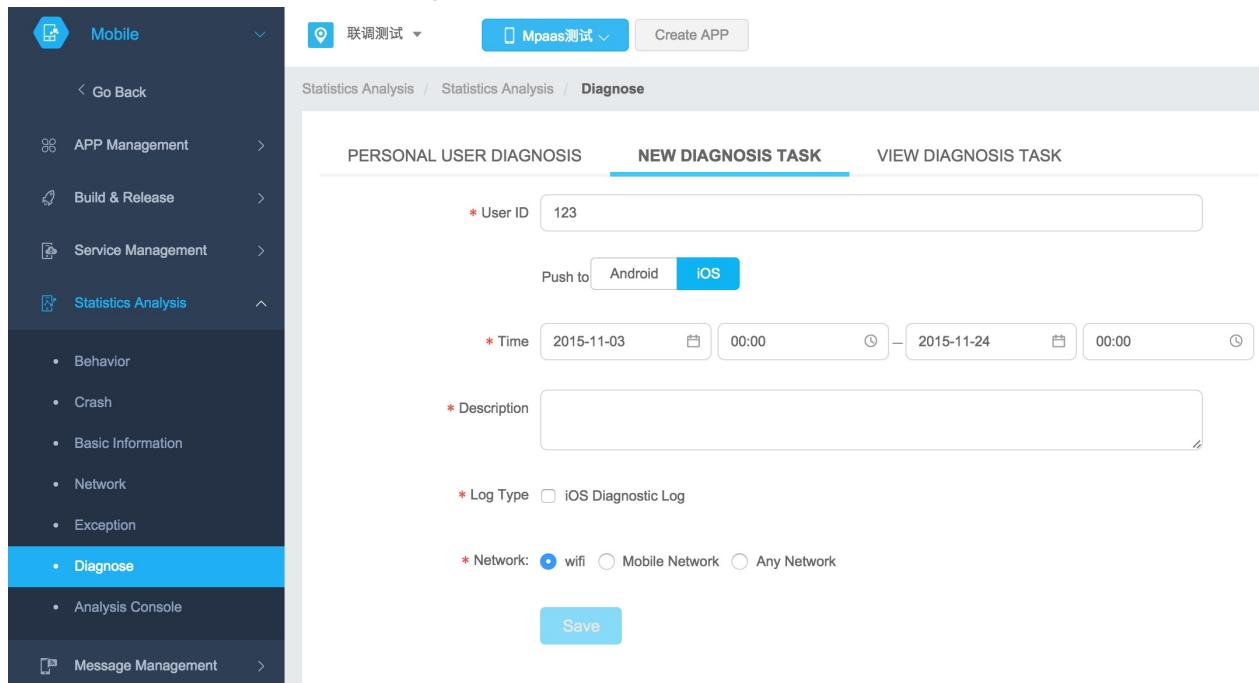
#define APLogWarn(tag,fmt, ...) \
APLogToFile(tag, kAPLogLevelWarn, fmt, ##_VA_ARGS__)

#define APLogInfo(tag,fmt, ...) \
APLogToFile(tag, kAPLogLevelInfo, fmt, ##_VA_ARGS__)

#define APLogDebug(tag,fmt, ...) \
APLogToFile(tag, kAPLogLevelDebug, fmt, ##_VA_ARGS_)
```

4.1.3 Creating collection task on primary site

Go to the primary site and select the application. In 'Statistic Analysis'- 'Diagnosis', select 'Add Collection Log' - iOS platform. Input the desired user ID and time periods for log collection, and click "Save".



The task you created can be found in 'View Collection Log'.

This screenshot shows the 'Diagnose' section of the Alipay Mpaas Diagnosis interface. On the left, there's a sidebar with 'Mobile' selected. Under 'Statistics Analysis', 'Diagnose' is highlighted. The main area shows a table of diagnosis tasks:

Creation Time	User ID	Description	Executor	Status	Operation
2015-11-30 15:32:01	123	Test	alipayAdmin@alipay.net	Push Success	View Restart Delete

Click 'View' to view the execution of the task. If the task is executed and the uploading is successful, you can download the log here.

This screenshot shows a 'Details' modal window. It displays task information and logs. The task info includes:

- Task Info: Task ID227
- User ID: 123
- Owner: alipayAdmin@alipay.net
- Description: Test
- Parameters: {}
- Task Status: Push Success [Restart](#)

The logs section shows:

- iOS Log: Time 2015-11-03 00:00:00 - 2015-11-24 00:00:00
- Network Category: WIFI
- Child Task Status: Unhandled

4.2 Client report-active

The client report-active logic is embedded in the framework. If the framework is not used, you can enable report-active feature with the method below.

```
[APRemoteLogger writeLogWithActionId:KActionID_Event extParams:nil app]
```

The default report-active logic of mPaaS client framework is:

- When there is no active application process, report once at application startup.
- When there are active application processes, report once for a switch from the background to the foreground.

To lower down the report-active frequency, you can rewrite the following method in mPaaSAppInterface and return an interval:

```
/**  
 * When the application is switched from the background to the foreground  
 * This will not affect the application startup when there is no active application  
 * @return Number of seconds  
 */  
- (NSInteger)appReportActiveMinIntervalSeconds;
```

4.3 Login reporting

The application can selectively upload user login events and pass userId in the report argument:

```
[APRemoteLogger writeLogWithActionId:KActionID_Event extParams:@[{"useId":  
"1234567890"}]]
```

4.4 Log module and uploading behavior

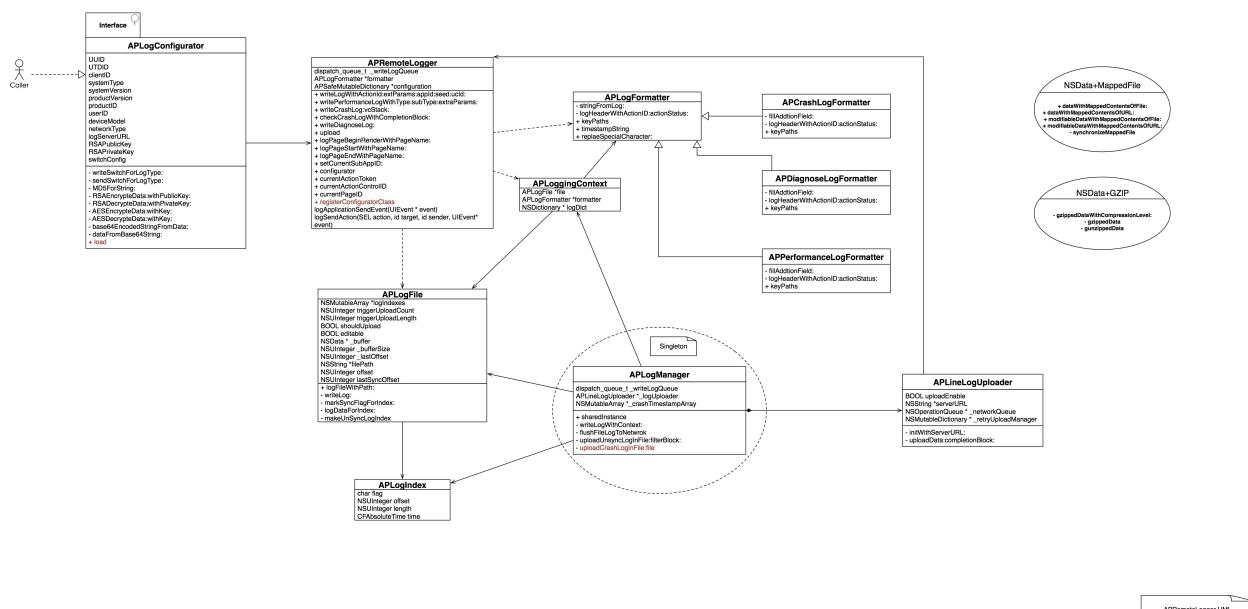
4.4.1 Working procedure of monitor point

- The monitor point writes logs into local files based on the monitor point interfaces called and incoming arguments. The file structure is one log in one line, with markers at the beginning of each log. The first figure

represents the uploading status of the log. 0 indicates that the log is not uploaded, and 1 indicates that the log is uploaded.

- Within a log file, if there are 100 logs, or the log size reaches 10 KB (the argument may be subject to slight changes), an uploading action will be triggered.
 - With the uploading successful, the log will be marked as uploaded. Otherwise the log will be uploaded again at the next uploading action triggered.
 - After the log is uploaded to the server, the server will categorize the logs into different log files based on the log file header or other arguments.
 - Crash logs are required to be uploaded in time. So the crash log will be uploaded every time when the application is restarted after a crash.

4.4.2 Log module class diagram



@cnName 5 FAQs @priority 5

5 FAQs

@cnName 6 Release History @priority 6

6 Release History

@cnName 7 Appendix @priority 7

7 Appendix

@cnName 1.1 Overview @priority 1

1.1 Overview

The RPC service remotely calls stable interfaces with open codes to the business level. But on the lower layer, it supports request sending through multiple channels such as HTTP, HTTPS, SPDY and Socket. It has the following features:

- The client calling codes are generated by tools automatically. The client does not need to care about the network communication protocols and the data formats used.
- The returned data from the server will be inversely solved to Objective-C objects automatically, with no additional coding required.
- RPC service provides optimization of data compression, cache and batch calling, and supports multi-lingual requests.
- Unified RPC exception processing, such as dialog box pop-ups and toasts.
- RPC supports RPC interceptors to achieve customized requests and processing.
- It is signature-verification-enabled.

@cnName 1.2 Basic Terminologies @priority 2

1.2 Basic Terminologies

Chinese Terminology	English Terminology	Description
远程过程调用	RPC	N/A
RPC 拦截器	RPC Interceptor	The RPC module calls the callback interface registered on a upper layer before and after sending the request so as to customize RPC requests and personalize error processing.
SPDY	SPDY	A TCP-based application layer protocol developed primarily at Google. It aims to minimize web page load latency and improve web speed to optimize web user experience.
RPC 网关	RPC Gateway	RPC requests must be dispatched by the gateway before they arrive at the business server. The gateway will also handle errors irrelevant to the business logics.

@cnName 1.3 Quick Start @priority 3

1.3 Quick Start

[TOC]

1.3.1 RPC module

The RPC module in mPaaS is MPHttpClient. The key classes and functions it contains include the following:

Class	Function
DTRpcClient	RPC client management class, singleton. It sends RPC requests and handles RPC request configurations.
DTRpcConfig	RPC configurations, including specifying the RPC gateway, cache policy and whether to use GZip. It can control the influence scope of every configuration item.
DTRpcMethod	Descriptions of RPC request methods. It records the method name, arguments and return types of RPC requests.
DTRpcOperation	Operating entity of RPC requests, a subclass of NSOperation.
DTRpcAsyncCaller	RPC asynchronous caller. It provides a quick interface for asynchronous RPC calls.

1.3.2 Gateway configuration

In implementation of mPaasApplInterface interfaces, the application returns the gateway URL using the interface below. If the system Setting Service is enabled, the RPC gateway URL can be configured in GatewayConfig.plist.

```
#pragma mark Network configurations

/**
 *  RPC server URL
 */
- (NSString*)appRPCGatewayURL;

/**
 *  ETag server URL of the RPC service
 */
- (NSString*)appRPCETagURL;
```

For more information, see [Using Setting Service](#)

1.3.3 Sending RPC request

The RPC request must be called in the subthread, while the callback method is in the main thread by default.

- Use DTRpcAsyncCaller

```
__block ALPRSAPKeyResult* result = nil;
[DTRpcAsyncCaller callAsyncBlock:^{
    @try
    {
        DTRpcMethod *method = [[DTRpcMethod alloc] init];
        method.operationType = @"alipay.client.getRSAKey";
        method.checkLogin = NO ;
        method.returnType = @"@\"ALPRSAPKeyResult\"";
        method.signCheck = YES;

        result = [ [DTRpcClient defaultClient] executeMethod:method p;
    }
    @catch (NSEException *exception) {
        NSLog(@"%@", exception);
    }
} completion:^{
    NSLog(@"%@", result);
}];
```

- Use DTViewController in the framework

In the example below, the RPC request is sent in the method of the ViewController subclass of the DTViewController.

```
__block ALPRSAPKeyResult* result = nil;
[self callAsyncBlock:^{
    @try
    {
        DTRpcMethod *method = [[DTRpcMethod alloc] init];
        method.operationType = @"alipay.client.getRSAKey";
        method.checkLogin = NO ;
        method.returnType = @"@\"ALPRSAPKeyResult\"";
        method.signCheck = YES;

        result = [ [DTRpcClient defaultClient] executeMethod:method p;
    }
    @catch (NSEException *exception) {
        NSLog(@"%@", exception);
    }
} completion:^{
    NSLog(@"%@", result);
}];
```

1.4 Guide to Advanced Features

[TOC]

1.4.1 Using RPC interceptor

1.4.1.1 Working principle of RPC interceptor

Below are the simplified codes of DTRpcClient for executing RPC operations. Before and after the request is sent, the DTRpcOperation object will be passed to every interceptor for processing. With no processing required, the interceptor can simply let the object pass.

```
- (id)executeOperation:(DTRpcOperation *)operation responseHeaderField:  
{  
    if (self.interceptor) {  
        [self.interceptor beforeRpcOperation:operation];  
    }  
  
    [self.requestQueue addOperation:operation];  
    [operation waitUntilFinished];  
  
    if (self.interceptor) {  
        operation = [self.interceptor afterRpcOperation:operation];  
    }  
  
    if (operation.error) {  
        [DTRpcException raise:(DTRpcErrorCode)operation.error.code message:  
    }  
  
    return operation.resultObject;  
}
```

1.4.1.2 Interface protocol of RPC interceptor

```

/**
 * Interceptor of RPC requests. All the RPC requests, before being se
 * will pass through the interceptor which will process the request ac
 */
@protocol DTRpcInterceptor <NSObject>

@optional

/**
 * Pre-interceptor of RPC requests. This method will be called before
 *
 * @param operation Current RPC request.
 */
- (DTRpcOperation *)beforeRpcOperation:(DTRpcOperation *)operation;

/**
 * Post-interceptor of RPC requests. This method will be called after
 *
 * @param operation Current RPC request.
 */
- (DTRpcOperation *)afterRpcOperation:(DTRpcOperation *)operation;

- (void)handleException:(NSError *)exception;

@end

```

- To register a RPC interceptor container

From the codes above, we can see that DTRpcClient will only log one interceptor object, but the application may have multiple interceptors processing RPC requests together. Here we can create an interceptor container implementing 'beforeRpcOperation' and 'afterRpcOperation' interfaces to dispatch events to all of its child interceptors.

This interceptor container should be added to the project by the developer. The class name should be returned in the method under mPaasAppInterface and RPC module will create objects for the container automatically.

```
- (NSString*)appRPCCommonInterceptorClassName
{
    return @"DTRpcCommonInterceptor";
}
```

RPC interceptor is a way to achieve control on RPC behaviors on the client. The application should use a Common Interceptor as the container class of all the interceptors. This Common Interceptor also implements '@protocol DTRpcInterceptor'. All the self-defined interceptors should be added to the Common Interceptor.

['Common Interceptor template code'](#)

- Basic error processing of RPC interceptors

The container interceptor cannot process any errors. The developer needs to connect their own interceptor for processing network errors. The Base Interceptor template provided below can process network errors and session expiration errors. The developer can modify it as needed and integrate it into the project code. Add this DTRpcBaseInterceptor to the interceptor container and it will be ready for use.

['Base Interceptor template code'](#)

1.4.2 RPC Request signature

- Signature logic

RPC requests have a signature mechanism to protect client requests from being tampered or falsified. The sign-check is done by mPaaS automatically. The basic signing method is to generate a string for the content in the `requestBody`, and encrypt the string with the key saved in the Security Guard. The signature is contained in the request and sent to the gateway. The gateway will sign the request with the same approach and verify whether the two signatures are consistent.

- To manage request sign-check

The `signCheck` property in the `DTRpcMethod` class specifies whether to sign the request or not:

```
/** Whether to sign */
@property(nonatomic, assign) BOOL signCheck;
```

In usual cases, all the RPC requests of the application require signature, which can be achieved through the RPC interceptor. Sample codes:

```
@implementation RPCDefaultSignCheckInterceptor

- (DTRpcOperation *)beforeRpcOperation:(DTRpcOperation *)operation
{
    operation.method.signCheck = YES;
}

@end
```

- To save signature key in Security Guard

The signature key used by the client is saved in Security Guard. You may use Xcode plugin of mPaaS Developer Tools to generate your own secure data.

When the application is generating a key file in the Security Guard, it creates an item in Appkey field, and the value is the AppKey of the application. The Appkey here must be consistent with the key used in mPaaS initialization in the mPaasInit method. The RPC module will use the key matched with this Appkey to sign the request.

```
int main(int argc, char * argv[]) {
    enable_crash_reporter_service();
    @autoreleasepool {
        mPaasInit(@"THFUND_IOS", [mPaasAppInterfaceImp class]);
        return UIApplicationMain(argc, argv, @"DFAApplication", @"DFAAppDelegate");
    }
}
```

@cnName 1.5 FAQs @priority 5

1.5 FAQs

[TOC]

I. Unable to send RPC requests on iOS9 platform

On the iOS9 platform, Apple poses a restriction by default that applications must use the HTTPS protocol. That is why the requests sending will fail if the debugging gateway is of the HTTP protocol. Solution: Add the following fields in the plist file of the application to allow the application to send HTTP requests.

▼ NSAppTransportSecurity	◆ Dictionary	(1 item)
NSAllowsArbitraryLoads	Boolean	YES

II. How to add special header files for RPC requests

Usually, the HTTP header of RPC requests is added by the RPC module by default. With any special header field existing in a certain RPC request, the special header field can be passed into the requestHeaderField dictionary at RPC invocation.

```
/**
 * Execute a RPC request according to the specified \code DTRpcMethod
 *
 * @param method An instance of the \code DTRpcCode type, describing the
 *               method to be executed.
 * @param params Parameters required by the RPC request.
 * @param field Header to be added to the request.
 * @param responseBlock HeaderFields value requiring callback of response
 *
 * @return If the request is successful, return an object of the specified
 *         type.
 */
- (id)executeMethod:(DTRpcMethod *)method params:(NSArray *)params re
```

The RPC interceptor can achieve the same effect. In the next example, we demonstrate how to add fields to the header while calling the RPC request of 'rpc.operationA'.

```
@implementation HeaderFillRpcInterceptor

- (DTRpcOperation *)beforeRpcOperation:(DTRpcOperation *)operation
{
    if (![operation.rpcOperationType isEqualToString:@"rpc.operationA"])
    {
        NSMutableURLRequest* request = (NSMutableURLRequest*)operation;
        [request setValue:@"field" forHTTPHeaderField:@"key"];
    }
    return operation;
}

@end
```

III. How to designate a special gateway for a RPC request?

After the global gateway of the application is configured, DTRpcConfig can be used to designate a special gateway for a RPC request.

The codes below designate the RPC request

'alipay.client.saveUserProposalInfo' to go through a specified gateway URL.

```
DTRpcConfig* config = [[[DTRpcClient defaultClient] configForScope:kDTRpcConfigScopeGlobal];
feedbackConfig.gatewayURL = [NSURL URLWithString:@"https://mobilegw.alipay.com/gateway.do"];
feedbackConfig.operationType = @"alipay.client.saveUserProposalInfo";
[[DTRpcClient defaultClient] setConfig:config forScope:kDTRpcConfigScopeGlobal];
```

@cnName 1.6 Release History @priority 6

1.6 Release History

@cnName 1.7 Appendix @priority 7

1.7 Appendix

@cnName 2 APNS @priority 2

2 APNS

2.1 Client

See [Mobile Framework-Guide to Advanced Features-Handling APNS](#)

2.2 Server and mPaaS primary site management

See [Wireless Notification Service](#)

@cnName 1 Hotpatch @priority 1

1 Hotpatch

[TOC]

1.1 Introduction

1.1.1 Components

App Store has a review process for distributing applications, which is not as flexible as it is on the Android platform. Hotpatch offers a quick way to fix severe bugs of a released application without the need to release a new version.

Hotpatch framework is located in the MPCommandCenter in mPaaS, including two modules, namely Command Push and Dynamic Patching.

'Command Push': This module is in charge of synchronizing scripts with the server, and controlling the running version and timing of scripts.

'Dynamic Patching': This module executes the Lua scripts and completes the hotpatch by replacing the Objective-C method.

1.1.2 Working procedures

- The application is started, and Hotpatch executes scripts downloaded and stored to a local place.
- The application is started, and Hotpatch communicates with the server and downloads scripts. Once the download is complete, it executes the new scripts immediately.

- The application switches from the background to the foreground, and Hotpatch communicates with the server and synchronizes the scripts. Once the download is complete, it executes the new scripts at next application startup.

1.2 Publishing scripts

After the local verification is successful, the developer can deploy the scripts to be published on the server. The percentage and user ID for the push can be configured. (This feature is under development.)

1.2.1 Signature verification of scripts

Hotpatch has a self-defined signature verification procedure to ensure security and correct script sources. The signature verification procedure is as follows:

- Read the Lua script content, and encrypt the content character strings using MD5.
- Read the Lua script content in the binary format, and encrypt the binary contents using AES128 encryption.
- Create a zip file and include the above two content items into the zip file.
- Read the zip file above in the binary format, and encrypt the zip file again using AES128 encryption. The further encrypted data is saved as a zip file.
- Read the zip file above in the binary format, and sign the zip file using the SHA1 algorithm.
- Read the zip file above in the binary format, add the data delimiter, signature data and form the .js scripts.

The client requests the .js scripts, and gets the real scripts for execution by following the reversed order of the steps above.

1.2.2 How to use Lua2Js tool

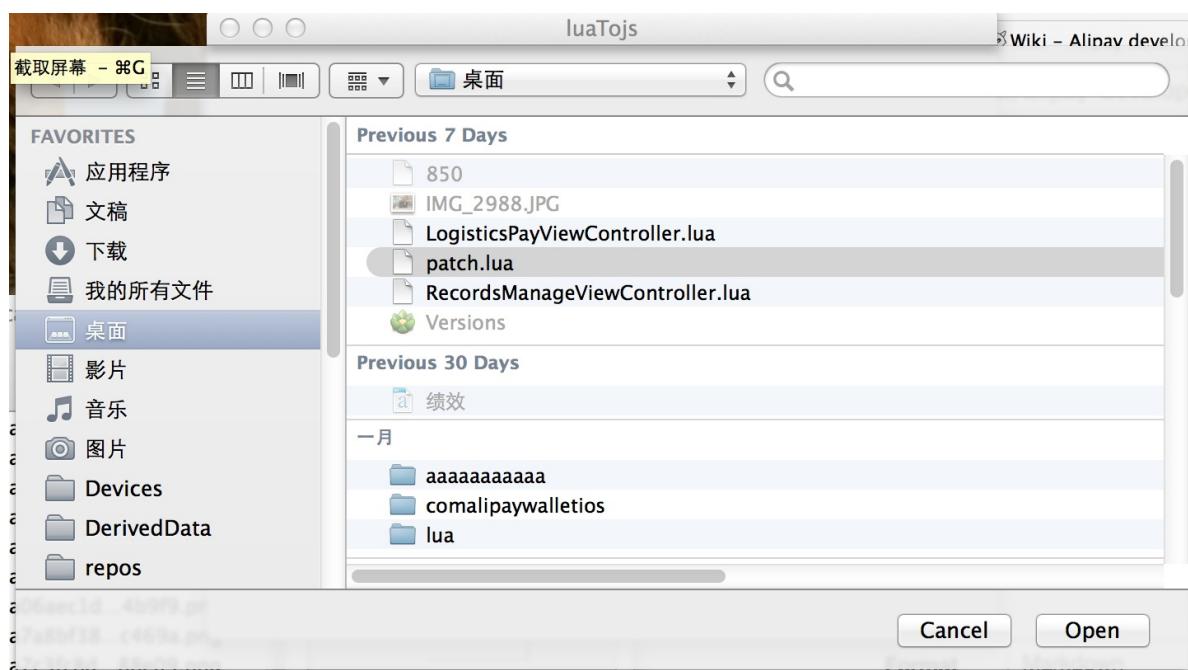
I. Generate your own Lua2Js tool

II. How to use

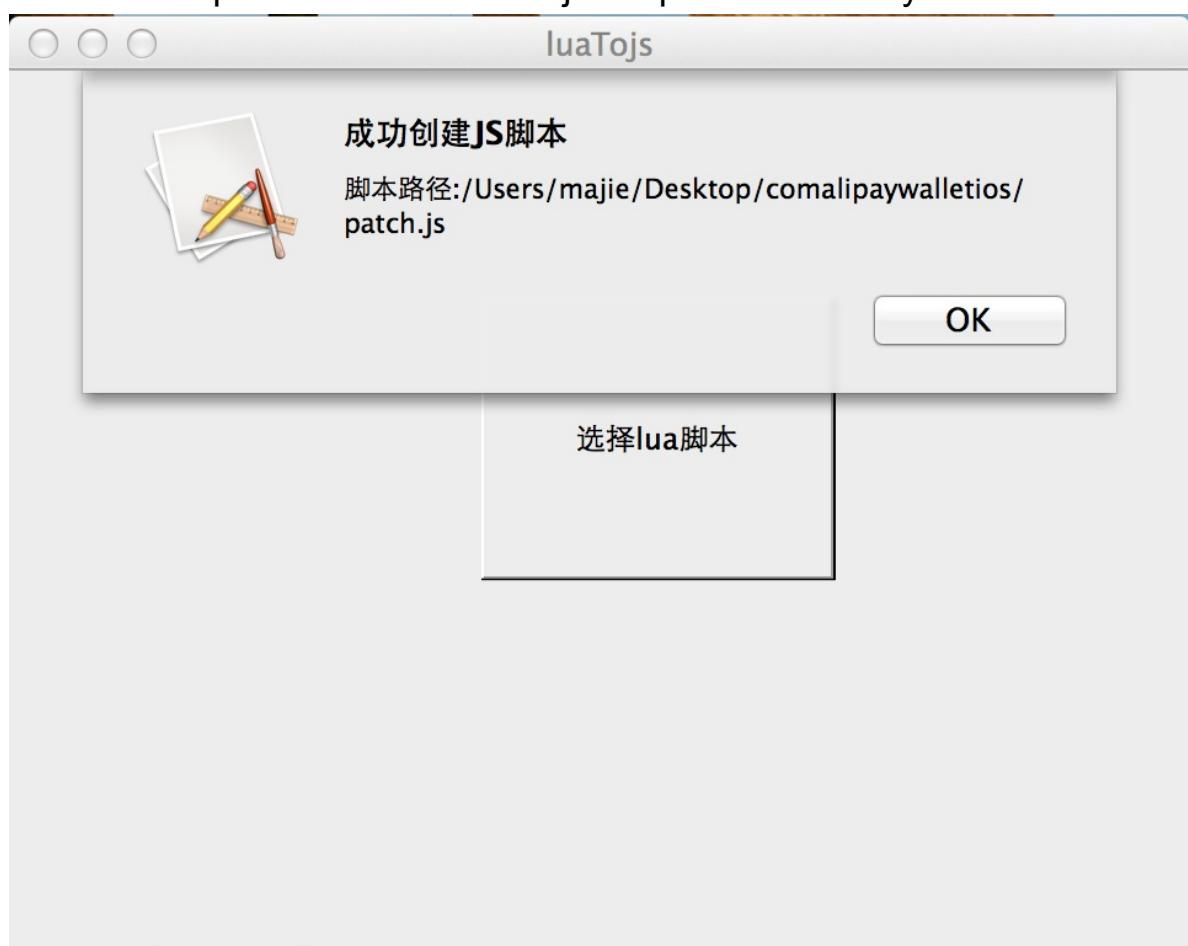
- Click the "Select Lua Scripts" button.



- Select the desired .lua scripts for conversion, and click the "Open" button.



- The .lua scripts are converted to .js scripts successfully.



1.3 Accessing Hotpatch on client

1.3.1 Managing key using symmetric encryption

Implement the interface under mPaasAppInterface to enable the Hotpatch module to call back the accessed application in order to acquire the key for the encryption in the Security Guard. (The application needs to configure the key and secret to the Security Guard for management.)

```
#pragma mark Hotpatch

/**
 *  The key name configured in Security Guard for encrypting the Hotpatch
 *
 *  @return The key name configured in Security Guard
 */
- (NSString*)appHotpatchScriptEncryptionKey;
```

1.3.2 Managing public key File using asymmetric encryption

Initialize your public key file by calling the method in APMobileSecurity library in the main function.

```
APSecBufferRef APSecGetPKS()
{
    char sig[] = {};

    size_t len = sizeof(sig);
    APSecBufferRef buf = (APSecBufferRef)malloc(sizeof(APSecBuffer) +
buf->length = len;
    memcpy(buf->data, sig, len);
    return buf;
}

int main(int argc, char *argv[])
{
    NSString *path;
    APSecBufferRef buf;
    int ret;

    path = [[NSBundle mainBundle] pathForResource:@"pubkey" ofType:@"";
APSecInitPublicKey([path UTF8String]);

    buf = APSecGetPKS();
    ret = APSecVerifyFile([path UTF8String], buf->data, buf->length);
    free(buf);
    if (ret != 0) {
        NSLog(@"The public key is modified.");
    }

    @autoreleasepool {
        mPaasInit(@"12345", [mPaasAppInterfaceImp class]);
        return UIApplicationMain(argc, argv, @"DFApplication", @"DFC1:");
    }
}
```

@cnName 2 Configuration Service @priority 2

2 Configuration Service

[TOC]

2.1 Introduction

The Configuration Service enables the client to pull arguments of dynamic configurations and modify client behaviors. It can be utilized for activity operation and access control.

The Configuration Service requires the corresponding RPC interface from the application background. When the client is started, it will invoke the RPC interface to acquire the switch configuration dictionary. The developer can configure the switch on the primary site. (The configuration service management on the primary site is still under development. It will be open to the public in the future.)

2.2 How to use

2.2.1 Obtaining Configuration Service instance

```
id<MPConfigService> configService = [mPaas() serviceWithName:@"Config"
```

It is recommended that the application encapsulate a macro or method to avoid mandatory type conversion every time. If the Configuration Service is not checked when the business accesses mPaas, nil will be returned.

2.2.2 Obtaining configuration

```
id<MPConfigService> configService = [mPaas() serviceWithName:@"Config";
NSString* aConfig = [configService stringValueForKey:@"AConfigName"];
```

2.2.3 Notification on successful pull of configuration updates

```
extern NSString* const MPConfigServiceDidFetchedFromServerNotification;
```

@cnName 3 Badge @priority 3

3 Badge

[TOC]

3.1 Instruduction

When the client needs to remind the user to pay attention to a certain UI element, it will display a badge on the element. The badges will accumulate with the levels of UI controls. mPaaS offers a set of solutions to the badge feature on the client.

The badge component of mPaaS has the following features:

- The 'BadgeView' widget used by the client on the interface has a unique identifier, i.e., the 'widgetId' property.
- The badge widget supports rich styles: red points, numbers, updates, etc. The developer can also customize the badge widget.
- The client uses 'BadgeManager' to manage the display, refreshing and deletion of badge widgets in a unified way.
- The client registers the badge widget in the 'BadgeManager' at interface initialization, and unregisters the badge widget in the 'BadgeManager' to destroy the badge widget.
- When the 'BadgeManager' receives display data, it will refresh the corresponding registered badge widget automatically, saving the developer's efforts to write additional codes for controlling the UI elements.
- The badge components have a parent-child relationship between each other. The number of badges on the parent node is the sum of all the badges on the child nodes. When all the badges on the child nodes disappear, the badges on the parent node will not show, either.

3.2 Definition of interface class

3.2.1 MPBadgeManager

MPBadgeManager manages badges, including registration, unregistration, refreshing and data of badge widgets.

I. Interface of badge manager

```
/**  
 * Set the configuration object of badge management.  
 *  
 * @param config Configuration object.  
 *  
 * @return Null  
 */  
- (void)setBadgeServiceConfig:(id<MPBadgeServiceConfig>)config;  
  
/**  
 * After user is logged in, this interface is called to process the d:  
 *  
 * @param userId ID of the user currently logged in.  
 *  
 * @return Null  
 */  
- (void)refreshAfterLogin:(NSString *)userId;  
  
/**  
 * Clear the badge data cached in memory and the display of all the ba:  
 *  
 * @param Null  
 *  
 * @return Null  
 */  
- (void)clearAllBadges;
```

II. Interface used by badge widget

Table of Contents |

```
/**  
 * Register badge widgets to the universal badge manager module.  
 *  
 * @param badgeView Badge widgets.  
 *  
 * @return Null  
 */  
- (void)registerBadgeView:(MPAbsBadgeView *)badgeView;  
  
/**  
 * Unregister the badge widget in the universal badge manager module.  
 *  
 * @param badgeView Badge widgets.  
 *  
 * @return Null  
 */  
- (void)unregisterBadgeView:(MPAbsBadgeView *)badgeView;  
  
/**  
 * Clear the badge if it is a leaf badge. Otherwise, leave it as is.  
 *  
 * @param badgeView Badge widgets.  
 *  
 * @return Null  
 */  
- (void)tapBadgeView:(MPAbsBadgeView *)badgeView;  
  
/**  
 * Click on the badge to trigger the display settings related to the badge.  
 *  
 * @param widgetId Badge widget ID.  
 *  
 * @return Null  
 */  
- (void)tapBadgeViewWithWidgetId:(NSString *)widgetId;
```

III. Interface for badge information acquisition

```
/**  
 * Acquire the badge information according to the specified badge path.  
 *  
 * @param badgeId Badge path ID, such as: "#M_PUBLIC_RD#2013112300021".  
 *  
 * @return Return badge information for successful acquisition. Otherwise, return null.  
 */  
- (MPBadgeInfo *)badgeInfoWithBadgeId:(NSString *)badgeId;  
  
/**  
 * Get the total badge count according to the specified business ID.  
 *  
 * @param bizId Business ID.  
 *  
 * @return The total badge count related to the business.  
 */  
- (NSUInteger)badgeCountWithBizId:(NSString *)bizId;  
  
/**  
 * Get the total badge count according to the specified badge widget ID.  
 *  
 * @param widgetId Badge widget ID.  
 *  
 * @return The total badge count related to the specified widget ID.  
 */  
- (NSUInteger)badgeCountWithWidgetId:(NSString *)widgetId;
```

IV. Interface for badge data injection

```
/**  
 * Full mode: overwrite the badge information returned from the server.  
 *  
 * @param badgeList MPBadgeInfo Array  
 *  
 * @return Null  
 */  
- (void)updateRemoteBadgeInfo:(NSArray *)badgeList;  
  
/**  
 * Incremental mode: insert the badge information returned from the server.  
 *  
 * @param badgeList MPBadgeInfo Array  
 *  
 * @return Null  
 */  
- (void)insertRemoteBadgeInfo:(NSArray *)badgeList;  
  
/**  
 * Incremental mode: insert the locale badge information.  
 * Attention: 1. No local cache; 2. Clicks on the badges are not reported.  
 *  
 * @param badgeList Badge data. The element is MPBadgeInfo.  
 *  
 * @return Null  
 */  
- (void)insertLocalBadgeInfo:(NSArray *)badgeList;
```

3.2.2 MPBadgeInfo

The badge data that defines badge styles and parent-child relationships.

```

@property(nonatomic, readonly) NSString *badgeId; // Badge path ID
@property(nonatomic, copy) NSString *style; // Widget display
@property(nonatomic, assign) NSUInteger temporaryBadgeNumber; //
@property(nonatomic, assign) NSUInteger persistentBadgeNumber; //

/**
 * Create badge data.
 *
 * @param badgeId Badge path ID.
 * Attention: When a parent-child relationship exists, separate
 *
 * @return Badge data.
 */
+ (MPBadgeInfo *)badgeInfoWithBadgeId:(NSString *)badgeId;

/**
 * Acquire all the badge widget ID on the badge path.
 *
 * @param Null
 *
 * @return The badge widget ID.
 */
- (NSArray *)allWidgetIds;

```

3.2.3 MPAbsBadgeView

The abstract base class of badge widget.

```

@property(nonatomic, copy) NSString *widgetId; // Badge widget ID. Wh
@property(nonatomic, copy) void (^ callback)(); // Provides the busine
@property(nonatomic, strong) MPWidgetInfo *widgetInfo; // Badge widget
@property(nonatomic, weak) id<MPBadgeViewDelegate> delegate; // Refre

/**
 * Calculation of numbers.

```

```
* Attention: 1. This property is not required for leaf nodes.  
*           2. When the badge widget ID is used in UITableViewCells, attention  
*  
* @"point" Calculate the "point" count only.  
* @"new"   Calculate the "new" count only.  
* @"num"   Calculate the "number" count only.  
* nil      Default calculation method, including the counts of all.  
*/  
@property(nonatomic, copy) NSString *numberCalculateMode;  
  
/**  
* Badge style for updates.  
*  
* @param badgeValue:    @"."    Display the red point.  
*                      @"new"  Display "new"  
*                      @"数字" Display the number. For numbers greater  
*                      @"惠"   Display the "惠" character  
*                      nil     Clear current badge display  
*  
* @return Null  
*/  
- (void)updateBadgeValue:(NSString *)badgeValue;  
  
/**  
* Draw badge styles.  
*  
* @param style:        @"."    Display the red point.  
*                      @"new"  Display "new"  
*                      @"数字" Display the number. For numbers greater  
*                      @"惠"   Display the "惠" character  
*                      nil     Clear current badge display  
*  
* @return Null  
*/  
- (void)drawBadgeStyle:(NSString *)style;
```

3.2.4 MPBadgeView

A badge widget and a subclass of MPAbsBadgeView.

3.2.5 MPBadgeServiceConfig

The interface for badge manager configuration.

```
/**  
 * Report clicks.  
 *  
 * @param widgetIds Report data.  
 * @param completion Callback after reporting completion.  
 *  
 * @return Null  
 */  
- (void)ack:(NSArray *)widgetIds completion:(void (^)(BOOL success))com  
  
/**  
 * Encrypt character strings.  
 *  
 * @param string Character strings to be encrypted.  
 *  
 * @return Encrypted character strings.  
 */  
- (NSString *)encryptString:(NSString *)string;
```

@cnName 4 User Feedback @priority 4

4 User Feedback

[TOC]

4.1 Introduction

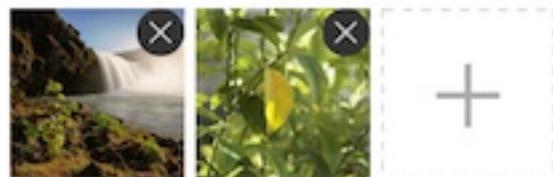
mPaaS integrates a simple but useful interface for user feedbacks with the image uploading feature. It also opens the interface for user feedbacks and access parties can rewrite their own UI interfaces.



Test

反馈

提交



+

236



4.2 How to use

MobileFramework > MobileFramework > Supporting Files > MobileRuntime.plist		
Key	Type	Value
Root	Dictionary (3 items)	
Applications	Array (18 items)	
Item 0	Dictionary (3 items)	
delegate	String	MPFeedbackAppDelegate
description	String	用户反馈
name	String	20000018
Item 1	Dictionary (3 items)	
Item 2	Dictionary (3 items)	

The micro application name can be customized by the access party, as long as it is a unique sub-application ID. The user feedback interface can be evoked with the following codes:

```
[DTContextGet() startApplication:@"20000018" params:@{@"userId":@"111":}
```

The 'userId' and 'mobileNo' arguments are optional. They are the information of the user currently logged in on the third-party application, for the convenience of contacting the user after the user's feedback is received.

@cnName 5 File Upload and Download @priority 5

5 File Upload and Download

[TOC]

5.1 Introduction

mPaaS offers file upload and download features to achieve uploading and downloading of the static resources. It also enables breakpoint resume and multi-part upload if the server supports such features. (At present, breakpoint resume and multi-part upload features are not open yet.)

5.2 Upload

5.2.1 Starting an upload task

```
NSString* fullPathName = @"...";
NSString* uploadUrl = @"...";
APFileTransferCenter* tcenter = [APFileTransferCenter sharedInstance];
NSInteger taskTag = [tcenter addUpTask:uploadUrl file:fullPathName de:
if(taskTag > 0)
{
    NSLog(@"UPLOAD[%d] start success", (int)taskTag);
}
else
{
    NSLog(@"UPLOAD[%d] start fail", (int)taskTag);
}
```

- Capable of creating tasks in any thread
- The fullPathName argument must be a file name with the full path.

- The uploadUrl argument is a full URL with HTTP or HTTPS.
- The delegate argument receives the notifications about the upload progress, success or failure (The notification is sent by the main thread).
- The returned value is exclusively used to mark the status of an upload task. When the value is greater than 0, it indicates upload success. Otherwise, it indicates upload failure.

5.2.2 APFileUpTaskDelegate

```
@protocol APFileUpTaskDelegate <NSObject>
- (void)APFileUpTaskDidSuccessed:(APFileUpTaskInfo*)info;
- (void)APFileUpTaskDidFailed:(APFileUpTaskInfo*)info;
- (void)APFileUpTaskDidProgressUpdated:(APFileUpTaskInfo*)info;
@end
```

5.3 Download

5.3.1 Starting a download task

```
NSString* fullPathName = @"...";
NSString* downloadUrl = @"...";
APFileTransferCenter* tcenter = [APFileTransferCenter sharedInstance];
NSInteger taskTag = [tcenter addDownTask:downloadUrl file:fullPathName];
if(taskTag > 0)
{
    NSLog(@"DOWNLOAD[%d] start success", (int)taskTag);
}
else
{
    NSLog(@"DOWNLOAD[%d] start fail", (int)taskTag);
}
```

- Create tasks capable of calls in any thread
- The fullPathName argument must be a file name with the full path.
- The downloadUrl argument is a full URL with HTTP or HTTPS.
- The delegate argument receives the notifications about the download progress, success or failure (The notification is sent by the main thread).
- The returned value is exclusively used to mark the status of a download task. When the value is greater than 0, it indicates download success. Otherwise, it indicates download failure.

5.3.2 APFileDownTaskDelegate

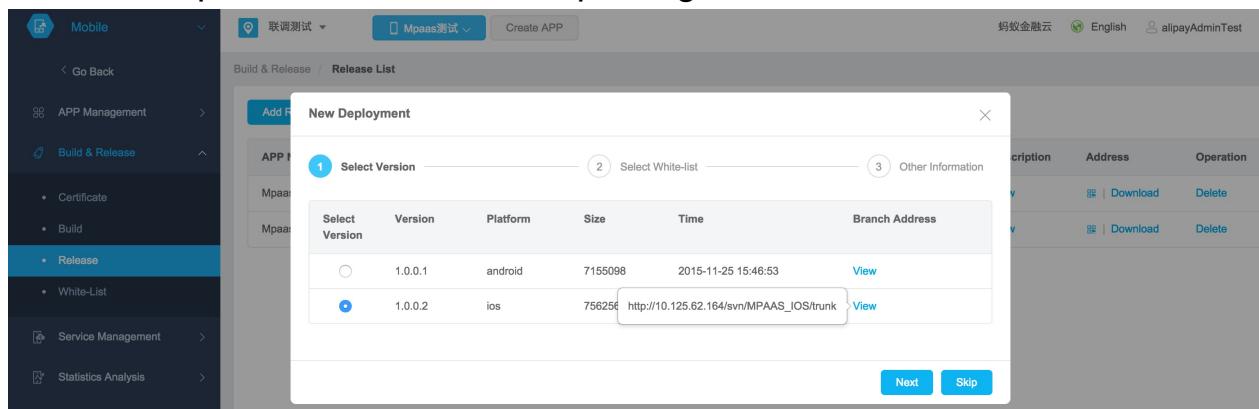
```
@protocol APFileDownTaskDelegate <NSObject>
- (void)APFileDownTaskDidSuccessed:(APFileDownTaskInfo*)info;
- (void)APFileDownTaskDidFailed:(APFileDownTaskInfo*)info;
- (void)APFileDownTaskDidProgressUpdated:(APFileDownTaskInfo*)info;
@end
```

@cnName 6 Checking for Updates @priority 6

6 Checking for Updates

6.1 Introduction

The mPaaS website offers the online packaging feature. Once the packaging is complete, the developer can choose the package of a specific version for release. The online application will detect the new version through the update interface and prompts users to download the update. The developer can also set update prompts, whether to enforce updates and other options for the released package.



'Note: App Store regulations prohibit built-in checking for updates in deployed applications. The checking for updates feature can be used during the development phase for internal testing.'

6.2 How to use

The feature provides the RPC interface file for checking for updates which can be integrated into the application codes. The application can call the interface on its own. Sample:

Table of Contents |

```
- (void)checkUpdate:(void (^)(MPAAS_CheckUpgradeResp* resp))callback
{
    __block MPAAS_CheckUpgradeResp *result = nil;
    [DTRpcAsyncCaller callAsyncBlock:^{
        MPAAS_CheckUpgradeServiceFacade* facade = [[MPAAS_CheckUpgradeServiceFacade alloc] init];
        MPAAS_CheckUpgradeReq* request = [[MPAAS_CheckUpgradeReq alloc] init];
        request.appkey = [mPaas() appKey];
        request.version = [[NSBundle mainBundle] objectForInfoDictionaryKey:@"CFBundleVersion"];
        request.systemPlatform = @"ios";
        @try
        {
            result = [facade checkUpdate:request];
        }
        @catch(DTRpcException *exception)
        {
            NSLog(@"%@", exception);
        }
    } completion:^{
        if (result)
        {
            callback(result);
        }
    }];
}
```

'Protocol File for Checking for Updates and Calling Codes'

@cnName 1 Overview @priority 1

1 Overview

1.1 Objective of using Data Center

- To reduce the usage of NSUserDefaults. Data of bigger sizes and private data are not stored in NSUserDefaults, which greatly enhances the access efficiency than using the NSUserDefaults.
- To reduce automatic maintenance of files by the business, and reduce the disorganized files in the Documents or Library directories.
- By storage space, Data Center can be divided into the user independent space and the storage space of the current user. The business layer does not need to care about the switches of users, and does not need to access the current user data with userId.
- Based on SQLite, Data Center supports DAO which is more flexible than CoreData. By encapsulating the database operations with configuration files and separating databases with the business, the business layer accesses data and operates on the database tables through interfaces.
- The underlying layer provides data encryption support.
- It offers versatile storage means to meet diversified demands and provides highly efficient memory cache service.

1.2 Descriptions of public class of Data Center

Class Name	Function
APDataCenter	Singleton class and the entry class of Data Center
APSharedPreferences	It corresponds to a database file and offers Key-Value storage interfaces. It stores DAO-created tables.
APDataCrypt	The struct of symmetric encryption
APLRUDiskCache	Disk cache that supports LRU knock-out rules
APLRUMemoryCache	Memory cache that supports LRU knock-out rules, thread-safe.
APObjectArrayService	Based on DAO, the class provides persistence function for NSCoding-supporting objects by different businesses. It supports encryption, capacity limit and memory cache.
APAsyncFileArrayService	Based on DAO, the class provides persistence function for binary data. It supports encryption, capacity limit and memory cache.
APCustomStorage	To customize the storage space in which the complete functions including user management, Key-Value and DAO storage are provided.
APDAOProtocol	Descriptions of interfaces supported by DAO objects.

@cnName 2.1 APDataCenter @priority 1

2.1 APDataCenter

[TOC]

2.1.1 Brief Introduction

APDataCenter is the entry class of Data Center. It is singleton and can be called anywhere in the codes.

```
[APDataCenter defaultDataCenter]
```

Macros can be used.

```
#define APDefaultDataCenter [APDataCenter defaultDataCenter]
```

It means to initialize APDataCenter.

2.1.2 Interface

Macro definition

```
#define APDefaultDataCenter [APDataCenter defaultDataCenter]
#define APCommonPreferences [APDefaultDataCenter commonPreferences]
#define APUserPreferences [APDefaultDataCenter userPreferences]
#define APCurrentVersionStorage [APDefaultDataCenter currentVersionSto
```

Constant definition

These event notifications usually require no attention from the business-level codes, but Data Center will throw these notifications.

```
/**  
 * Event notification on that the database file of the previous user  
 */  
extern NSString* const kAPDataCenterWillLastUserResign;  
  
/**  
 * Notification on that the user status has been switched. It is pos:  
 * The auxiliary object to this notification is a dictionary. If it :  
 */  
extern NSString* const kAPDataCenterDidUserUpdated;  
  
/**  
 * The user didn't switch, and APDataCenter receives the sign-in even:  
 */  
extern NSString* const kAPDataCenterDidUserRenew;
```

2.1.3 Interface and property

void APDataCenterLogSwitch(BOOL on);

Enable or disable the console log of Data Center. It is enabled by de-

@property (atomic, strong, readonly) NSString* currentUserId;

The userId of the user currently logged in.

+ (NSString*)preferencesRootPath;

Get the path where the commonPreferences and userPreferences database

- (void)setCurrentUserId:(NSString*)currentUserId;

Set the user ID of the user currently logged in. Do not call it in the

- (void)reset;

Fully reset the Data Center directories. Please use it with caution.

- (APSharedPreferences*)commonPreferences;

The user independent global storage database.

- (APSharedPreferences*)userPreferences;

The storage database of the user currently logged in. When it is not :

**- (APSharedPreferences)preferencesForUser:
(NSString)userId;**

Return the storage object of the specified user ID. The business layer

**- (APPreferencesAccessor)accessorForBusiness:
(NSString)business;**

```
Generate an data accessor based on the business name. The business layer will return the appropriate accessor for the business name.
```

```
APPreferencesAccessor* accessor = [[APDataCenter defaultDataCenter] accessorForName:@"BusinessName"];
[[accessor commonPreferences] doubleForKey:@"aKey"];
```

```
// Equal to
```

```
[[[[APDataCenter defaultDataCenter] commonPreferences] doubleForKey:@"aKey"]];
```

- (APCustomStorage*)currentVersionStorage;

Data Center will maintain a database of the current version. When the application starts up, it will automatically update the database.

- (id<APDAOProtocol>)daoWithPath:(NSString*)*filePath* userDependent:(BOOL)*userDependent*;

Generate a DAO access object from a configuration file.

```
@param filePath DAO The path of the configuration file. For files in the mainBundle, use [[NSBundle mainBundle] pathForResource:@"file" ofType:@"ext"]
```

```
NSString* filePath = [[NSBundle mainBundle] pathForResource:@"file" ofType:@"ext"];
```

```
@param userDependent Specifies the database operated by the DAO object.
```

```
@return The DAO object. The business layer does not need to care about the DAO object.
```

- (id<APDAOProtocol>)daoWithPath:(NSString)*filePath* databasePath:(NSString)*databasePath*;

Create a DAO access object maintaining its own independent database file.
The DAO object created using daoWithPath:userDependent: interface opens
This interface will create a DAO object operating on the database file.
Multiple DAO objects can be created pointing to the same databasePath.

`@param filePath` The same as daoWithPath:userDependent: interface.
`@param databasePath` The location of the DAO database file. Absolute path
`@return DAO object.`

@cnName 2.2 Key-Value storage @priority 2

2.2 Key-Value storage

[TOC]

2.2.1 Brief Introduction

For many scenarios on the client, the Key-Value storage is sufficient for use. Usually we will use NSUserDefaults, but NSUserDefaults does not support encryption and is slow for persistence data.

The Key-Value storage of Data Center provides interfaces for storing: PList objects such as NSInteger, long long (the same as NSInteger in 64-bit environment), BOOL, double and NSString, objects supporting NSCoder, Objective-C objects that can be converted to JSON objects through reflection, greatly reducing the complexity of persistent objects on the client.

For descriptions on most of the interfaces of Key-Value storage, see the method description in the APSharedPreferences.h header file.

2.2.2 Storing basic type

Data Center provides the following interfaces for storing the primary types of objects:

```
- (NSInteger)integerForKey:(NSString*)key business:(NSString*)business;
- (NSInteger)integerForKey:(NSString*)key business:(NSString*)business defaultValue:(NSInteger)defaultValue;
- (void)setInteger:(NSInteger)value forKey:(NSString*)key business:(NSString*)business;

- (long long)longLongForKey:(NSString*)key business:(NSString*)business;
- (long long)longLongForKey:(NSString*)key business:(NSString*)business defaultValue:(long long)defaultValue;
- (void)setLongLong:(long long)value forKey:(NSString*)key business:(NSString*)business;

- (BOOL)boolForKey:(NSString*)key business:(NSString*)business;
- (BOOL)boolForKey:(NSString*)key business:(NSString*)business defaultValue:(BOOL)defaultValue;
- (void)setBool:(BOOL)value forKey:(NSString*)key business:(NSString*)business;

- (double)doubleForKey:(NSString*)key business:(NSString*)business;
- (double)doubleForKey:(NSString*)key business:(NSString*)business defaultValue:(double)defaultValue;
- (void)setDouble:(double)value forKey:(NSString*)key business:(NSString*)business;
```

The 'defaultValue' argument is the default value returned when the data does not exist.

2.2.3 Storing Objective-C object

2.2.3.1 Interface description

Data Center provides the following interfaces for storing Objective-C objects:

```
- (NSString*)stringForKey:(NSString*)key business:(NSString*)business;
- (NSString*)stringForKey:(NSString*)key business:(NSString*)business extens:
- (void)setString:(NSString*)string forKey:(NSString*)key business:(NS
- (void)setString:(NSString*)string forKey:(NSString*)key business:(NS
- (id)objectForKey:(NSString*)key business:(NSString*)business;
- (id)objectForKey:(NSString*)key business:(NSString*)business extens:
- (void)setObject:(id)object forKey:(NSString*)key business:(NSString*
- (void)setObject:(id)object forKey:(NSString*)key business:(NSString*
- (void)archiveObject:(id)object forKey:(NSString*)key business:(NSSt
- (void)archiveObject:(id)object forKey:(NSString*)key business:(NSSt
- (void)saveJsonObject:(id)object forKey:(NSString*)key business:(NSS
- (void)saveJsonObject:(id)object forKey:(NSString*)key business:(NSS
```

setString & stringForKey

The setString and stringForKey interfaces are recommended for storing NSString objects as the names are more interpretative.

If the data is not encrypted, strings stored with this interface can be viewed in Sqlite DB viewer, being more intuitive. Strings stored in the setObject method will be first converted to NSData through Property List and then saved to the database.

setObject

The setObject method is recommended for storing Property List objects to achieve the highest efficiency.

'Property List objects': NSNumber, NSString, NSData, NSDate, NSArray, and NSDictionary. The subobjects in NSArray and NSDictionary must also be PList objects.

If the `setObject` method is used for storing Property List objects, the object acquired using the `objectForKey` method is mutable. The `savedArray` acquired in the codes below is of the `NSMutableArray` class.

```
NSArray* array = [[NSArray alloc] initWithObjects:@"str", nil];
[APCommonPreferences setObject:array forKey:@"array" business:@"biz"].

NSArray* savedArray = [APCommonPreferences objectForKey:@"array" busin
```

archiveObject

For Objective-C objects supporting the `NSCoding` protocol, Data Center calls the system `NSKeyedArchiver` and converts the object into an `NSData` object for persistence storage.

Property List objects can use this interface, too, but with a low efficiency and thus not recommended.

saveJsonObject

When an Objective-C object is neither a Property List object nor an `NSCoding`-supporting one, this method can be utilized for persistence storage.

Through runtime dynamic reflection, this method maps Objective-C objects to JSON strings. But not all the Objective-C objects can be saved with this method, such as properties containing C structure pointers, Objective-C objects that reference each other, or objects containing dictionaries or arrays in properties.

objectForKey

When Data Center saves the data of Objective-C objects, it will record the archiving methods as well. The `objectForKey` method is used for acquiring objects. Attention: Strings saved using the `setString` method should be acquired with the `stringForKey` method.

2.2.3.2 Data encryption

Using default encryption

Interfaces with extension arguments support encryption, and APDataCrypt struct is passed.

'APDefaultEncrypt' is the default encryption method using AES symmetric encryption.

'APDefaultDecrypt' is the default decryption method and share the key with 'APDefaultEncrypt'.

In usual cases, the default encryption provided by Data Center is used, as follows:

```
[APUserPreferences setObject:aObject forKey:@"key" business:@"biz" ext:  
  
id obj = [APUserPreferences objectForKey:@"key" business:@"biz" ext:  
// or  
id obj = [APUserPreferences objectForKey:@"key" business:@"biz"];
```

As the default encryption is adopted, the extension argument can be skipped for the interfaces acquiring data.

Using self-defined encryption

If the business has higher security requirements for encryption, the APDataCrypt struct can be implemented with specified function pointers for encryption and decryption. But please ensure that the encryption and decryption methods are matched to save and recover data correctly.

Encrypt primary types

To encrypt BOOL, NSInteger, double, and long long objects for storage, you can convert them into strings or put them into the NSNumber class, and then call the `setString` or `setObject` interface.

@cnName 2.3.1 Overview @priority 1

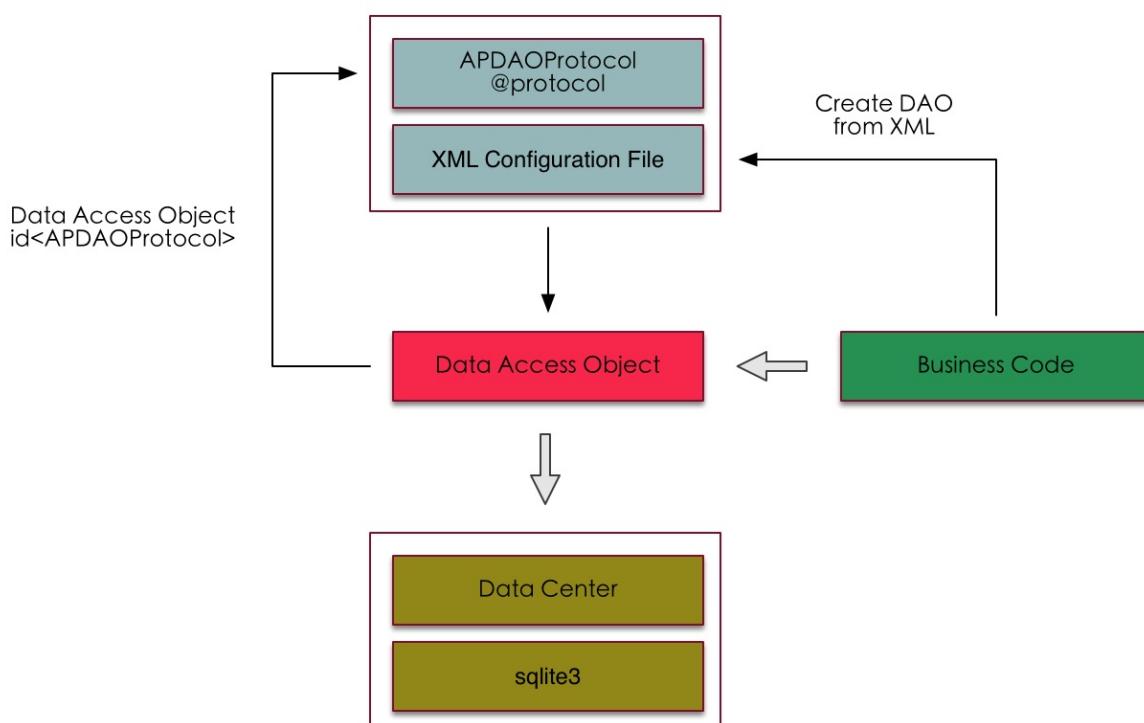
2.3.1 Overview

[TOC]

When to use

General KV storage can only store simple data types, or encapsulated OC objects and does not support search. When the business requires the access to SQLite, the DAO function of Data Center can be utilized for simplification and encapsulation.

Basic working principle



- Define an XML-format configuration file to describe the functions, returned data types and encrypted fields of various SQLite operations.
- Define an interface for DAO objects, DAOInterface (@protocol). The interface method names and arguments should stay consistent with those described in the configuration file.

- The business layer passes the XML configuration file to the daoWithPath method of APDataCenter, and the DAO access object is generated. This object is enforced to be converted to id.
- Next, the business layer will be able to directly call the DAO object methods, and Data Center will translate the method into the database operations described in the configuration file.

A simple example

- The first line of the configuration file defines the default table name and database version, as well as the initialization method.
- The insertItem and getItem are two methods for inserting and reading data. They receive arguments and format the arguments into SQL expressions.
- The createTable method will be called once on the underlying layer by default.

```

<module name="Demo" initializer="createTable" tableName="demoTable">
    <update id="createTable">
        create table if not exists ${T} (index integer primary key, con-
    </update>

    <insert id="insertItem" arguments="content">
        insert into ${T} (content) values(${content})
    </insert>

    <select id="getItem" arguments="index" result="string">
        select * from ${T} where index = ${index}
    </select>
</module>

```

- Definition of DAO interfaces.

```
@protocol DemoProtocol <APDAOProtocol>
- (APDAOResult*)insertItem:(NSString*)content;
- (NSString*)getItem:(NSNumber*)index;
@end
```

- Create a DAO proxy object. Suppose the configuration file is named demo_config.xml and is located in Main Bundle.
- Write a piece of data with the insertItem method, acquire its index, and then read the written data with the index.

```
NSString* filePath = [[NSBundle mainBundle] pathForResource:@"demo"
id<DemoProtocol> proxy = [[APDataCenter defaultCenter] daoWithFile:filePath];
[proxy insertItem:@"something"];
long long lastIndex = [proxy lastInsertRowId];
NSString* content = [proxy getItem:[NSNumber numberWithInt:lastIndex]];
NSLog(@"%@", content);
```

- The lastInsertRowId is a method of APDAOProtocol, used for getting the rowId of the row last inserted. To enable the DAO object to support this method, you only need to make the DemoProtocol inherit from the APDAOProtocol in declaration.

@cnName 2.3.2 Keywords @priority 2

2.3.2 Keywords

[TOC]

module

```
<module name="demo" initializer="createTable" tableName="tableDemo" v>
```

- The initializer parameter is optional. For the update method specified by the initializer, DAO regards it a method for creating database tables and will execute it once by default at the first DAO request.
- The tableName specifies the table name for the default operation in the method below. It can be replaced by \${T} or \${t} in SQL statements, so that you don't have to input the table name every time. It is recommended that one configuration file be targeted to one table. The tableName can be empty, so that the same configuration file can operate on multiple tables sharing the same format. For example, to process chat messages in tables, the setTableName method of DAO objects can be called for setting the name of the table to be operated on.
- The version represents the version number of the configuration file, in 'x.x' format. After the table is created, the tableName will serve as the key and the table version will be saved to the TableVersions table in the database file. TableVersions will work in concert with the upgrade block for table updates.
- resetOnUpgrade. If its value is true or YES, when the version is updated, the old table will be deleted instead of calling the upgrade block. If this argument does not exist, its value is false by default.
- upgradeMinVersion. If it is not empty, database files of a version lower than its value will be reset directly. Otherwise the upgrade operation will

be performed.

const

```
<const table_columns="(id, time, content, uin, read)" />
```

- It defines a constant of the string type. The table_columns is the name of a constant, and the content after the equal mark is the constant value. The constant can be referenced in the configuration file with \${constant name}.

select

```
<select id="find" arguments="id, time" result=":messageModelMap">
    select * from ${T} where id = #{id} and time > @{time}
</select>
```

@arguments

- List of the argument names, separated by ','. Incoming arguments from the caller are named in turn according to the descriptions in arguments. Selectors of DAO objects will not carry the argument name in calling, so arguments must be named in sequence here.
- If an argument has a \$ symbol at its beginning, it indicates that this argument does not accept the nil value. Calls of DAO interfaces from the business layer allow nil arguments. But if an argument has a \$ symbol at its beginning and the caller accidentally passes a nil value, the DAO call will fail automatically to prevent unexpected issues from happening.
- For ways to reference arguments, see [DAO Reference](#).

In the codes above, the corresponding selector is:

```
- (MessageModel*)find:(NSNumber*)id time:(NSNumber*)time;
```

If the DAO object calls [daoProxy find:@1234 time:@2014], the ready SQL statement will be:

```
select * from tableDemo where id = ? and time > 2014
```

and the NSNumber value @1234 will be handed over to SQLite for binding.

@result

- The result can be the return value of the DAO method. The use of square brackets [] indicates to return the array type of value and iterations occur for the returned values of the SELECT method until no result is returned from SQLite. If the square brackets [] are removed, it indicates to return one result only and one iteration is conducted for the SELECT method. It is similar to the next method for FMResultSet in FMDB database.

Return types:

- int: only one result, of the [NSNumber numberWithInt] type.
- long long: only one result, of the [NSNumber numberWithLongLong] type.
- bool: only one result, of the [NSNumber numberWithBool] type.
- double: only one result, of the [NSNumber numberWithDouble] type.
- string: only one result, of the NSString* type.
- binary: only one result, of the NSData* type.
- [int]: an array, with values of the [NSNumber numberWithInt] type in the array.
- [long long]: an array, with values of the [NSNumber numberWithLongLong] type in the array.
- [bool]: an array, with values of the [NSNumber numberWithBool] type in the array.

- [double]: an array, with values of the [NSNumber numberWithDouble] type in the array.
- [string]: an array, with values of the NSString* type in the array.
- [binary]: an array, with values of the NSData* type in the array.
- [{}]: an array, with mapping of column name->column value in the array.
- [AType]: an array, with filled self-defined classes in the array.
- {}: only one result, mapping of column name->column value.
- AType: only one result, with the filled self-defined class.
- [:AMap]: an array, with XML-defined mapped objects of AMap in the array.
- :AMap: only one result. The AMap defined in the configuration file is used to describe the object.

In the example above, the returned type is ":messageModelMap". The Objective-C types returned and columns that require special mapping will be defined in messageModelMap. Refer to the keyword "map".

@foreach The SELECT method also supports "foreach" field and its usage is similar to that of INSERT, UPDATE and DELETE methods to be introduced later. The difference is, if the SELECT method specifies the foreach argument, the SELECT operation will be executed for N times, and the results will be returned in an array. So if the SELECT method in DAO specifies the foreach argument, its return value must be defined as "NSArray**" in the protocol.

insert, update, delete

```

<insert id="addMessages" arguments="messages" foreach="messages.model">
    insert or replace into ${T} (id, content, uin, read) values(${model})
</insert>

<update id="createTable">
    <step>
        create table if not exists ${T} (id integer primary key, content + 
    </step>
    <step>
        create index if not exists uin_idx on ${T} (uin)
    </step>
</update>

<delete id="remove" arguments="msgId">
    delete from ${T} where msgId = #{msgId}
</delete>

```

- The INSERT, UPDATE and DELETE methods share the same format and their arguments concatenation and references are the same with that of the SELECT method.
- The INSERT and DELETE keywords serve to differentiate purposes of methods. You can write a "DELETE FROM TABLE" operation in the "UPDATE" function.
- In DAO interfaces, when the INSERT, UPDATE or DELETE methods are executed, the returned value is APDAOResult*, indicating whether the execution succeeded. See [DAO A Simple Example](#)

@foreach

- When a 'foreach' field follows the INSERT, UPDATE or DELETE methods, the method will query every element in the argument array one by one when the method is called.
- The 'foreach' field must follow the format below: collectionName.item. The “collectionName” must correspond to an argument in "arguments"

and specify the loop container object. It must be of the NSArray or NSSet type when called. The “item” represents the object fetched from the container and will be used as the loop variable. The item name cannot be duplicated with that of any argument in the "arguments". The item can be used as a normal argument in the SQL statement.

For example, the delegate method is:

```
- (void)addMessages:(NSArray*)messages;
```

The “messages” is a MessageModel array. For every model in the messages, an SQL call will be executed so that elements in the array will be inserted to the database in one shot while the upper layer does not need to care about the loop call. The underlying layer will merge the operation into one transaction to improve efficiency.

@step

- In INSERT, UPDATE and DELETE methods, you may run into such a circumstance: To execute a DAO method, the SQL update operation is called for multiple times to execute multiple SQL statements. For example, after a table is created, the user wants to create some indexes. The statement between in the function will be executed independently as one SQL update operation, and the underlying layer will merge all the operations into one transaction, such as the createTable method in the figure above.
- If the step clause exists in a function, no texts are allowed outside the step. The step cannot contain another step.

map

```
<map id="messageModelMap" result="MessageModel">
  <result property="msgId" column="id"/>
</map>
```

- It defines a mapping named messageModelMap. The actual Objective-C object generated is of the MessageModel class.
- The msgId property of the Objective-C object maps the column value of Column id in the table. Properties not listed are regarded consistent with the column name in the table, thus omitted.

upgrade

```
<upgrade toVersion="1.1">
  <step>
    alter table...
  </step>
  <step>
    alter table...
  </step>
</upgrade>
```

- With the version updated, the database may have the demand for upgrade. The SQL statements for upgrade are written here. For example, at the very beginning, the version of the configuration file module is 1.0. After upgrade, the configuration file version is changed to 1.1. When the new version configuration file module runs DAO methods for the first time, it will check the current table version with the configuration file version. With an inconsistency found, it will execute the upgrade step by step. This method is called automatically by the underlying layer. The DAO method will be executed after the upgrade is done.
- The upgrade is executed according to the SQL UPDATE statement. If there are multiple statements, they can be enclosed by , which is similar to the implementation of .
- If the operations defined by the upgrade block are required for upgrading the table, the "resetOnUpgrade" argument in the module must be set to "false".

crypt

```
<crypt class="MessageModel" property="content"/>
```

- It describes that the property of the specified class will be encrypted. When data is written to the database, values fetched from this property will be encrypted. When data is read from the database, the generated object will first decrypt the value before assigning values to the property.

For example, in execution of this DAO method, model is of the MessageModel class. As the content property of model is fetched, the value will be encrypted before being written into the database.

```
<insert id="insertMessage" arguments="model">
    insert into ${T} (content) values(${model.content})
</insert>
```

The execution of this SELECT method will return an object of the MessageModel class. When the underlying layer fetches data from the database and writes the content property to the MessageModel instance, it will first decrypt the data before writing the data, and then return the ready MessageModel object.

```
<select id="getMessage" arguments="msgId" result="MessageModel">
    select * from ${T} where msgId = ${msgId}
</select>
```

The methods for setting encryption are defined in [APDAOProtocol](#), as follows:

```

/**
 * Set an encryptor for encrypting the data in columns marked for encryption.
 *
 * @param crypt The encryption struct which will be copied. If the implementation
 *              does not support copying, it must implement a copy constructor.
 */
- (void)setEncryptMethod:(APDataCrypt*)crypt;

/**
 * Set a decryptor for decrypting the data in columns marked for decryption.
 *
 * @param crypt The decryption struct which will be copied. If the implementation
 *              does not support copying, it must implement a copy constructor.
 */
- (void)setDecryptMethod:(APDataCrypt*)crypt;

```

If no setting is made, the default encryption of APDataCenter will apply. See [KVStore](#). If a DAO proxy object is id, and the DAOProtocol is @protocol, the DAO proxy object can be directly used for calling setEncryptMethod and setDecryptMethod for setting encryption and decryption methods.

if

```

<insert id="addMessages" arguments="messages, onlyRead" foreach="message">
    <if true="model.read or (not onlyRead)">
        insert or replace into ${T} (msgId, content, read) values(${model})
    </if>
</insert>

```

- The IF conditional statements can be nested in INSERT, UPFATE, DELETE and SELECT methods. When IF conditions are met, the texts in the IF block will be concatenated into the final SQL statement.
- The IF can be followed by true="expr" or false="expr". The "expr" stands for expression. It can utilize the argument of the method, and "." to list the argument object properties to call.

- Operators supported by the expression are as follows:

(): brackets

+: positive sign

-: negative sign

+: plus sign

-: minus sign

*: multiplication sign

/: division sign

\: exact division sign

%: modulus

>: greater than

<: less than

>=: greater than or equal to

<=: less than or equal to

==: equal to

!=: unequal to

and: AND, case-insensitive

or: OR, case-insensitive

not: NOT, case-insensitive

xor: exclusive OR, case-insensitive

- The greater-than sign and less-than sign must be escaped. See <http://lidongbest5.com/blog/5/>
- The arguments inside are names of the incoming arguments from

external calls, but do not enclose the arguments with #{} or @{} like in the case of the SQL block.

- The meaning of nil here is the same as the nil in Objective-C.
- Strings in the expression should be started or ended with single quotes. Escape characters are not supported, but "\'" is supported to represent a single quote.
- When the arguments are an Objective-C object, '.' is supported for accessing its property, such as the model.read in the example above. If the argument is an array or dictionary, 'argument name.count' can be used to get the element count.

Below is a more complex expression:

```
<if true="(title != nil and year > 2010) or authors.count >= 2">
```

The “title”, “year” and “authors” are all arguments passed from the caller. A nil value is allowed for “title” in the call. The expression above means “when the book title is not empty, and the year of publication is later than 2010, or there are more than 2 authors for the book”.

choose

```
<choose>
  <when true="title != nil">
    AND title like #{title}
  </when>
  <when true="author != nil and author.name != nil">
    AND author_name like #{author.name}
  </when>
  <otherwise>
    AND featured = 1
  </otherwise>
</choose>
```

- It implements similar syntax to SWITCH statements, and its expression requirements are similar to the IF statement. The WHEN keyword can also be followed by true="expr" or false="expr".
- Only the first eligible WHEN or OTHERWISE statement will be executed. The OTHERWISE argument can be void.

foreach

```
<foreach item="iterator" collection="list" open="(" separator="," close=")">
    @{iterator}
</foreach>
```

- The open, separator, close, and reverse arguments can be omitted.
- The “item” represents the loop variable, and “collection” represents the argument name of the loop array.

For example, a method receives string array arguments from the outside.

The list content is @[@"1", @"2", @"3"]. There is another argument, prefix=@"abc", enclosed by '()', and separated by ','. The execution result will be: (abc1,abc2,abc3).

```
<update id="proc" arguments="list, prefix">
    <foreach item="iterator" collection="list" open="(" separator="," close=")">
        {prefix}{iterator}
    </foreach>
</update>
```

Foreach statements are usually used for concatenating the “in” blocks in the SELECT statement, for example:

```

select * from ${T} where id in
<foreach item="id" collection="idList" open="(" separator="," close="")
#{id}
</foreach>

```

where, set, trim

```

<where onVoid="quit">
  <if true="state != nil">
    state = #{state}
  </if>
  <if true="title != nil">
    AND title like #{title}
  </if>
  <if true="author != nil and author.name != nil">
    AND author_name like #{author.name}
  </if>
</where>

```

The WHERE condition will handle superfluous AND, OR (case insensitive) conditions, and it may not return anything when no condition is met, even the WHERE. It is used to concatenate WHERE clauses with a large number of conditions in SQL. As shown in the example above, if only the last judgment is tenable, the statement will correctly return "where author_name like XXX", instead of "where AND author_name like XXX".

```

<set>
  <if false="username != nil">username=#{username},</if>
  <if false="password != nil">password=#{password},</if>
  <if true="email != nil">email=#{email},</if>
  <if true="bio != nil">bio=#{bio},</if>
</set>

```

The SET keyword will handle the superfluous ',' signs in the end, and return nothing when no condition is met. It is similar to the WHERE statement, but only that it handles the suffixal commas.

```
<trim prefix="WHERE" prefixOverrides="AND | OR | and | or " onVoid="ignoreAfter">
</trim>
<!--
   Equivalent to <where>
-->

<trim prefix="SET" suffixOverrides="," onVoid="quit">
</trim>
<!--
   Equivalent to <set>
-->
```

- The WHERE and SET statements can be replaced by TRIM statements. TRIM statements define the overall prefix of the statement, and the list of superfluous prefixes and suffixes that each clause will handle (divided by "|").
- The onVoid argument can appear in WHERE, SET and TRIM statements. It has two values: "ignoreAfter" and "quit". The values imply the logics used when an empty string is generated because no clause in the TRIM statement is tenable. ignoreAfter means to ignore the following formatting statements and return the current SQL statement for execution, while "quit" means not to execute this SQL statement but to return success.

sql

```

<sql id="userColumns"> id,username,password </sql>
<select id="selectUsers" result="{}">
    select ${userColumns}
    from some_table
    where id = #{id}
</select>

```

It defines the reusable SQL code segments and uses \${name} to quote the source code segments from other statements. The "name" in \${name} cannot be 'T' or 't', because \${T} and \${t} represents the default table name. Inside the sql block other sql blocks can be further quoted.

try except

```

<insert id="insertTemplates" arguments="templates" foreach="templates">
    <try>
        insert into ${T} (tplId, tplVersion, time, data, tag) values(${templates})
    <except>
        update ${T} set %{'* = #{model.*}'}, data, tag} where tplId = #{model.tplId}
    </except>
    </try>
</insert>

```

Sometimes the same model may be inserted into the database for multiple times. In this case, the INSERT or REPLACE statements will lead to the loss of old data when conflicts occur in the model primary key (Another model of the same primary key already exists in the database), and the data is re-inserted. This will lead to changes in the rowid of the same piece of data. The TRY EXCEPT statement block can solve the problem, among others. TRY EXCEPT statements can only be used in definitions of DAO methods and should have no preceding or following statements. Other statement blocks can be included inside the TRY and EXCEPT statements.

During execution of this DAO method, if execution of the TRY statement fails, the execution will move to the EXCEPT statement automatically. Only when both of them fail will this DAO call return the failure result.

@cnName 2.3.3 Reference @priority 3

2.3.3 Reference

[TOC]

@ reference

@{something}: used for method arguments and the argument name is "something". To format SQL statements, all the object contents will be concatenated into SQL statements. As all the arguments are of the ID type, [NSString stirngWithFormat:@ "%@", id] is used by default for formatting. The @{something or ""} format represents that if the incoming argument is nil, it will be converted to an empty string instead of NULL.

We do not recommend @{} for referencing arguments as it is not efficient and subject to the SQL injection risk. If the argument object is of the NSString class, the string will be automatically enclosed in quotes after concatenation to ensure the correctness of the SQL statement format. But if the user has added quotes in the configuration file, the underlying layer will not add the quotes again.

If the @{something no ""} format is adopted, you can enforce not to add quotes.

```
<select id="getMessage" arguments="id" result="[MessageModel]">
    select * from ${T} where id = @{id}
</select>
```

In the example above, the incoming id argument is of the NSString class and the codes above are correct. The generated SQL will format and concatenate the id and enclose it with quotes.

reference

`#{something}`: used for method arguments and the argument name is "something". To format SQL statements, the argument will be converted to '?' and the object is bound to SQLite. This method is recommended for SQL coding as it is efficient. The `#{something or ""}` format represents that if the incoming argument is nil, it will be converted to an empty string instead of NULL.

\$ reference

`${something}`: used for referencing contents in the configuration file, such as the default table name `${T}` or `${t}`, and constants and SQL statement blocks defined in the configuration file.

Chain access

For @ reference and # reference, you can use '.' to access the property of argument objects. For example, the incoming argument name is "model" of the MessageModel type. It has a property of `NSString`*: "content". With `@{model.content}`, you will be able to fetch the value of its content property. The internal implementation is `[NSObject valueForKey:]`. So if the argument is a dictionary (The dictionary's `valueForKey` is equivalent to `dict[@""]`), you can also use `#{adict.aaa}` to reference the `adict[@""aaa"]` value.

@cnName 2.3.4 DAO Proxy @priority 4

2.3.4 DAO Proxy

Each proxy object of generated DAO objects supports APDAOProtocol.

E.g.,

```
@protocol MyDAOOperations <APDAOProtocol>
- (APDAOResult*)insertMessage:(MyMessageModel*)model;
@end
```

The specific method can be found in function annotations of the codes. With any special method required, submit a request to me.

```
#import <Foundation/Foundation.h>
#import <sqlite3.h>
#import "APDataCrypt.h"
#import "APDAOResult.h"
#import "APDAOTransaction.h"

@protocol APDAOProtocol;

typedef NS_ENUM (NSUInteger, APDAOProxyEventType)
{
    APDAOProxyEventShouldUpgrade = 0,      // Will be upgraded very soon
    APDAOProxyEventUpgradeFailed,           // Table upgrade failed.
    APDAOProxyEventTableCreated,           // Table is created.
    APDAOProxyEventTableDeleted,           // Table is deleted.
};

typedef void(^ APDAOProxyEventHandler)(id<APDAOProtocol> proxy, APDAOI
/** 
 * The method defined by this protocol is supported by all the DAP p
*/
```

Table of Contents |

```
@protocol APDAOProtocol <NSObject>

/***
 *  The module in the configuration file can set table names. If you want to use different tables for conversation message, you can set the table name by this protocol.
 */
@property (atomic, strong) NSString* tableName;

/***
 *  Return the path of the database file where the operated table by the proxy is stored.
 */
@property (atomic, strong, readonly) NSString* databasePath;

/***
 *  Acquire the handle of the database file operated table by the proxy.
 */
@property (atomic, assign, readonly) sqlite3* sqliteHandle;

/***
 *  Register the global variable arguments. All the methods in configuration file will call this method to register their arguments.
 */
@property (atomic, strong) NSDictionary* globalArguments;

/***
 *  The event callback of this proxy is set by the business. The callback function will be called when the event occurs.
 *  Pay attention to the circular reference. The business object holds the proxy object, so the proxy object holds the business object.
 */
@property (atomic, copy) APDAOProxyEventHandler proxyEventHandler;

/***
 *  Set an encryptor for encrypting the data in columns marked for encryption.
 *
 *  @param crypt The encryption struct which will be copied. If the implementation of APDataCrypt is not provided, it will be copied directly.
 */
@property (atomic, assign) APDataCrypt* encryptMethod;

/***
 *  Set a decryptor for decrypting the data in columns marked for encryption.
 */
@property (atomic, assign) APDataCrypt* decryptMethod;
```

```
*  
* @param crypt The decryption struct which will be copied. If the in  
*/  
@property (atomic, assign) APDataCrypt* decryptMethod;  
  
/**  
 * Return the rowId of the last row of SQLite.  
 *  
 * @return sqlite3_last_insert_rowid()  
*/  
- (long long)lastInsertRowId;  
  
/**  
 * Obtain the list of all the methods defined in the configuration file.  
 */  
- (NSArray*)allMethodsList;  
  
/**  
 * Delete the table defined in the configuration file. It can be used  
 */  
- (APDAOResult*)deleteTable;  
  
/**  
 * Delete all the tables conforming to a regular expression rule. Mal  
 *  
 * @param pattern Regular expression  
 * @param autovacuum After the deletion is complete, whether to call  
 * @param progress Progress callback. Nil value is allowed. The cal  
 *  
 * @return Whether the operation is successful.  
 */  
- (APDAOResult*)deleteTablesWithRegex:(NSString*)pattern autovacuum:(BOOL)  
  
/**  
 * Call the database link of your own to execute the “vacuum” operat  
 */  
- (void)vacuumDatabase;
```

Table of Contents |

```
/**  
 * DAO objects can put their operations in transactions to speed up +  
 */  
- (APDAOResult*)daoTransaction:(APDAOTransaction)transaction;  
  
/**  
 * Create a parallel connection for the database, for the purpose of  
 * The connections created will be automatically closed and the busi  
  
 * @param autoclose Automatically close the connections if they are :  
 */  
- (void)prepareParallelConnection:(NSTimeInterval)autoclose;  
  
@end
```

@cnName 2.3.5 Transaction @priority 5

2.3.5 Transaction

Each DAO call is a transaction, even if there are multiple SQL statements for execution in one DAO operation. E.g.,

```
<update id="createTable">
    <step>
        create table if not exists ${T} (localID long, clientMsgID char(36) primary key, content
    </step>
    <step>
        create index if not exists messageState_idx on ${T} (messageState)
    </step>
</update>
```

in this operation for creating a table (DAO operations for table creation will be automatically executed), there are two SQL calls divided by “step”: first, create a table; second, create indexes. The two statements constitute one transaction.

Here is another example. A foreach argument exists in INSERT operation.

```
<insert id="addMessages" arguments="messages" foreach="messages.model">
    insert or replace into ${T} (id, content, uin, read) values(#${model.id}, #${model.content}, #${model.uin}, #${model.read})
</insert>
```

When the business layer calls this DAO method, it will pass the message model list "messages", DAO will execute one SQL INSERT operation for each model (loop in the underlying layer), and the whole operation is also a transaction.

Sometimes there will be multiple DAO calls. Through each call is a transaction, the overall efficiency is low because of too many calls. The business layer can start transactions before all the DAO calls, and end the transactions after all the DAO calls.

```
NSString* filePath = [[NSBundle mainBundle] pathForResource:@"demo_cor_id<DemoProtocol> proxy = [[APDataCenter defaultCenter] daoWithPath:[proxy daoTransaction:APDAOTransactionBegin];  
...// massive DAO invocations  
[proxy daoTransaction:APDAOTransactionCommit];
```

Here we directly call the daoTransaction method of the proxy. This method is actually declared in APDAOProtocol, so the DemoProtocol should be inherited from the APDAOProtocol. (Actually, all the DAO proxy objects support methods in APDAOProtocol. It is for the business layer to decide whether to declare that its protocols are inherited from the APDAOProtocol.)

Calling the daoTransaction method of the proxy is the most easy and straightforward way. In fact, as the userDependent argument in the example above is "YES" when the proxy is created, the database file it operates on is actually APDataCenter.userPreferences. So the example above is equivalent to the call below:

```
NSString* filePath = [[NSBundle mainBundle] pathForResource:@"demo_cor_id<DemoProtocol> proxy = [[APDataCenter defaultCenter] daoWithPath:[APUserPreferences daoTransaction:APDAOTransactionBegin];  
...// massive DAO invocations  
[APUserPreferences daoTransaction:APDAOTransactionCommit];
```

If the userDependent argument is "NO" when the proxy is created, the daoTransaction method of APCommonPreferences can be called.

Here is a well-structured calling template for putting a series of DAO operations into the transaction.

```
- (BOOL)multipleDAOInvocations
{
    APDAOResult* transactionResult = [proxy daoTransaction:APDAOTransac-
    if ([transactionResult failed])
        return NO;

    BOOL daoResult = YES;
    @try
    {
        // massive DAO invocations
        daoResult = [[proxy daoMethod1] succeeded];
        if (!daoResult)
            return NO;

        daoResult = [[proxy daoMethod2] succeeded];
        if (!daoResult)
            return NO;

        return daoResult;
    }
    @finally
    {
        [proxy daoTransaction:daoResult ? APDAOTransactionCommit : API
    }
}
```

@cnName 2.3.6 Function overloading @priority 6

2.3.6 Function overloading

DAO interfaces (@protocol) match the methods in the XML configuration file using the method name and argument count. The argument name has nothing to do with the matching. Argument names in the XML configuration file can be different from the selector argument names of the DAO interface, as long as they correspond to each other in order.

For example, two interfaces are defined in XML, with the same method name, but different argument counts.

```
<select id="getMessage" arguments="id" result="[{}]">
    select * from ${T} where id=#{id}
</select>
<select id="getMessage" arguments="id, time, state" result="{}">
    select * from ${T} where id=#{id} and time=#{time} and state=#{sta
</select>
```

- Definition in DAO interfaces. ````C @protocol DemoProtocol
- (NSArray)getMessage:(NSString)msgId;
- (NSDictionary)getMessage:(NSString)msgId msgTime:
(NSNumber)msgTime msgState:(NSNumber)msgState; @end ````

We can see that the selector argument name in the second method in the DAO interface is inconsistent with the argument name in XML file. This is allowed and the call will not be affected.

@cnName 2.3.7 Concurrent SELECT operations @priority 7

2.3.7 Concurrent SELECT operations

Default SQLite distributions are fully serialized. The SQLite that comes with the iOS platform is multi-threaded and can support connections with multiple databases. DAO has a new feature supporting concurrent SELECT operations, and supporting specifying a SELECT method for connections of the user's database. In this way, in case of intensive writing operations to the database, the SELECT operation will not be suspended waiting.

```
<select id="getTemplatesById" arguments="id" result="[PPDynamicTemplate]>
    select * from ${T} where id = #{id}
</select>
```

The definition method is as above, by simply adding the argument parallel="true" after the SELECT method.

Add the prepareParallelConnection method for APDAOProtocol

```
 /**
 * Create a parallel connection for the database, for the purpose of
 * The connections created will be automatically closed and the business
 *
 * @param autoclose Automatically close the connections if they are not used
 */
- (void)prepareParallelConnection:(NSTimeInterval)autoclose;
```

You can call this method while creating the DAO proxy object. Data Center will prepare an additional connection for this database to cope with possible intensive operations that may follow. If you didn't call this method and prepare the database connection in advance, you can create it as needed

when calling the SELECT method. These database connections will be closed automatically in the idle status and the business layer does not need to manage them.

@cnName 2.3.8 High-level syntax @priority 8

2.3.8 High-level syntax

[TOC]

Shortcut syntax

```
<insert id="addMessages" arguments="messages" foreach="messages.model">
    insert or replace into ${T} (id, time, content, uin, read, date) .
</insert>
```

Taking the INSERT statement above for example, if many properties of the model object need to be inserted to the database, each property in the values will be written, which is both time- and effort-consuming.

Here is another example.

```
<update id="updateTemplate" arguments="template">
    update ${T} set data = #{template.data}, tag = #{template.tag} where ...
</update>
```

if many properties after the SET statement require to be updated, the SQL statements will be very complicated.

The shortcut syntax can come in handy:

```
%{ '#{model.*}' }, msgId, time, content, uin, read, date}
```

is equivalent to

```
#{model.msgId}, #{model.time}, #{model.content}, #{model.uin}, #{model.read},
```

```
%{* = #{model.*}', data, tag, id, version}
```

is equivalent to

```
data = #{model.data}, tag = #{model.tag}, id = #{model.id}, version =
```

The basic format is:

```
%{'format', ...}
```

The first argument is bracketed by single quotes, indicating the format. Following it is the data to be formatted. DAO will format each piece of data once and concatenate the data after DAO replaces the “*” in format with corresponding data. Do not worry about the compromising performance as this pre-processing will be executed only once.

The two examples above can be modified to:

```
<insert id="addMessages" arguments="messages" foreach="messages.model">
    insert or replace into ${T} (id, time, content, uin, read, date) \
</insert>

<update id="updateTemplate" arguments="template">
    update ${T} set %{* = #{template.*}'}, data, tag} where tplId = #
</update>
```

TRY EXCEPT

```

<insert id="insertTemplates" arguments="templates" foreach="templates"
    <try>
        insert into ${T} (tplId, tplVersion, time, data, tag) values(${model.*})
    <except>
        update ${T} set %{'* = #{model.*}', data, tag} where tplId = ${model.tplId}
    </except>
    </try>
</insert>

```

Sometimes the same model may be inserted into the database for multiple times. In this case, the INSERT or REPLACE statements will lead to the loss of old data when conflicts occur in the model primary key (Another model of the same primary key already exists in the database), and the data is re-inserted. This will lead to changes in the rowid of the same piece of data. (Usually you don't need to care about the rowid. But sometimes the rowid is required to be persistent throughout the entire life cycle of the database file.) The TRY EXCEPT statement block can solve the problem, among others. TRY EXCEPT statements can only be used in definitions of DAO methods and should have no preceding or following statements. Other statement blocks can be included inside the TRY and EXCEPT statements.

During execution of this DAO method, if execution of the TRY statement fails, the execution will move to the EXCEPT statement automatically. Only when both of them fail will this DAO call return the failure result.

Non-nil argument

```

<insert id="addMessage" arguments="$ model">
    insert or replace into ${T} ${table_columns} values(#${model.msgId}, ${model.message})
</insert>

```

If an argument has a \$ symbol at its beginning, it indicates that this argument does not accept the nil value. Calling DAO interfaces from the business layer allows nil arguments. But if an argument has a \$ symbol at its beginning and the caller accidentally passes a nil value, the DAO call will fail automatically to prevent unexpected issues from happening.

Conditional formatting

Conditional formatting is akin to the foreach keyword. It will receive a dictionary argument, and format the k-v pairs in the dictionary into the format of "k1=v1, k2=v2".

```
<insert id="updateMessage" arguments="values, conditions">
    update ${T} <format prefix="set" pairs="values" join="," /><format
</insert>
```

If the incoming values are @{@"uin":@(12345), @"content":@"12345"}, and the conditions are @{@"id":@"100"},

SQL statements generated by this method are as follows:

```
update table set uin = ?, content = ? where id = ?
```

and the @(12345), @"12345" and @"100" are bound to SQLite.

[Attention] When a condition passes an empty value for pairs, the prefix argument will not be concatenated into the SQL statements either.

@cnName 2.4.1 APLRUMemoryCache @priority 1

1.2.4.1 APLRUMemoryCache

[TOC]

Introduction

APLRUMemoryCache offers to the memory a cache service in the LRU elimination algorithm for caching ID objects. APLRUMemoryCache is thread-safe and the LRU algorithm is achieved based on the chain table structure, featuring high efficiency.

Interface introduction

- **@property (nonatomic, assign) BOOL handleMemoryWarning; // default NO**

This interface sets whether to process the memory warnings in the system.

- **(id)initWithCapacity:(NSInteger)capacity;**

This interface specifies the capacity at initiation.

- **(void)setObject:(id)object forKey:(NSString*)key;**

This interface stores the object into the cache. If the object is nil,

- **(void)setObject:(id)object forKey:(NSString*)key expire:(NSTimeInterval)expire;**

This interface stores the object into the cache and specifies an expire time.

- (id)objectForKey:(NSString*)key;

This interface fetches the object.

- (void)removeObjectForKey:(NSString*)key;

This interface deletes the object.

- (void)removeAllObjects;

This interface deletes all the objects.

- (void)addObjects:(NSDictionary*)objects;

This interface adds data in batch. It is not able to set the expire time.

- (void)removeObjectsWithRegex:(NSString*)regex;

This interface deletes data in batch. The data key matches regex.

- (void)removeObjectsWithPrefix:(NSString*)prefix;

This interface deletes all the data with a certain prefix in batch.

- (void)removeObjectsWithSuffix:(NSString*)suffix;

This interface deletes all the data with a certain suffix in batch.

- (void)removeObjectsWithKeys:(NSSet*)keys;

This interface deletes all the data specified by the keys.

- (NSArray*)peekObjects:(NSInteger)count fromHead:(BOOL)fromHead;

This interface reads the cached data into an array, but no LRU cache is used.

- (BOOL)objectExistsForKey:(NSString*)key;

This interface quickly judges whether an object for a certain key exists.

- (void)resetCapacity:(NSInteger)capacity;

This interface updates the capacity. If the new capacity is smaller than the current capacity, the capacity will be set to the new value.

@cnName 2.4.2 APLRUDiskCache @priority 2

2.4.2 APLRUDiskCache

[TOC]

Introduction

APLRUDiskCache offers persistence cache to the database in the LRU elimination algorithm and supports NSCoding objects. Databases are easier to maintain than files and databases also make the disk tidier.

Interface introduction

- (id)initWithName:(NSString*)name capacity:(NSInteger)capacity userDependent:(BOOL)userDependent crypted:(BOOL)crypt;

This interface creates a persistence LRU cache. The cached objects should be NSCoding objects.
* @param name Cache name, used as the table name of the database.
* @param capacity Capacity. The real capacity will be greater than the specified value.
* @param userDependent Dependent on the user or not. If yes, when All users log in, the same object will be shared by all users. After the user is switched, the cache will automatically point to the new user's data.
* @param crypted Whether to encrypt the data.
* @return The cache instance. The business should hold it.

- (void)setObject:(id)object forKey:(NSString*)key;

This interface caches an object. The expire value is 0 by default, which means the object will never expire.

- (void)setObject:(id)object forKey:(NSString*)key expire:(NSTimeInterval)expire;

This interface caches an object and specifies an expiration timestamp
* @param object Object. If it is nil, the object for the specific key will be removed.
* @param key Key
* @param expire The expiration timestamp. It specifies an absolute time.

- (id)objectForKey:(NSString*)key;

This interface fetches the object. If the specified expiration time has passed, the object will be deleted.

- (void)removeObjectForKey:(NSString*)key;

This interface deletes the object.

- (void)removeAllObjects;

This interface deletes all the objects.

- (void)addObjects:(NSDictionary*)objects;

This interface adds data in batch.

- (void)removeObjectsWithSqlLike:(NSString*)like;

This interface deletes data in batch. The data keys are matched with the specified SQL like condition.

- (void)removeObjectsWithKeys:(NSSet*)keys;

This interface deletes all the data specified by the keys.

@cnName 2.5.1 APCustomStorage @priority 1

2.5.1 APCustomStorage

[TOC]

Introduction

The default storage space for APDataCenter is the '/Documents/Preferences' directory of the application sandbox.

If a business is independent, or has a large volume of data, APCustomStorage can be utilized to create a storage directory of your own. All the services provided by Data Center apply to this directory, which is similar to the APDataCenter. E.g.,

```
APCustomStorage* storage = [APCustomStorage storageInDocumentsWithName:@"Documents/Contact"];
```

This will create a 'Documents/Contact' directory. The 'commonPreferences' for storing public data and the 'userPreferences' for storing user-specific data also exist in the directory. APCustomStorage is similar to APDataCenter, and the business also needs to pay no attention to user switching.

Interface introduction

+ (instancetype)storageInDocumentsWithName:(NSString*)name;

This interface creates a custom storage with its path being /Documents/

- (id)initWithPath:(NSString*)path;

This method is usually not used for creating a custom storage in an application.

- (APBusinessPreferences*)commonPreferences;

User independent global storage objects access data using the key-value pair interface.

- (APBusinessPreferences*)userPreferences;

The storage object of the current user logged in accesses data using the key-value pair interface.

- (id)daoWithPath:(NSString*)filePath userDependent:(BOOL)userDependent;

Refer to the homonymic interface in APDataCenter.

- (APAsyncFileArrayService)asyncFileArrayServiceWithName:(NSString)name userDependent:(BOOL)userDependent capacity:(NSInteger)capacity crypted:(BOOL)crypted;

This interface creates an asynchronous file array management service.

@param name Service name, cannot be empty.

@param userDependent Dependent on the user or not.

@param capacity Capacity. When the defined value is exceeded, the system will automatically increase it.

@param crypted Whether to encrypt the file contents.

@return The service object, which should be held by the business.

**- (APOBJECTARRAYSERVICE) objectArrayServiceWithName:
(NSString)name userDependent:(BOOL)userDependent capacity:
(NSInteger)capacity cacheCapacity:(NSInteger)cacheCapacity crypted:
(BOOL)crypted;**

This interface creates an array management service for storing similar objects.

@param name Service name, cannot be empty.
@param userDependent Dependent on the user or not.
@param capacity Capacity. When the defined value is exceeded, the system will automatically delete old data.
@param cacheCapacity The capacity of records in the memory cache. When the capacity is exceeded, the system will automatically delete old data.
* @param crypted Whether to encrypt the ID objects.

@return The service object, which should be held by the business.

@cnName 2.5.2 APAsyncFileArrayService @priority 2

2.5.2 APAsyncFileArrayService

[TOC]

Introduction

APAsyncFileArrayService offers file array storage using the database. You can use this service for storing a large number of similar files (such as voice chatting files). APCustomStorage is used to create this service. The business should create the APCustomStorage first, specify your working directory, create an APAsyncFileArrayService with APCustomStorage, and then use the service to write the file array in the database file in your own directory.

Interface introduction

For the asynchronous interface with incoming completion argument, the completion argument will definitely be called back in the main thread, and its value can be nil.

- (void)writeFile:(NSData)data name:(NSString)name completion:(void(^)(BOOL result))completion;

This interface writes a file named "name". When the asynchronous write

- (void)readFile:(NSString)name completion:(void(^)(NSData data))completion;

This interface reads the file named "name" asynchronously. When the op

- (NSData)readFileSync:(NSString)name;

This interface reads the file synchronously.

**- (void)readFilesLike:(NSString)pattern completion:(void(^)(
NSDictionary result))completion;**

This interface uses the like function of SQLite to read batch data. Th

**- (void)renameFile:(NSString)name newName:(NSString)newName
completion:(void(^)(BOOL result))completion;**

This interface renames the file asynchronously.

- (void)removeFile:(NSString*)name;

This interface deletes the file asynchronously.

- (void)removeFilesLike:(NSString*)pattern;

This interface deletes data in batch using the LIKE function of SQLite.

- (void)removeAllFiles;

This interface deletes all the files asynchronously.

- (BOOL)fileExists:(NSString*)name;

This interface checks whether a file exists. It is a synchronous inter

- (NSArray*)allFileNames;

This interface gets names of all the files. It is a synchronous inter-

- (NSInteger)fileCount;

This interface gets the file count. It is a synchronous interface.

@cnName 2.5.3 APObjectArrayService @priority 3

2.5.3 APObjectArrayService

[TOC]

Introduction

APObjectArrayService is implemented based on APAsyncFileArrayService and stores Objective-C objects supporting the NSCoding protocol. Unlike APAsyncFileArrayService, this service offers the memroy cache feature. It should be created using APCustomStorage. The business should create the APCustomStorage first, specify your working directory, create an APObjectArrayService with APCustomStorage, and then use the service to write the object array in the database file in your own directory.

Interface introduction

- (void)setObject:(id)object forKey:(NSString*)key; // Write IO is asynchronous.

This interface sets the object.

- (id)objectForKey:(NSString*)key; // If the object does not exist in the cache, read the database synchronously.

This interface fetches the object.

- (void)objectForKey:(NSString*)key completion:(void(^)(id object))completion;

This interface reads the object asynchronously. After the read is suc

**- (void)objectsForKeyLike:(NSString)pattern completion:(void(^)(
NSDictionary result))completion;**

This interface uses the LIKE function of SQLite to read batch data. Th

- (void)removeObjectForKey:(NSString*)key;

This interface deletes data.

- (void)removeObjectsForKeyLike:(NSString*)pattern;

This interface deletes data in batch using the LIKE function of SQLite

- (void)removeAllObjects;

This interface deletes all the data.

- (BOOL)objectExistsForKey:(NSString*)key;

This interface judges whether a key-value data exists.

- (NSArray*)allKeys;

This interface gets all the keys in the cache.

- (NSInteger)objectCount;

This interface gets the count of cached objects.

@cnName 3 FAQs @priority 3

3 FAQs

3.1 How to set the user mode for Data Center?

Applications connected to mPaaS will use their own account systems. If you want to manage the user mode data with Data Center, notify the Data Center as soon as possible for the Data Center to switch user databases. Then you can notify other business layers.

```
[[APDataCenter defaultDataCenter] setCurrentUserId:userId];
```

When a user signs out, you don't have to call the setCurrentUserId method. Data Center will continue to open the database of last user, which will not cause problems.

3.2 How to set your default encryption key?

Data Center provides a default encryption method. Its key will be automatically generated by the appKey passed by the mPaasInit method. We recommend applications accessed to the mPaaS use their own keys.

You can implement the methods below of the mPaasApplInterface to pass the key in NSData to Data Center.

```
#pragma mark Data Center

/***
 * If this method is implemented, you need to return the 32-byte enc
 * You can also choose not to implement this method. In this case, Da
 *
 * @return The 32-byte key in NSData.
 */
- (NSData*)appDataCenterDefaultCryptKey;
```

We recommend you generate your own 32-byte key and convert it to a Base64-encoded string to be stored in the Security Guard. In this method, the static interface of the Security Guard fetches this string and converts it to the NSData type by inverse solution.

3.3 Is Data Center thread-safe?

Yes. All the storage interfaces of Data Center take the thread safety into consideration. They can be called in any thread.

@cnName Audio Recording and Playback @priority 2

Audio Recording and Playback

[TOC]

1 ALPAudioRecord

```
@interface ALPAudioRecord : NSObject <AudioRecordFinishDelegate>

//Volume change timer
//@property(nonatomic, retain) NSTimer *voicePowerTimer;

//Recording data
@property(nonatomic, retain) NSData *cafData;

//Maximum duration for recording. 30 seconds by default.
@property(nonatomic, assign) double maxDuration;

//Minimum duration for recording. 1 second by default.
@property(nonatomic, assign) double minDuration;

//Create time.
@property(nonatomic, retain) NSString *createAt;

//When filePath is empty, it indicates that no file needs to be stored
@property(nonatomic, retain) NSString *filePath;

@property(nonatomic, assign) id <ALPAudioRecordDelegate> delegate;

//When it is set to YES, transcoding and storage will be performed in
//The recommended value is YES.
@property(nonatomic, assign) BOOL recordAsync;

/**
```

Table of Contents |

```
* Start recording.  
* @param filePath The path for saving the recordings. The file name  
* @return The voice type with the voice file URL.  
*  
*/  
- (void)startRecordSavedInFile:(NSString *)filePath;  
  
/**  
* Cancel recording.  
*/  
- (void)cancelRecord;  
  
/**  
* Complete recording.  
*/  
- (void)finishRecord;  
  
/**  
* Recording permission.  
*/  
+ (void)requestMicrophonePermission:(void (^)(BOOL granted))block;  
@end
```

Implementation delegate

```

@protocol ALPAudioRecordDelegate <NSObject>
@optional
/** 
 *  Callback when recording ends.
 *  Cases of recording ending.
 *  1. Call finishRecord to finish recording.
 *  2. Call cancelRecord to cancel recording.
 *  3. The recording duration exceeds the maximum duration set.
 *  4. The path for saving the recording file is invalid.
 *  5. The recording fails.
 */
- (void)recordFinishWithALPAudioMessage:(ALPAudioMessage *)message re
 

/** 
 * The recording times out and the format conversion and automatic sto
*/
- (void)recordWillFinishForTimeOut;

@end

```

2 ALPAudioPlayer

```

@interface ALPAudioPlayer : NSObject

@property (nonatomic, retain) ALPAudioMessage *message;

//+ (ALPAudioPlayer *)sharedALPAudioPlayer;

/** 
 * The page for creating ALPAudioPlayer will be called in the viewWi
*/
- (void)stop;

/**

```

Table of Contents |

```
* Switching to the speaker will trigger some UI display elements to
*/
- (void)switchToSpeaker:(LWAudioRecorderCallbackBlock)block;

/***
 * Play the audio file.
 * @param message Audio file.
*/
- (void)playWithAudioMessage:(ALPAudioMessage *)message;

/***
 * Stop playing the audio file.
 * @param message Audio file.
*/
- (void)stopWithAudioMessage:(ALPAudioMessage *)message;

/***
 * Play the audio file.
 * @param message Audio file.
 * @param block Callback after playing is finished.
*/
- (void)playWithAudioMessage:(ALPAudioMessage *)message stopBlock:(LW

/***
 * Play.
*/
- (void)play;

/***
 * Whether the current audio file is being played.
*/
- (BOOL)isPlayWithAudioMessage:(ALPAudioMessage *)message;

/***
 * Keep to the last play mode, usually executed before the playXXX me
*/
- (void)keepLastPlayMode;
```

@end

@cnName Sharing Component @priority 3

Sharing Component

[TOC]

1 Introduction

The sharing component MPShareKit offers the feature of sharing to micro blogs, WeChat, friends in Alipay wallet, QQ, Laiwang, text messages and other channels. It provides a unified interface to the developers so that they do not need to cope with the differences between various SDK interfaces.

2 Access channel

Before accessing the channels, you must apply for an account on the official websites of various channels. The websites are as follows:

- Weibo: <http://open.weibo.com/>
- WeChat: <https://open.weixin.qq.com/>
- QQ: <http://open.qq.com/>
- Alipay: <http://open.alipay.com/index.htm>
- Laiwang: Contact mPaaS developer

3 Accessing SDK

MPShareKit carries the resource packages for multiple channels which should be added to the application. In addition, the urlScheme (available in various sharing channels) should be added to the Info.plist file of the application.

- The schemes of WeChat and Laiwang for callback of the source application are both the allocated 'key'.

- The scheme of Weibo is "wb"+key'.
- The scheme of QQ is "tencent"+APPID'.
- The identifier of Alipay is "alipayShare" and its scheme is the allocated 'appID'.

For details, see official documents of the channels.

4 Initializing key of sharing channel

The mPaaS MPShareKit offers two ways to register keys. They differ with the client frameworks.

4.1 When client framework is used

The method under the mPaaSAppInterface protocol is implemented and the key and secret values are returned for the method in the form of a dictionary.

```
- (NSDictionary*)appShareKitConfig
{
    /**
     * The key and secret are used by the demo application for sharing
     */
    NSDictionary *configDic = @{@"laiwang" : @{@"key" : @"your_key", @"secret" : @"your_secret"}, @"weixin" : @{@"key":@"wxcc5c09c98c276ac86", @"secret":@"wx3134a3a7a0e33ec3"}, @"weibo" : @{@"key":@"1877934830", @"secret":@"secret"}, @"qq" : @{@"key":@"1104122330", @"secret":@"secret"}, @"alipay" : @{@"key":@"2015060900117932",@"secret":@"secret"}};

    return configDic;
}
```

4.2 When client framework is not used

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

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWitho
{
    // The key and secret are used by the demo application for sharing
    NSDictionary *dic = @{@"laiwang" : @{@"key" : @"your_key", @"secret" : @"your_secret"}, @"weixin" : @{@"key" : @"wxid_5c09c98c276ac86", @"secret" : @"d56057d8a4"}, @"weibo" : @{@"key" : @"1877934830", @"secret" : @"1067b501c42f484262"}, @"qq" : @{@"key" : @"1104122330", @"secret" : @"WyZkbNmE6d0rDTLf"}, @"alipay" : @{@"key" : @"2015060900117932"}/*The bundleID for the key and secret*/};

    [APSClient registerAPPConfig:dic];
}
```

5 Interface description

5.1 Evoke Sharing Launchpad

You can specify the channels to be displayed while evoking the Sharing Launchpad.

```
NSArray *channelArr = @[kAPSChannelQQ, kAPSChannelLaiwangContacts, kAPSChann
self.launchPad = [[APSLaunchpad alloc] initWithChannels:channelArr so
self.launchPad.delegate = self;
[self.launchPad showForView:[[UIApplication sharedApplication] keyWindow]];
```

5.2 Completing sharing operation

Handle sharing in the callback of 'sharingLaunchpad' function in '@protocol APSLaunchpadDelegate'.

```
- (void)sharingLaunchpad:(APSLaunchpad *)launchpad didSelectChannel:  
{  
    [self shareUrl:channelName];  
    [self.launchPad dismissAnimated:YES];  
}  
  
- (void)shareUrl:(NSString*)channelName  
{  
    //Generate data and call corresponding channels for sharing.  
    APSKMessage *message = [[APSKMessage alloc] init];  
    message.contentType = @"url";//The types are "text", "image", and  
    message.content = [NSURL URLWithString:@"www.sina.com.cn"];  
    message.icon = [UIImage imageNamed:@"1"];  
    message.title = @"title";  
    message.description = @"description";  
  
    APSKClient *client = [[APSKClient alloc] init];  
  
    [client shareMessage:message toChannel:channelName completionBloc  
        //userInfo is the extended information.  
        if(!error)  
        {  
            //your logistic  
        }  
        NSLog(@"error = %@", error);  
    }];  
}
```

5.3 Returning from channel application

When the client framework is used, no processing is required and the framework will take charge. When the client framework is not used, refer to the codes below:

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
{
    //After the sharing is complete, return from the channel applicat:
    BOOL ret;
    ret = [APSKClient handleOpenURL:url];
    return ret;
}
```

Handling of Chinese Pinyin

[TOC]

1 Introduction

The processing of Chinese Pinyin is a tedious job. The Chinese Pinyin processing module provided by mPaaS is able to greatly simply the processing on iOS applications. The module has the following features:

(1) 'Chinese characters to Pinyin'

- This feature can generate Pinyin according to the Unicode of Chinese characters.

(2) 'Search'

- This feature supports searching by two fields. Among them, the primary field supports searching of Pinyin and polyphonic characters, and the subsidiary field searches by matched strings.
- The search result will return the index and location of matching data for the primary field, and the index of matching data for the subsidiary field. The matching data for the two fields are not duplicated.
- This feature is fast for establishing indexes and searching.

2 Function

2.1 Chinese characters to Pinyin

Call the method in the 'APPinyinSearchManager' class.

```

/**
 *brief Acquire the Pinyin string of the character string.
@param text: The input character string.
@return The Pinyin string.
*/
+ (NSString *)getPinYinWithText:(NSString *)text;

```

2.2 Search

2.2.1 Organizing data

The data models to be searched should implement the protocol for searching data and define the primary and subsidiary fields for the search.

```

@protocol APPinyinSearchDataProtocol <NSObject>
@required
- (NSString *)primarySearchData; //The primary search field.
@optional
- (NSString *)secondarySearchData; //The subsidiary search field.
@end

```

2.2.2 Creating search index

The caller creates and holds the 'APPinyinSearchManager' instance object. Create indexes by passing the organized data to the 'APPinyinSearchManager' object. The two methods below are supported for creating the search index:

1. The APPinyinSearchManager will not hold the incoming data, but the location of the data in the original dictionary and array, and the primary and subsidiary fields.
2. Calling the buildIndex method for the APPinyinSearchManager will overwrite the old search indexes. When it passes a nil value, the search index will be reset.

3. You can establish multiple APPinyinSearchManager instances to search for data in different dimensions.
4. The index building is asynchronous. If you want to use the search function while an index is being built, the search operation will be executed and called back automatically after the indexing is complete.

```
/**
 * @brief Building search index.
 * @param dict: Data for implementing the protocol (Ex: {A:[contact1,
 * @param indexChar: The ordered keys of data dictionaries. It is used
 */
- (void)buildSearchIndexWithDataDict:(NSDictionary *)dict indexChar:(I

/**
 * @brief Building search index.
 * @param array: Data array. Ex: [contact1,contact2,contact3]
 */
- (void)buildSearchIndexWithDataArray:(NSArray *)array;
```

2.2.3 Search

If the search is called while an index is being created, 'APPinyinSearchManager' will conduct sequential search and callbacks after the index is created.

```
/**
 * @brief Searched data. The primary field is prioritized and matched by
 * @param searchText: Searched string.
 * @param owner: The call owner. Usually "self" is passed. It is used -
 * @param callBack: Callbacks of the search result.
 */
- (void)search:(NSString*)searchText owner:(id)owner completionBlock:
```

2.2.4 Returning search result

The search result will be returned through the callback function block below.

```
/**  
 * @brief Callbacks of the searched string.  
 * @param searchText: Searched string.  
 * @param primaryMatchArrayWithPosition: Matched data for the primary  
 * @param secondaryMatchArray: Matched data for the subsidiary field.  
 * @param error: Reserved error field.  
 */  
typedef void (^ APPinyinSearchCallback) (NSString * searchText, NSMutableArray  
                                         * primaryMatchArrayWithPosition,  
                                         * secondaryMatchArray, NSError ** error);
```

The data in the 'primaryMatchArrayWithPosition' array is structured as follows:

```
@interface APPinyinSearchPosition : NSObject
/**
@brief The starting point of the name match.
*/
@property (nonatomic, assign) int matchStart;
/**
@brief The ending point of the name match.
*/
@property (nonatomic, assign) int matchEnd;
/**
@brief Whether to match all Pinyin.
*/
@property (nonatomic, assign) BOOL matchAllInPy;
/**
@brief Whether to match all words.
*/
@property (nonatomic, assign) BOOL matchAllInWord;
/**
@brief The matching IndexPath of the searched data. If it is a dictionary,
*/
@property (nonatomic, retain) NSIndexPath * indexPath;
```

Example calling code:

```
[self.contactSearchManager search:searchText owner:self completionBlock:  
    if (weakSelf.isSearchMode && [searchText isEqualToString:self.searchText]) {  
        weakSelf.searchResultArray = [[NSMutableArray alloc] init];  
        //Find the matched data for the primary field.  
        for (int i = 0; i < [primaryMatchArrayWithPosition count]; i++) {  
            APPinyinSearchPosition * position = [primaryMatchArrayWithPosition objectAtIndex:i];  
            NSIndexPath * indexPath = position.indexPath;  
            APContactInfo * contact = [weakSelf contactInfoInDataDictWithIndex:indexPath];  
            //Add to the result array.  
            [weakSelf.searchResultArray addObject:contact];  
        }  
        //Record the matched data position for the primary field.  
        weakSelf.searchResultPositionArray = primaryMatchArrayWithPosition;  
        //Find the matched data for the subsidiary field.  
        for (int i = 0; i < [secondaryMatchArray count]; i++) {  
            NSIndexPath * indexPath = [secondaryMatchArray objectAtIndex:i];  
            APContactInfo * contact = [self contactInfoInDataDictWithIndex:indexPath];  
            //Add to the result array.  
            [weakSelf.searchResultArray addObject:contact];  
        }  
        [weakSelf.tableView reloadData];  
    }  
}];
```

2.2.5 Highlighting matches

Display the position of matched data for the primary field using 'APPinyinSearchPosition'.

```
if (self.nameSearchPostion) {  
    NSUInteger start = self.nameSearchPostion.matchStart;  
    NSUInteger end = self.nameSearchPostion.matchEnd;  
    if (self.contactInfo.displayName.length > 0) {  
        isMatch = YES;  
        [self.contactNameLabel setText:self.contactInfo.displayName a-  
            NSRange redRange = NSMakeRange(start, (end-start+1));  
            [mutableAttributedString addAttribute:(NSString *)kCTForeg-  
            return mutableAttributedString;  
    }];  
}  
}
```

Display the position of matched data for the subsidiary field.

```
__weak APContactTableViewCell * weakSelf = self;  
[self.detailLabel setText:detailString afterInheritingLabelAttributes/  
    NSRange redRange = [detailString rangeOfString:weakSelf.searchText];  
    [mutableAttributedString addAttribute:(NSString *)kCTForegroundCo-  
    return mutableAttributedString;  
}];
```

2.2.6 Others

```
/**  
 * @brief Reset search index.  
 */  
- (void)resetSearchTree;  
/**  
 * @brief Cancel response to search operations.  
 * @param owner: Caller of the search method.  
 */  
- (void)cancelSearchForOwner:(id)owner;
```


@cnName iOS APSecurityUtility @priority 5

iOS APSecurityUtility

[TOC]

1 Introduction

APSecurityUtility offers a series of useful security testing interfaces mainly including the runtime Hook checks, external call checks and debugging status checks.

The application scenario should be explicated before you call the Hook testing interface. You should select the correct testing interfaces according to the programming language, whether the framework API functions of the iOS system will be tested, or the self-defined functions.

2 Interface introduction

2.1 APPDetectClassSelectorSwizzling

```
/**  
 * This method checks whether the (+)selector is hooked by an injector.  
 * It only applies to the selector implemented within this application.  
 *  
 * (BOOL)APDetectClassSelectorSwizzling(NSString *className, SEL sel)  
 *  
 *-----  
 * @param className selector Name of the class the selector belongs to  
 * @param sel Selector  
 *  
 * @return The returned value is of the BOOL type. When YES is returned,  
 * it means the selector is hooked.  
 *-----  
 *  
 * For example:  
 * BOOL isHooked = APDetectClassSelectorSwizzling(@"SomeClass", @selector(someMethod));  
 */
```

2.2 APDetectInstanceSelectorSwizzling

```
/**  
 * This method checks whether the (-)selector is hooked by an injector  
 * It only applies to the selector implemented within this application  
 *  
 * (BOOL)APDetectInstanceSelectorSwizzling(NSString *className, SEL :  
 *  
 *-----  
 * @param className selector Name of the class the selector belongs to  
 * @param sel Selector  
 *  
 * @return The returned value is of the BOOL type. When YES is returned,  
 *-----  
 *  
 * For example:  
 * BOOL isHooked = APDetectInstanceSelectorSwizzling(@"SomeClass", @:  
 *  
 */
```

2.3 APDetectCFuncCalledByOthers & APDetectNonFrameworkSelectorCalledByOthers

```
/**  
 * This method is added in the internal implementation of C functions.  
 *  
 * For example:  
 * bool isRisky = APDetectCFuncCalledByOthers;  
 *  
 */
```

2.4 APDetectUIKitInstanceSelectorSwizzling

```
/**  
 * This method checks whether the (-)selector in the UIKit framework  
 * It only applied to the API defined in the UIKit Framework of the :  
 *  
 * (BOOL)APDetectUIKitInstanceSelectorSwizzling(NSString *className,  
 *  
 *-----  
 * @param className selector Name of the class the selector belongs to  
 * @param sel Selector  
 *  
 * @return The returned value is of the BOOL type. When YES is returned  
 *-----  
 *  
 * For example:  
 * BOOL isHooked = APDetectUIKitInstanceSelectorSwizzling(@"UITextField"  
 *  
 */
```

2.5 APDetectLocalAuthInstanceSelectorSwizzling

```
/**  
 * This method checks whether the (-)selector in the LocalAuthentication  
 * It only applied to the API defined in the LocalAuthentication.framework.  
 * It does not apply to the simulator.  
 *  
 * Attention: The caller needs to ensure the linking with LocalAuthenti  
 * When systems lower than iOS8 cannot link the LocalAuthentication.  
 *  
 * (BOOL)APDetectLocalAuthInstanceSelectorSwizzling(NSString *className,  
 *  
 *-----  
 * @param className selector Name of the class the selector belongs to.  
 * @param sel Selector  
 *  
 * @return The returned value is of the BOOL type. When YES is returned  
 *-----  
 *  
 * For example:  
 * BOOL isHooked = APDetectLocalAuthInstanceSelectorSwizzling(@"LACo  
 *  
 */
```

2.6 APDetectJailbroken

```
/**  
 * Jailbreak check.  
 *  
 * (BOOL)APDetectJailbroken;  
 */
```

2.7 APDetectDebugger

```
/**  
 * This method checks whether the current application is in the debug  
 * (BOOL)APDetectDebugger;  
 *  
 * @return The returned value is of the BOOL type. When YES is returned,  
 */
```

2.8 APGetTextSectionDigest

```
/**  
 * This method acquires the abstract of binary code snippets of executables.  
 *  
 * (NSString *)APGetTextSectionDigest;  
 */
```

2.9 APDetectLinkerFlag

```
/**  
 * This method checks the "restrict" flag in the linker flag of binary files.  
 *  
 * (BOOL)APDetectLinkerFlag;  
 * @return The returned value is of the BOOL type. When YES is returned,  
 */
```

2.10 APCheckNotificationSecurity

```
/**  
 * This method checks whether the notification sender is from inside  
 * Important: The notification name string must be a static string.  
 *  
 * Sample:  
 *  
 * NSString *name = [notification name];  
 * if (!APCheckNotificationSecurity(name)) {  
 *     //Do not respond to notifications sent from outside the wallet  
 *     return;  
 * }  
 *  
 * @return The returned value is of the BOOL type. When YES is returned  
 */
```

@cnName H5 Container @priority 6

H5 Container

[TOC]

1 Introduction

H5 Container is composed of the kernel and shell.

The kernel is the Poseidon.framework which is re-encapsulation of UIWebview. It offers abstract objects of the container, JSBrigde, unified URL intercept, event dispatch, plug-in managing, JSAPI managing and other functions.

The shell is a simple instance of the kernel. It offers a webViewController and webview for external use. It also provides the universal JSAPI, universal plug-ins, and simple UI interfaces.

2 Features

1. Abundant universal JSAPI (See details at
<http://ux.alipayinc.com/index.php/H5%E5%AE%B9%E5%99%A8%E6%96%87%E6%A1%A3>).
2. It supports adding self-defined JSAPI.
3. Customized UI elements.

3 Module access

1. Import mPaas.framework to the project.
2. Add the iOS system framework the container is dependent on:
CoreTelephony, SystemConfiguration, MobileCoreServices,
MessageUI, EventKit, AudioToolbox, CoreMotion, QuartzCore, and

CFNetwork.

3. Link H5Service.bundle and Poseidon.bundle to the root directory of the main project. The two bundles will be dispatched to the developer together with mPaas.framework.

After the module is accessed, you can evoke an H5 container with the codes below:

```
H5WebViewController *vc = [[H5Service sharedService] createWebViewCont  
[self.navigationController pushViewController:vc animated:YES];
```

4 Self-defined UserAgent

Configure the customUA field in the startup arguments of the container.

```
H5WebViewController *vc = [[H5Service sharedService] createWebViewCont  
[self.navigationController pushViewController:vc animated:YES];
```

5 Adding JSAPI

5.1 Modifying Poseidon-Config.plist

1. Create a new class inherited from the JsApiHandlerBase, and a handler:context:callback: function will be provided for processing argument input and result output.
2. Add a new record in the JsApis array in Poseidon-Config.plist. The record should contain the class name and the API name exposed to JS, declaring that the JSAPIs here will be registered one by one at the container startup.

This practice is not recommended. We suggest the accessed application manage the newly added APIs and conduct dynamic registration when the application is run. Because after the Poseidon-Config file is modified, you need to pay attention to the overwrite issue when upgrading the H5 container.

5.2 Runtime dynamic registration

1. Define the JsAPI name in the format of “business name.method name”, such as: wealth.getMoney.
2. Acquire the h5service instance.
3. Register JsAPI through h5service.

```
static NSString *jsApiName = @"wealth.getMoney";
H5Service *h5Service = [H5Service sharedService];
PSDJsApiHandlerBlock handlerBlock = ^(NSDictionary *data, PSDContext *context) {
    // The data refers to the arguments passed from the page.
    // dosomething
    // ...
    // The h5webviewcontroller can be obtained in PSDContext.
    // For callback to the page, invoke the responseCallbackBlock.
};

// Unregister the JsAPI first.
[h5Service unregisterApiName:jsApiName];
// Then register the JsAPI.
[h5Service registerApis:@{jsApiName: handlerBlock}];
```

6 New plug-in

1. Create a new class inherited from the PluginBase or implement PSDPluginProtocol. Each plug-in will receive event notifications. You can receive event notifications by implementing the handleEvent: method and perform corresponding operations on an event.

2. The timing of plug-in registration can be summarized to two scenarios. The first one, like the JSAPI, is to add new records in the Plugins array in Poseidon-Config.plist. This category of plug-ins will register JsAPI at the container startup and listens to all the events. The second one is to register plug-ins dynamically before starting a specific H5 application.
3. Plug-ins are referenced by weak references. Please hold them yourself.

```
Plugin4SafePay *payPlugin = [[Plugin4SafePay alloc] init];
[self.plugins addObject:payPlugin];
[self.session addEventListener:kEvent_Navigation_All withListener:payI
```

@cnName 1 UI Controls @priority 1

1 UI Controls

[TOC]

APActionSheet

Class name: APActionSheet, inherited from UIActionSheet.

Description: It defined the ActionSheet base class, and manages ActionSheet in a unified way.

Interface:

```
+ (void)setIsBackGroundMode:(bool)isBackGroundMode;
```

Property:

```
@property(nonatomic,weak)id<UIActionSheetDelegate>dtDelegate;
```

Example:

```
- (void)setupActionSheetWithTitle:(NSString *)title
{
    if (self.mobileContact){
        _contactActionSheet = [[APActionSheet alloc] initWithTitle:title];
        NSArray *ary = [self.mobileContact phoneAry];
        ary = (ary != nil) ? ary : [self.mobileContact accountIdAry];
        for (NSString *content in ary) {
            [_contactActionSheet addButtonWithTitle:content];
        }
        // The cancel button is placed at the end.
        [_contactActionSheet addButtonWithTitle:TRANSFER_TEXT_TO_ACCOUNT];
        [_contactActionSheet showInView:self.view];
    }
}
```

APActionSheetManager

Class name: APActionSheetManager, inherited from NSObject.

Description: It manages APActionSheet.

Interface:

```
/*
 * Create a singleton class.
*/
+ (APActionSheetManager *)sharedAPActionSheetManager;

/*
 * Delete all the actionsheet.
*/
-(void)dismissAllUIActionSheet;

/*
 * Add actionsheet to the "shown" queue.
*/
-(void)addToShowedPool:(APActionSheet*)alertView;
`
```

Property:

```
@property(nonatomic,assign)bool backGroundMode;
```

Example:

```
APActionSheetManager *manager = [APActionSheetManager sharedAPActionS
```

APAgreementBox

Class name: APAgreementBox, inherited from UIControl.

Description: It shows the components of user registration agreements.

Interface:

```
/**  
 * It shows the components of user registration agreements.  
 * @param frame The position and size of the view in the parent view.  
 * @param labelText The title.  
 * @param linkText The hyper link title.  
 * @return The initialized registration agreement components newly created.  
 */  
- (id)initWithFrame:(CGRect)frame labelText:(NSString *)labelText linkText:(  
    NSString *)linkText  
    // Whether to show the check box.  
- (void)setCheckBoxHidden:(BOOL)hidden;
```

Property:

```
// User checks the check box.  
@property(nonatomic, readonly) APCheckBox *checkBox;  
// User clicks the hyperlink button.  
@property(nonatomic, readonly) APLinkButton *linkButton;
```

Example:

```
APAgreementBox *agreementBox = [APAgreementBox alloc] initWithFrame:  
    CGRectMake(100, 100, 200, 150)  
    labelText:@"User Agreement"  
    linkText:@"Read More"  
    checkBoxHidden:NO];  
[self.view addSubview:agreementBox];
```

APAlertView

Class name: APAlertView, inherited from UIAlertView.

Description: Customized UIAlertView. When there are only two buttons, the buttons will be arranged along the up-down direction.

Usage: Like UIAlertView, use this class when you want the two buttons arranged in the up-down direction. Use UIAlertView in other cases.

Example:

```
APAlertView *alertView = [[APAlertView alloc] initWithTitle:@"New alert"
                                                       message:@"We will
                                                       delegate:self
                                                       cancelButtonTitle:@"Get it"
                                                       otherButtonTitles:@"Issues", @"Cancel"];
[alertView show];
```

APAlertViewManager

Class name: APAlertViewManager, inherited from NSObject.

Description: It manages APAlertView.

Interface:

```
/**  
 * Create a singleton class.  
 */  
  
+ (APAlertViewManager *)sharedAPAlertViewManager;  
  
/**  
 * Delete all alertView.  
 */  
- (void)removeAllalertView;  
  
/**  
 * Add alertView to the "shown" queue.  
 */  
- (void)pushAPAlertView:(UIAlertView *)alertView;
```

Property:

```
@property(nonatomic,assign)bool isBackGroundMode;
```

Example:

```
APAlertViewManager *manager = [APAlertViewManager sharedAPAlertViewMa
```

APBankCardTextField

Class name: APBankCardTextField, inherited from APTextField.

Description: It is the text field of the bank card.

Interafces: Refer to APTextField.

Example:

```
APBankCardTextField *textField = [[APBankCardTextField alloc] initWithFrame:  
    textField.validator = [APTextValidator bankCardValidator];
```

APBarButtonItem

Class name: APBarButton, inherited from UIBarButtonItem.

Description: APBarButton is the button object placed on the UIToolbar or UINavigationBar. It responds to the user's actions.

Interface:

Table of Contents |

```
/**  
 *  @return Create a new object of the UIBarButtonItemStyleFlexibleSpace  
 */  
+ (UIBarButtonItem *)flexibleSpaceItem;  
  
/**  
 *  It creates and initializes a button object of "Back".  
 *  @param title      The button title. Note: No effect in settings. The  
 *  @param target     The object responding to the button click event.  
 *  @param action     The function responding to the button click event  
 *  @return The initialized "Back" button object newly created.  
 */  
+ (UIBarButtonItem *)backBarButtonItemWithTitle:(NSString *)title target:  
  
/**  
 *  It creates and initializes a button object with a specified image  
 *  @param image      Button image.  
 *  @param target     The object responding to the button click event.  
 *  @param action     The function responding to the button click event  
 *  @return The newly-created and initialized button object with a spe  
 */  
- (id)initWithImage:(UIImage *)image style:(UIBarButtonItemStyle)style;  
  
/**  
 *  It creates and initializes a button object with a specified title  
 *  @param title      Button title.  
 *  @param target     The object responding to the button click event.  
 *  @param action     The function responding to the button click event  
 *  @return The newly-created and initialized button object with a spe  
 */  
- (id)initWithTitle:(NSString *)title style:(UIBarButtonItemStyle)sty:
```

Example:

```
APBarButtonItem *item = [[APBarButtonItem alloc] initWithImage:[UIImage imageNamed:@"Bill"]];  
self.navigationItem.rightBarButtonItem = item;
```

```
APBarButtonItem *item = [[APBarButtonItem alloc] initWithTitle:@"Bill"];  
self.navigationItem.rightBarButtonItem = item;
```

APBorderedButton

Class name: APBorderedButton, inherited from UIButton.

Description: It achieves the three button styles below by default:

Main button;

Secondary button; Self-defined button. No style will be applied and no initialization of the frame.

Interface:

```
/**  
 * An auxiliary method of the method, used for creating and initializing  
 *  
 * @param buttonType Button type. It must be one of the values defined  
 * @param title      Button title.  
 * @param target     The object responding to the button click event.  
 * @param action     The function responding to the button click event.  
 *  
 * @return The initialized button object newly created.  
 *  
 * The button object created with this method has a default frame of  
 * The events of the specified target and action are <code>UIControlEventTouchUpInside</code>  
 */  
+ (APBorderedButton *)buttonWithType:(APBorderedButtonType)buttonType  
                           title:(NSString *)title  
                          target:(id)target  
                         action:(SEL)action;  
  
/**  
 * It creates and initializes a button object.  
 *  
 * @param title      Button title.  
 * @param buttonType Button type. It must be one of the values defined  
 *  
 * @return The initialized button object newly created.  
 */  
- (id)initWithButtonTitle:(NSString *)title buttonType:(APBorderedButtonType)buttonType  
                           target:(id)target  
                         action:(SEL)action;
```

Example:

```
APBorderedButton *confirmBtn = [APBorderedButton buttonWithType:APBButtonTypeDefault  
                           title:@"Confirm" target:self  
                         action:@selector(confirmAction)];
```

APButton

Class name: APButton, inherited from UIButton.

Description: It achieves the three button styles below by default:

Main button;

Secondary button; Warning button. The button background is usually in the catchy red color. Self-defined button. No style will be applied and no initialization of the frame.

Interface:

Table of Contents |

```
/**  
  
 * An auxiliary method of the method, used for creating and initializing  
 * @param buttonType Button type. It must be one of the values defined  
 * @param title      Button title.  
 * @param target     The object responding to the button click event.  
 * @param action     The function responding to the button click event.  
 * @return The initialized button object newly created.  
  
 * The button object created with this method has a default frame of  
 * The events of the specified target and action are UIControlEventTouchUpInside  
 */  
+ (APButton *)buttonWithType:(APButtonType)buttonType title:(NSString *)title  
  
/**  
  
 * It creates buttons with a specified type.  
 * @param buttonType Button type. It must be one of the values defined  
 * @return The initialized button object newly created.  
 */  
- (id)initWithButtonType:(APButtonType)buttonType;
```

Example:

```
APButton *confirmBtn = [APButton buttonWithType:APButtonTypeDefault t:  
APButton *cancelBtn = [APButton buttonWithType:APButtonTypeSecondary t:
```

APCheckBox

Class name: APCheckBox, inherited from UIControl.

Description: This control indicates a specific status (option): check (on, the value is "true"), or uncheck (off, the value is "false"). Interface:

```
/*
 * Returns an initialized <code>APCheckBox</code> object with the he:
 * @return An initialized check box object.
 */
- (id)init;
```

Property:

```
/**  
 * A boolean value indicates whether the <code>APCheckBox</code> is checked.  
 *  
 * The default value is NO.  
 */  
@property(nonatomic, assign, getter = isChecked) BOOL checked;  
  
/** Gets or sets the image for check state. */  
@property(nonatomic, retain) UIImage *checkedImage;  
  
/** Gets or sets the image for unchecked state. */  
@property(nonatomic, retain) UIImage *uncheckedImage;  
  
/**  
 * The inset or outset margins for the rectangle around the check-box.  
 * The default value is <code>UIEdgeInsetsMake(0, 0, 0, 5.0)</code>.  
 */  
@property(nonatomic, assign) UIEdgeInsets imageEdgeInsets;  
  
/** Returns the label used for title of check box. */  
@property(nonatomic, strong, readonly) UILabel *titleLabel;
```

Example:

```
APCheckBox *checkBox = [[APCheckBox alloc] initWithFrame:CGRectZero];  
[checkBox sizeToFit];  
[self addSubview:checkBox];
```

APCircleRefreshControl

Class name: APCircleRefreshControl, inherited from UIControl.

Description: The refresh view with a circle.

Interface:

```
// Create views on the specified UIScrollView.  
- (id)initInScrollView:(UIScrollView *)scrollView;  
  
// Start the refreshing animations.  
- (void)beginRefreshing;  
  
// End the refreshing animations.  
- (void)endRefreshing;
```

Example:

```
self.refreshControl = [[APCircleRefreshControl alloc] initInScrollView:  
[self.refreshControl addTarget:self action:@selector(refreshEventRecie  
[self.refreshControl beginRefreshing];
```

APCreditCardTextField

Class name: APCreditCardTextField, inherited from APTextField.

Description: It is the text field of the credit card.

Interfaces: Refer to APTextField.

Example:

```
_textField = [[APCreditCardTextField alloc] initWithFrame: UIEdgeInsets:  
_textField.validator = [APTextValidator creditCardValidator];
```

APExceptionView

Class name: APExceptionView, inherited from UIView.

Description: The interface of network exceptions.

Interface:

Table of Contents |

```
/**  
  
 * It initializes the exception view and sets the exception type.  
  
 * @param frame Coordinates of the view, required.  
  
 * @param type Exception type, required.  
  
 * @return APExceptionView.  
  
 */  
  
- (id)initWithFrame:(CGRect)frame exceptionType:(APExceptionEnum) type  
  
/**  
 *Get the read-only UIImageView of large icons.  
 *@return UIImageView  
*/  
  
- (UIImageView *)getIconImageView;  
  
/**  
 *Get the read-only UILabel description copy.  
 *@return UILabel  
*/  
  
- (UILabel *)getInfoLabel;  
  
/**  
 * Get the action button. *@return UIButton  
*/  
  
- (UIButton *)getActionButton;
```

Property:

```
typedef enum {  
    APExceptionEnumNetworkError,  
    APExceptionEnumEmpty,  
    APExceptionEnumAlert  
} APExceptionEnum;
```

Example:

```
APExceptionView *exceptionView = APExceptionView alloc initWithFrame::
```







APIImageAuthCodeBox

Class name: APIImageAuthCodeBox, inherited from APAuthCodeBox.

Description: The input field of image verification code.

Interface:

```
/**  
 * It creates the input field of image verification code.  
 * @param originY The y-coordinate of the component.  
 * @return The input field of image verification code.  
 */  
- (APIImageAuthCodeBox *)initWithOriginY:(CGFloat)originY;
```

Property:

```
@property(strong, nonatomic) UIImageView *authCodeImageView; // Image  
@property(strong, nonatomic) UIImageView *circleImageView; // The c
```

Example:

```
- (void)viewDidLoad  
{  
    APIImageAuthCodeBox *imageBox = [[APIImageAuthCodeBox alloc] initWithFrame:  
        CGRectMake(100, 100, 200, 150)];  
    imageBox.inputBox.textField.placeholder = @"Verification code on the screen";  
    imageBox.authCodeImageView.image = [UIImage imageNamed:@"auth_code.png"];  
    [self.view addSubview:imageBox];  
}  
- (void)refreshAuthImage  
{  
    [imageBox startWaiting];  
    [self performSelector:@selector(authImageReady) withObject:nil afterDelay:1];  
}  
- (void)authImageReady  
{  
    [imageBox stopWaiting];  
}
```

APInputBox

Class name: APInputBox, inherited from UIView. Description: It contains a text field which can be acquired through the textField property, and may contain a button shown on the right side in the input field (The button can be acquired from the iconButton property). If the textFieldFormat is specified, it will group texts in the text field with spaces (usually used for inputting ID numbers). Create: with no button on the right side in the input field.

```
/**  
 * Create input field components.  
 * @param originY The y-coordinate of the input field.  
 * @param type The type of the text field.  
 * @return The input field components.  
 */  
+ (APInputBox *)inputboxWithOriginY:(CGFloat)originY  
    inputboxType:(APInputBoxType)type;
```

with a button on the right side in the input field.

```
/**  
 * Create input field components of buttons with icons.  
 * @param originY The y-coordinate of the input field.  
 * @param icon Icons on the button. Size: 44x44.  
 * @param type The type of the text field.  
 * @return The input field components of buttons with icons.  
 */  
+ (APInputBox *)inputboxWithOriginY:(CGFloat)originY  
    buttonIcon:(UIImage *)icon  
    inputboxType:(APInputBoxType)type;
```

Property:

Internal properties of APInputBox		
@property(strong, nonatomic)	APTextField	*textField;
@property(strong, nonatomic)	NSString	*textFieldText
@property(strong, nonatomic)	NSString	*textFieldFormat
@property(strong, nonatomic)	UIButton	*iconButton
@property(assign, nonatomic)	BOOL	hidesButtonWhileNe
@property(nonatomic, readonly)	UILabel	*titleLabel
@property(assign, nonatomic)	APInputBoxStyle	style
@property(assign, nonatomic)	APInputBoxType	inputBoxType

Example:

- Create example codes of buttons with icons.

```
APIInputBox *phoneNumberBox = [APIInputBox inputboxWithOriginY:0 buttonIcon:[UIImage imageNamed:@"icon"]];  
[phoneNumberBox.iconButton addTarget:self action:@selector(onShow:) forControlEvents:UIControlEventTouchUpInside];  
phoneNumberBox.titleLabel.text = @"Mobile number";  
phoneNumberBox.textField.placeholder = @"Mobile number of contact";  
phoneNumberBox.hidesButtonWhileNotEmpty = YES;
```

Create example codes of buttons without icons. The codes are segmented by the mobile number automatically.

```
APIInputBox *phoneNumberBox = [APIInputBox inputboxWithOriginY:0 inputType:kInputTypeMobilePhone];  
phoneNumberBox.titleLabel.text = @"Mobile number";  
phoneNumberBox.textFieldFormat = @"+### ### ####";  
phoneNumberBox.textField.placeholder = @"Mobile number of contact";
```

Method: Description ```C /**

- Add spaces to the text according to a specified format.
- @param Text content.
- @return Texts after spaces are added. */
- *(NSString)formatText:(NSString)text;*

/**

- This method applies to inputBox with no icon specified at the initialization.
- @param icon Icons on the button. Size: 44x44. /
- *(void)setRightButtonIcon:(UIImage *)icon;*

/**

- Check the input validity. */
- *(BOOL)checkInputValidity;*

```
/**
```

- Filter texts. Only digits are allowed and the maximum length is limited.
- The arguments are the corresponding delegate arguments. The maxLength indicates the maximum length.
 - -(BOOL)shouldChangeRange:(NSRange)range replacementString:(NSString *)string
withMaxLength:(int)maxLength;

```
/**
```

- Limit the maximum length.
- @maxLength The maximum length, not including format spaces.
 - -(BOOL)shouldChangeRange:(NSRange)range replacementString:(NSString *)string withFormatStringMaxLength:(int)maxLength; `

APLinkButton

Class name: APLinkButton, inherited from UIControl.

Description: The button of the hyper link style.

Interface:

No self-defined initialization functions. The initialization functions defined by the parent class are used.

```
/**  
 * Set the color of titles in a specified status.  
 * @param color Color of the title.  
 * @param state The specified status.  
 */  
- (void)setTitleColor:(UIColor *)color forState:(UIControlState)state;  
  
/**  
 * Get the color of titles in a specified status.  
 * @param state The specified status.  
 * @return Color of the title.  
 */  
- (UIColor *)titleColorForState:(UIControlState)state;
```

Property:

```
/** View of the titles of link buttons */  
@property(nonatomic, readonly) UILabel *titleLabel;  
  
/** Titles of link buttons */  
@property(nonatomic, strong) NSString *title;  
  
/** Whether to underline titles */  
@property(nonatomic, assign) BOOL underline;
```

Example:

```
APLinkButton *forgotButton = [[APLinkButton alloc] initWithFrame:shadowFrame];  
[forgotButton setTitle:@"Forgot password?" ] ;  
[forgotButton sizeToFit];
```

APLoginBox

Class name: APLoginBox, inherited from UIView.

Description: Login box control.

Interface:

```
/**  
 *  Input box and password box.  
 */  
- (APInputBox *)usernameInputBox;  
- (APInputBox *)passwordInputBox;  
  
/**  
 * Constructor  
 *  
 * @param originY It specifies the frame.origin.y.  
 * @param flag Whether to include the register button.  
 * @return The login components.  
 *  
 */  
+ (APLoginBox *)loginBoxWithOriginY:(CGFloat)originY registerButton:(BOOL)registerButton;  
  
#pragma mark -  
#pragma mark Customize login to email box of credit card statements  
/**  
 * Set prompt contents in the drop-down box and show the pull-down list.  
 *  
 * @param historyItems Prompt contents in the drop-down box.  
 * @return The increased height after the adjustment.  
 */  
- (NSInteger)setHistoryItemsAndDisplayHistory:(NSArray *)historyItems;  
  
/**  
 * Set text field changes and modify the notification logic.  
 */  
- (void)setupDidBeginEditNotification;
```

Property:

```
**  
 * Account input box.  
 */  
@property(nonatomic, readonly) UITextField *usernameField;  
/**  
 * Password input box.  
 */  
@property(nonatomic, readonly) UITextField *passwordField;  
/**  
 * Verification code input box.  
 */  
@property(nonatomic, readonly) APAAuthCodeBox *authCodeBox;  
/**  
 * Forgot password "link".  
 */  
@property(nonatomic, readonly) APLinkButton *forgotButton;  
/**  
 * Register button.  
 */  
@property(nonatomic, readonly) UIButton *registerButton;  
/**  
 * Login button.  
 */  
@property(nonatomic, readonly) UIButton *loginButton;  
  
/**  
 * History accounts.  
 *  
 * An array. Each element is an NSString and will be shown on the hist  
 */  
@property(nonatomic, copy) NSArray *historyItems;  
/**  
 * Whether to show history account list.  
 */  
@property(nonatomic, assign) BOOL historyTableVisible;  
/**
```

```
* Whether to show verification code control.  
*/  
@property(nonatomic, assign) BOOL authCodeBoxVisible;  
/**  
 * Proxy.  
 */  
@property(nonatomic, assign) id<APLoginBoxDelegate> delegate;
```

Example:

```
_loginBox = [ALPLoginBox loginBoxWithOriginY:90 registerButton:YES];  
_loginBox.historyItems = @[@"13127995722", @"mozart0@tom.com", @"hbhf"];  
_loginBox.delegate = self;  
[_loginBox setLoginBoxOriginXCommonPix];  
[self.view addSubview:_loginBox];
```

APMobileTextField

Class name: APMobileTextField, inherited from APTextField.

Description: Input box control of the mobile number.

Interface: See APTextField.

Example:

```
_textField = [[APMobileTextField alloc] initWithFrame:UIEdgeInsetsInsetByEdgeInsets(0, 0, 0, 0)];  
_textField.validator = [APTextValidator mobileNumberValidator];
```

APNetErrorView

Class name: APNetErrorView, inherited from UIView.

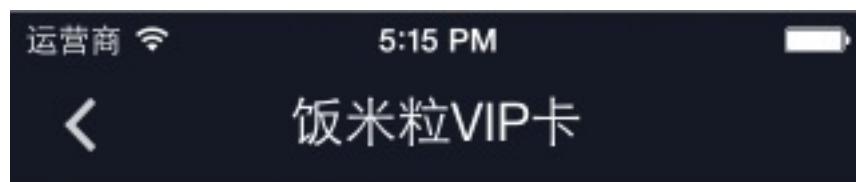
Description: Network error prompt.

Interface:

```
/**  
 * The old control style of network exceptions. The new control APEx  
 * @param parent Show in the specified parent view.  
 * @param target The object responding to the button click event.  
 * @param action The function responding to the button click event  
 * @return The initialized view object newly created.  
 */  
+ (id)showInParentView:(UIView *)parent withTarget:(id)target action:(SEL)action;  
  
/**  
 * Cancel view.  
 */  
- (void)dismiss;  
  
/**  
 * Specify actions to clicks.  
 * @param target The object responding to the button click event.  
 * @param action The function responding to the button click event  
 */  
- (void)setTarget:(id)target action:(SEL)action;
```

Example:

```
self.netErrorView = [APNetErrorView showInParentView:self.mainView withTarget:  
self dismissOnTap:YES];
```



网络无法连接

请检查你的手机是否联网

APNextPagePullView

Class name: APNextPagePullView, inherited from UIView.

Description: Show a circling box when the user clicks on "More". Once the next page is loaded, the circling box disappears.

Interface:

```
/**  
  
 * The method that scroll did occur in the view.  
  
 */  
  
- (void)refreshScrollViewDidScroll:(UIScrollView *)scrollView;  
  
/**  
 * The method that drag did occur in the view.  
 */  
- (void)refreshScrollViewDidEndDragging:(UIScrollView *)scrollView;  
  
/**  
 *Data loading complete.  
 */  
- (void)refreshScrollViewDataSourceDidFinishedLoading:(UIScrollView *)scrollView;  
  
/**  
 * After data loading is complete, the circling box disappears after  
 - (void)refreshScrollViewDataSourceDidFinishedLoadingAndDelay:(UIScrollView *)scrollView;
```

Property:

```
@property(nonatomic, weak) id delegate;

@protocol APNextPagePullViewDelegate
/**
 * Judge whether the next page is being loaded.
 */
- (BOOL)isTableViewLoadingNextPage:(APNextPagePullView *)pullView;

/**
 * Notification that requires view refreshing.
*/
- (void)upPullTableDidTriggerRefresh:(APNextPagePullView *)pullView;
```
Example:
```
APNextPagePullView *view = [[APNextPagePullView alloc] initWithFrame:...]
```

APNumericKeyboard

Class name: APNumericKeyboard, inherited from UIView.

Description: It offers a numeric keyboard with the following static methods.
When the OK button of the keyboard is clicked, the delegate of the textField will receive the textFieldShouldReturn: Message

Interface:

```
+ (APNumericKeyboard *)sharedKeyboard;
```

Property:

```
//Set textinput manually. The y-axis of the keyboard should be set ext
@property (nonatomic, strong) id<UITextInput> textInput;
//Whether the keyboard is used for entering the ID number.
@property (nonatomic, assign, getter = isIdNumber) BOOL idNumber;
//Whether to hide the decimal point.
@property (nonatomic, assign, getter = isDotHidden) BOOL dotHidden;
//Whether to hide the button for tucking the keyboard.
@property (nonatomic, assign, getter = isDismissHidden) BOOL dismissH:
//Whether to hide the OK button.
@property (nonatomic, assign, getter = isSubmitEnable) BOOL submitEna
```

Example:

```
APIInputBox *inputBox1 = [APIInputBox inputBoxWithOriginY:15 placeHolder:
inputBox1.textField.inputView = [ALPNumericKeyboard sharedKeyboard];
[self.view addSubview:inputBox1];
```

APNumPwdInputView

Class name: APNumPwdInputView, inherited from UIView.

Description: The input box control of 6-digit password.

Interface:

Table of Contents |

```
/**  
 * Initialize the 6-digit password.  
 * quadWidth: Width of quad.  
 * quadHeight: Height of quad.  
 */  
- (id)initWithQuadWidth:(CGFloat)quadWidth quadHeight:(CGFloat)quadHe:  
  
/**  
 * Initialize the 6-digit password.  
 *  
 * quadWidth: Width of quad.  
 * quadHeight: Height of quad.  
 * password: Initial password.  
 */  
- (id)initWithQuadWidth:(CGFloat)quadWidth quadHeight:(CGFloat)quadHe:  
  
/**  
 * Clear the password box, which will not trigger the callback of the  
 */  
- (void)reset;  
  
/**  
 * Show the keyboard.  
 */  
- (BOOL)showKeyBoard;  
  
/**  
 * Hide the keyboard.  
 */  
- (BOOL)hideKeyBoard;  
  
/**  
 * Set the background image.  
 */  
- (void)setBackgroundImage:(UIImage *)image;
```

Property:

```
/** User entered password */
@property (nonatomic, strong, readonly) NSString *password;

/** 
 * APNumPwdInputViewDelegate. The delegate can implement the callback
 * - (void)onPasswordDidChange:(APNumPwdInputView*)sender;
 */
@property (nonatomic, assign) id<APNumPwdInputViewDelegate> delegate;

/** 
 * Judge whether it is a 6-digit password control. If NO, the general
 */
@property (nonatomic, assign, getter = isNumericPassword) BOOL numeric;
```

Example:

```
APNumPwdInputView *pwdInputView = [[APNumPwdInputView alloc] initWithFrame:CGRectMake(100, 100, 200, 200)];
pwdInputView.delegate = self;
[self.view addSubview:pwdInputView];
```

APNumPwdPopupView

Class name: APNumPwdPopupView, inherited from UIView.

Description:

A pop-up box to show the 6-digit password control, as well as the following elements:

Back button (optional) Title Subtitle (optional) Cancel button (optional) OK button Interface:

```
/** Create a pop-up box to show the 6-digit password control.
 * title           Title
 * cancelButtonTitle Cancel the button title.
 * confirmButtonTitle Confirm the button title.
 */
- (id)initWithTitle:(NSString*)title
    cancelButtonTitle:(NSString*)cancelButtonTitle
    confirmButtonTitle:(NSString*)confirmButtonTitle;

/** Create a pop-up box to show the 6-digit password control.
 * title           Title
 * subtitleView     Subtitle
 * cancelButtonTitle Cancel the button title.
 * confirmButtonTitle Confirm the button title.
 */
- (id)initWithTitle:(NSString*)title
    subtitleView:(UIView *)subtitleView
    cancelButtonTitle:(NSString*)cancelButtonTitle
    confirmButtonTitle:(NSString*)confirmButtonTitle;

/** If no target or selector is set, the default execution action is :
 - (void)addLeftBackButtonTarget:(id)target selector:(SEL)selector even
 */

/** Get the fixed width of subtitle */
+ (CGFloat )subtitleViewWidth;

/** 
 *Self-defined view to show the subtitle.
 *This view will show the subtitles that any business needs to define
 *If the subtitle has been set, a divider line will be shown between +
 */
- (void)showSubtitleView:(UIView *)subtitleView;

/*
Unlike UIAlertView, no automatic dismissing will be conducted after I
*/
- (void)dismiss;
```

```
- (void)dismissWithCompletionBlock:(void (^)(void))block;
- (void)clearBlock;
```

Example:

```
- (void)showALPNumPwdInputView:(NSInteger)tag {

    _pwdPopup = [[APNumPwdPopupView alloc] initWithTitle:@"Enter password"];
    self.alertViewTag = tag;

    __weak ALPNumPwdPopupView * weekpop = _pwdPopup;
    __weak SWPhonePasswordViewController * weekvc = self;

    _pwdPopup.cancelBlock = ^(ALPNumPwdPopupView * sender, int buttonIndex) {
        [weekpop dismiss];
        [weekvc reloadData];
    };

    _pwdPopup.confirmBlock = ^(ALPNumPwdPopupView * sender, int buttonIndex) {
        [MBProgressHUD showHUDAddedTo:weekpop animated:YES];
        [weekvc rpcForModifyGesture:sender.pwdInputView.password];
    };

    [_pwdPopup presentWithinWindow:DTContextGet().window];
}
```

APPickerView

Class name: APPickerView, inherited from UIView.

Description: It encapsulates compound widgets with "Cancel" and "Complete" buttons on the UIPickerView widget.

Interface:

```
/*
 * Create components.
 * @param title Title. Its value can be nil.
 * @return The created component, not shown by default. The show method must be called to show it.
 */
+ (APPickerView *)pickerViewWithTitle:(NSString *)title;

/*
 * Initialize objects.
 * @param frame Display position.
 * @param title Show the title. Set the value to nil if you do not want to show the title.
 * @return Do not show the object by default. To show the object, call the show method.
 */
- (id)initWithFrame:(CGRect)frame initWithTitle:(NSString *)title;

/*
 * Show
 */
- (void)show;

/*
 * Hide
 */
- (void)hide;

/**
 * Reload data.
 */
- (void)reload;
```

Property:

```
/** UIPickerView widget. Used to set its dataSource and delegate properties */
@property(nonatomic, strong) UIPickerView *pickerView;

/** 
 * Set APPickerDelegate. The delegate must implement the delegate functions.
 * Callback when clicking to pick messages.
 *- (void)cancelPickerView:(APPickerView *)pickerView;
 * Callback when clicking to complete. The selected items can be returned.
 *- (void)selectedPickerView:(APPickerView *)pickerView;
*/
@property(nonatomic, weak) id<APPickerDelegate> delegate;
```

Example:

```
_pickView = [APPickerView pickerViewWithTitle:nil];
_pickView.delegate = self;
_pickView.pickerView.dataSource = self;
[self.view addSubview:_pickView];
// Set the selected position.
[_pickView.pickerView selectRow:0 inComponent:0 animated:YES];
[_pickView show];
```

Demo:



APSegmentedControl

Class name: APSegmentedControl, inherited from UISegmentedControl.

Description: It specifies the segmenting widget frame as CGRectMake (0, 0, 320.0, 44.0).

Interface:

```
// Create and initialize the internal frame to CGRectMake (0, 0, 320.0, 44.0)
- (id)initWithItems:(NSArray *)items;
```

APSmsAuthCodeBox

Class name: APSmsAuthCodeBox, inherited from APAuthCodeBox.

Description: The input field of text verification code.

Interface:

```
/**  
 * Create the input field of text verification code.  
 * @param originY The y-coordinate of the component.  
 * @param interval The wait time before the text message is sent.  
 * @return The input box of text verification code.  
 */  
- (APSmsAuthCodeBox *)initWithOriginY:(CGFloat)originY  
                           Interval:(NSTimeInterval)interval;
```

Property:

```
// Valid time.  
@property (assign, nonatomic) NSTimeInterval interval;  
// Start time. Internally the current time will be used to assign value.  
// So this property is not used when the text message verification window  
@property (assign, nonatomic) NSTimeInterval startTime;
```

Example:

```
CGRect frame = CGRectMake(10, originY, 300, 49);  
APSmsAuthCodeBox *box = [[APSmsAuthCodeBox alloc] initWithFrame:frame];  
box.interval = interval;  
box.inputBox.textField.keyboardType = UIKeyboardTypeDecimalPad;  
box.inputBox.textField.placeholder = @"Text verification code.";
```

APTextField

Class name: APTextField, inherited from UITextField.

Description: The input box widget which enriches the functions of APTextInputValidator.

Interface:

```
/**  
 * Create input field components.  
 * @param style The text box style.  
 * @param originY The y-coordinate of the input field.  
 * @return The input field components.  
 */  
+ (APTextField *)textFieldWithStyle:(APTextFieldStyle)style originY:(  
  
/**  
 * Initializes and returns an newly created text field object with th  
 *  
 * @return An initialized text field object.  
 */  
- (id)init;  
  
/**  
 * Check the input validity.  
 * @return The validation result.  
 */  
- (BOOL)checkInputValidity;  
  
/**  
 * Format texts.  
 * @return The formatted texts.  
 */  
- (NSString *)formatText:(NSString *)text;  
  
/**  
 * The original texts before formatting.  
 * @param textField Text box.  
 * @return The texts before formatting.  
 */  
- (NSString *)getOriginTextWithFormatText:(NSString *)formatText;
```

Property:

```

@property(strong, nonatomic) APTextValidator *validator;
@property(strong, nonatomic) NSString *normalizedText;
@property(nonatomic, strong) APTextFieldDelegateProxy *delegate;
//The two properties below are not recommended any more, because the API
//is deprecated.
@property(strong, nonatomic) UIButton *buttonX;
@property(assign, nonatomic) BOOL keyboardTypeIDCard;

```

```

typedef NS_ENUM(NSInteger, APTextFieldStyle){
    APTextFieldStylePlain,
    APTextFieldStyleBordered,
};

typedef NS_ENUM(NSInteger, APKeyboardType){
    APKeyboardTypeIDCard = 100,
}

```

Example:

```

_textField = [[APTextField alloc] initWithFrame:UIEdgeInsetsInsetRect(
    self.bounds, UIEdgeInsetsZero)];
_textField.validator = [APTextValidator mobileNumberValidator];

```

APTipView

Class name: APTipView, inherited from UIView.

Description: The tip view.

Interface:

Table of Contents |

```
/**  
 * Create a tip view with clickable button(s).  
 * @param frame      The position and size in the parent view.  
 * @param tip        The tip text.  
 * @param title      Button title.  
 * @return The object of the tip view with clickable button(s).  
 */  
- (id)initWithFrame:(CGRect)frame tipMessage:(NSString *)tip title:(NS  
  
/**  
 * Create a tip view.  
 * @param frame      The position and size in the parent view.  
 * @param tip        The tip text.  
 * @return The tip view object.  
 */  
- (id)initWithFrame:(CGRect)frame tipMessage:(NSString *)tip;
```

Property:

```
/** APTipViewDelegate. It must implement the delegate methods of click  
 * - (void)tipButtonClick:(UIButton *)sender;  
 */  
@property (nonatomic, weak) id <APTipViewDelegate> delegate;
```

Example:

```
self.tipView = [[APTipView alloc] initWithFrame:self.view.frame tipMe:  
[self.view addSubview:self.tipView];
```

Demo:



APToastView

Class name: APToastView, inherited from UIView.

Description: Toast message.

Interface:

```
/**
 * Show toast.
 *
 * @param superview The view in which the toast will be shown.
 * @param icon The icon type. The following types are supported:
 *   APToastIconNone No icon.
 *   APToastIconSuccess Success icon.
 *   APToastIconFailure Failure icon.
 * @param text Text.
 * @param duration Duration.
 *
 */
+ (void)presentToastWithin:(UIView *)superview withIcon:(APToastIcon):
```



```
 /**
 * @param superview The view in which the toast will be shown.
```

Table of Contents |

```
* @param text Text.
* @return The toast object shown.
*/
+ (APToastView *)presentToastWithin:(UIView *)superview text:(NSString *)text;

/*
 * Modality view prompt. In this mode the screen will not respond to touch events.
 */
+ (APToastView *)presentToastWithText:(NSString *)text;

/*
 * Modality toast
 * @param superview Parent view.
*/
+ (APToastView *)presentModelToastWithin:(UIView *)superview text:(NSString *)text;

/*
 * Make the toast disappear.
*/
- (void)dismissToast;

/***
 * Show toast.
 *
 * @param superview The view in which the toast will be shown.
 * @param icon The icon type. The following types are supported:
 *   APTToastIconNone No icon.
 *   APTToastIconSuccess Success icon.
 *   APTToastIconFailure Failure icon.
 * @param text Text.
 * @param duration Duration.
 * @param completion Callback after the toast disappears automatically.
 */
+ (void)presentToastWithin:(UIView *)superview withIcon:(APToastIcon):icon
    text:(NSString *)text
    duration:(NSTimeInterval)duration
    completion:(void (^)(void))completion;

/***
 * Show modality toast.
*/
```

```
*  
* @param superview The view in which the toast will be shown.  
* @param icon The icon type. The following types are supported:  
*   APTToastIconNone No icon.  
*   APTToastIconSuccess Success icon.  
*   APTToastIconFailure Failure icon.  
* @param text Text.  
* @param duration Duration.  
* @param completion Callback after the toast disappears automatically.  
*  
*/  
+ (void)presentModalToastWithin:(UIView *)superview withIcon:(APToastIcon)icon  
    text:(NSString *)text duration:(NSTimeInterval)duration completion:(void (^)())completion
```

Example:

```
[APToastView presentToastWithin:weakSelf.view withIcon:ALPTToastIconNone  
    text:@"该卡暂不能开通快捷支付,请用其他卡" duration:3.0 completion:nil]
```

Demo:



该卡暂不能开通快捷支付,请用其他卡

APRefreshTableHeaderView

Description: The pull-to-refresh widget encapsulated based on open-source EGOResfreshTableHeaderView.

APTableViewCell

Interface:

```
/**  
 * Constructor. The derived class calls the base class constructor.  
 */  
- (id)initWithStyle:(UITableViewCellStyle)style reuseIdentifier:(NSString *)  
    reuseIdentifier  
    forIndexPath:(NSIndexPath *)indexPath
```

Property:

```
typedef NS_ENUM(NSUInteger, APTableViewCellStyle) {  
    /** Simple cell with text label and optional image view. */  
    APTableViewCellStyleSimple,  
    APTableViewCellStyleSubtitle = UITableViewCellStyleSubtitle,  
};  
  
/**  
 * Default UIEdgeInsetsZero. Add additional padding area around content.  
 */  
@property(nonatomic) UIEdgeInsets contentInset;
```

APTableViewCellBaseCell



1.1 Class name: APTableViewBaseCell

1.2 Inheritance: 1.3 Applicable scope: It meets the needs in most cases of single-row cells.

1. Operation and action: 3.1 Interface:

```
+ (float)cellHeight;

/**
 * Reset all the child controls. This method should be called for reuse.
 */
- (void)reset;

/**
 * Set the avatar icon.

```

Table of Contents |

```
/*
- (void)setLogoImg:(UIImage *)img;

/***
 * Attention: The SDWebImage.framework should be imported for this method.
 *
 * @param imgUrl      Image URL.
 * @param defaultImg Default image UIImage instance.
 */
- (void)setLogoImgWithUrl:(NSString *)imgUrl withDefault:(UIImage *)defaultImg;

/***
 * Set the title.
 */
- (void)setTitle:(NSString* )title;

/***
 * Set whether to show the "Extend" icon.
 */
- (void)setExtendLogo;

/***
 * Set the default background color.
 */
- (void)setNormalBackground:(UIColor *)normalColor;

/***
 * Set the background color in selected status.
 */
- (void)setSelectedBackground:(UIColor *)selectedColor;

/***
 * Set the prompt message. It provides additional information to the user about the
 */
- (void)setInfo:(NSString *)info;

/***
 * Set the font color of the prompt message.
*/
```

```
/*
- (void)setInfoFontColor:(UIColor *)color;

/**
 * Design the prompt icon. The position is the same as above. The pro-
 */
- (void)setInfoImg:(UIImage *)img;

/**
 * Set switches.
 * @param isOpen Default switch settings.
 * @param target Responding object of switch events.
 * @param selector Responding method of switch events.
*/
- (void)setSwitchWithDefault:(BOOL) isOpen withTarget:(id) target with-
    selector:(SEL)selector;

/**
 * Set specific details. For the effect not existing in this class, R-
 */
- (void)setInfoDetail:(NSString *)detail;

- (BOOL)isShowExtend;

- (void)addView:(UIView *)subView;

- (void)enableLineImageView:(BOOL)yesOrNo;

- (NSInteger )contentViewRightGap;
```

2.2 Callback event: 2.3 Others: Attention: To use `(void)setLogoImgWithURL:(NSString)imgUrl withDefault:(UIImage)defaultImg`, the SDWebImage.framework needs to be imported for the interface to take effect. Set the avatar to a specified URL, and set the default avatar. The defaultImg argument cannot be empty.

1. Example:

```
- (UITableViewCell*) tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    APTableViewBaseCell* cell = [tableView dequeueReusableCellWithIdentifier:@"APButtonCell"];
    if (cell) {
        [cell reset];
    }
    else {
        cell = [[APTableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"APButtonCell"];
        [cell setTitle:@"Alipay"];
        UIImage *img = [UIImage imageNamed:@"icon_58.png"];
        [cell setInfo:@"alipay"];
        [cell setLogoImg:img];
    }
    return cell;
}
```

APButtonCell

Class name: APButtonCell, inherited from UITableViewCell.

Description: Tables with buttons.

Interface:

```
/**  
 * Method to add buttons in tables.  
 */  
- (void)addButton:(APButton *)button;  
  
/**  
 * Add a type of button of the click event in the table.  
 */  
- (APButton *)addButtonType:(APButtonType)buttonType title:(NSString *)title  
    target:(id)target selector:(SEL)selector;
```

Example:

```
APButtonCell *buttonCell = ??[APButtonCell alloc] initWithFrame:CGRectMake(0, 0, 100, 100);  
APButton *button = ??[APButton alloc] initWithFrame:CGRectMake(0, 0, 100, 100);  
[button setTitle:@"OK"];  
?[buttonCell addButton:button];  
[buttonCell addButtonWithType:APButtonTypeCustom title:@"OK" target:self selector:@selector(buttonClicked)];
```

APIInputBoxCell

Class name: APIInputBoxCell, inherited from UITabbleViewCell.

Description: Tables with buttons.

Interface:

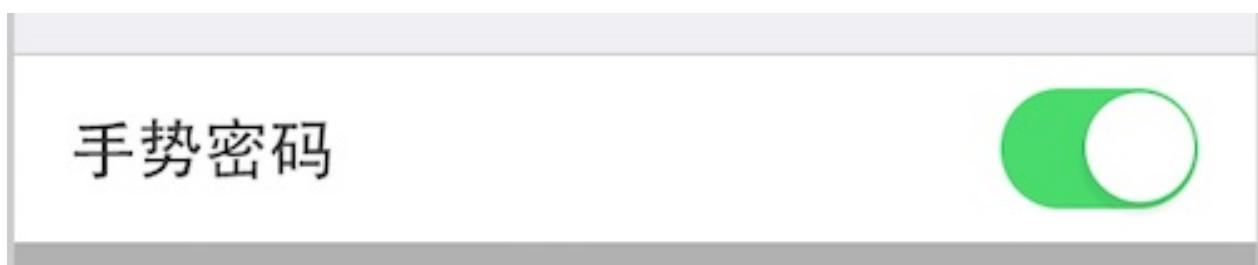
```
/**  
 * Widget height.  
 */  
+ (float)cellHeight;  
  
/**  
 * Return the input box in the cell.  
 */  
- (APIInputBox *)textFieldInCell;
```

Example:

```
APIInputBoxCell *inputBoxCell = ??[APIInputBox alloc] initWithFrame:UIT:  
APIInputBox *box = [inputBoxCell textFieldInCell];
```

APSwitchCell

1. Description of the widget:
 - 1.1 Class name: APSwitchCell
 - 1.2 Inheritance: 1.3 Applicable scope: It is used for settings.
1. Interface property 2.1 Effect:



1. Operation and action: 3.1 Interface:

```

    /**
     * Return the cell height.
     *
     * @return The height value.
     */
+ (float)cellHeight;

    /**
     * Initialization
     *
     * @param reuseIdentifier The reuse ID of the cell.
     *
     * @return APSwitchCell
     */
- (id)initWithReuseIdentifier:(NSString *)reuseIdentifier;

    /**
     * UISwitch widget
     */
@property (nonatomic, readonly) UISwitch *switchControl;

```

3.2 Callback event: 3.3 Others:

1. Example:

```

APSwitchCell *cell = nil;
identifier = @"switch";
cell = [_tableView dequeueReusableCellWithIdentifier:identifier];
if (!cell) {
    cell = [[APSwitchCell alloc] initWithReuseIdentifier:identifier];
    cell.textLabel.text = @"Gesture password";
    cell.selectionStyle = UITableViewCellStyleNone;
    _tableView.separatorStyle = UITableViewCellStyleSingleLine;
}

```

APTableViewDoubleLineCell

Class name: APTableViewDoubleLineCell, inherited from APTableViewCell.

Description: The cell with double lines.

Interface:

```
/**  
 * APTableViewDoubleLineCell The initialization function. The set style  
 * We recommend this interface for initialization.  
 * @param reuseIdentifier The reuse identifier.  
 * @return The initialized instance.  
 */  
- (id)initWithReuseIdentifier:(NSString *)reuseIdentifier;  
  
/**  
 * Return the cell height.  
 * @return The cell height.  
 */  
+ (float)cellHeight;  
  
/**  
 * Cell reset */  
- (void)reset;
```

Example:

```
APITableViewDoubleLineCell *cell = nil;  
identifier = @"DoubleLineCell";  
cell = [tableView dequeueReusableCellWithIdentifier:identifier];  
if (!cell) {  
    cell = [APITableViewDoubleLineCell alloc initWithReuseIdentifier:ident:  
    cell.selectionStyle = UITableViewCellSelectionStyleNone;  
}
```

APITableViewTwoTextCell

1.1 Class name: APITableViewTwoTextCell

1.2 Inheritance: Parent class: APITableViewCell

1.3 Applicable scope The cell with two text tags on the left and right respectively.

The two tags will be aligned automatically according to the text length.

1. Interface property 2.1 Effect:



卡类型

招商银行 储蓄卡

1. Example:

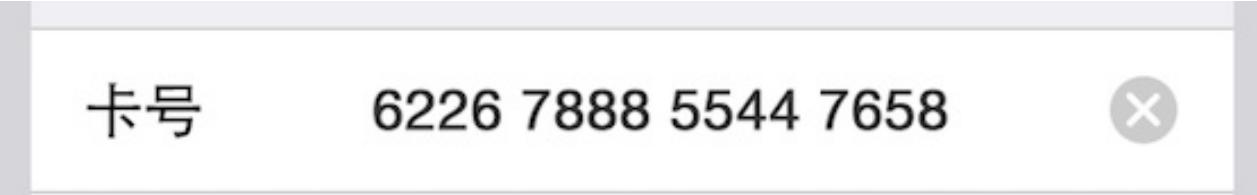
```
cell = [[APTableViewCellTwoTextCell alloc]  
initWithStyle:UITableViewCellStyleDefault  
reuseIdentifier:cellId  
leftText:@"Text on the left"  
rightText:@"Text on the right"]];
```

APTextFieldCell

1.1 Class name: APTextFieldCell

1.2 Applicable scope: It is used for input boxes in the cell.

1.3 Effect:



卡号 6226 7888 5544 7658

1.4 Interface:

```
@property(nonatomic, strong, readonly) APTextField *textField;  
@property(nonatomic, assign) CGFloat textLabelWidth;
```

@cnName Description File @priority 1

Description File

@cnName Custom Theme @priority 2

Custom Theme

@cnName APNavigationBar @priority 1

APNavigationBar

@cnName 3 IconFont @priority 3

3 IconFont

[TOC]

3.1 Introduction

iOS applications will carry many PNG icon resources. You need to package multiple sets of resources for different screen sizes. To reduce the icon resources and improve display effects, IconFont is based on vectorgraphs and offers alternative solutions for some icon resources. It can effectively downsize the install package.

The source files of IconFont are font files in the ttf format. The ttf file describes the vector information of each icon. In usage, mPaaS will load the ttf fonts and the business-level codes will be able to draw the icon.

More resources can be found and downloaded on the home page of IconFont: <http://iconfont.cn/>

3.2 How to use

3.2.1 Add IconFont

3.2.1.1 Description file

After the MPCommonUI module is imported to mPaaS, you need to add APCommonUI.bundle to the project. Under the bundle, there is an 'ap_iconfont_name.plist' for configuring the IconFont.

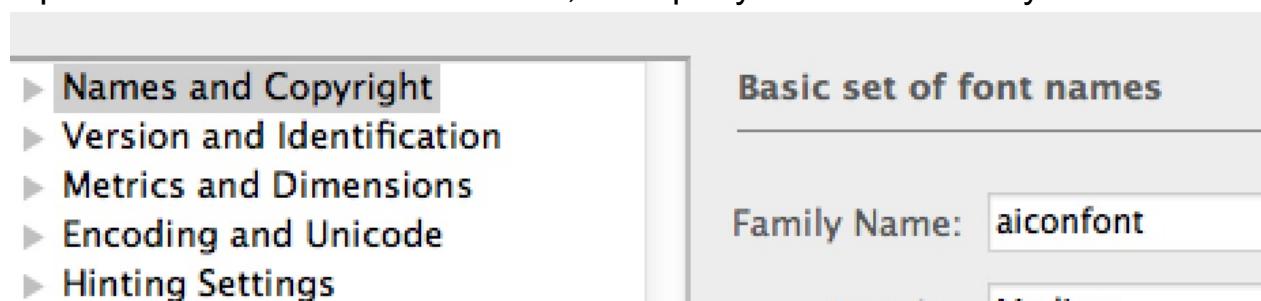
Each resource contains two fields: the font file and the configuration file, as shown in the figure:

Key	Type	Value
Root	Dictionary	(1 item)
iconfont	Dictionary	(2 items)
file	String	iconfont
config	String	iconfontconfig

The resource name is the 'Family Name' of the font. The 'file' indicates the location of the ttf font file relative to 'main bundle', and the 'config' indicates the location of the description file relative to 'main bundle'. Note that both of them do not require the extension.

3.2.1.2 Viewing family name of font

Open the ttf file in FontLab Studio, and query the font's 'Family Name'.



3.2.2 IconFont interface

The text value for drawing the IconFont is an 8-bit Unicode character. mPaaS provides UIView extensions to display IconFont conveniently.

```
@interface APIIconFont : NSObject

//Use default ttf (the first item in the configuration file)
+ (UIFont *)iconFont;
+ (UIFont *)iconFontWithSize:(NSInteger)fontSize;

+ (UIFont *)iconFontWithFontName:(NSString *)fontName;
+ (UIFont *)iconFontWithFontName:(NSString *)fontName size:(NSInteger)fontSize;

@end

@interface UIView (APIIconFont)

//Use default ttf (the first item in the configuration file)
+ (UIView *)iconFontViewWithPoint:(CGPoint)point name:(NSString *)name;
+ (UIView *)iconFontViewWithPoint:(CGPoint)point name:(NSString *)name size:(NSInteger)fontSize;

+ (UIView *)iconFontViewWithPoint:(CGPoint)point name:(NSString *)name fontName:(NSString *)fontName;
+ (UIView *)iconFontViewWithPoint:(CGPoint)point name:(NSString *)name fontName:(NSString *)fontName size:(NSInteger)fontSize;

- (BOOL)setIconFontColor:(UIColor *)color;
- (BOOL)setIconFontName:(NSString *)name;

@end
```

3.2.3 Drawing IconFont

```
UILabel * label = [[UILabel alloc] init];
label.text = @"\U00003439";
label.font = [APIIconFont iconFont];
[self.view addSubview:label];

UIView *fontView = [UIView iconFontViewWithPoint:CGPointMake(100, 100)
//Set icon position and size dynamically with the frame.
fontView.frame = CGRectMake(100, 100, 48, 48);
[self.view addSubview:fontView];

fontView = [UIView iconFontViewWithPoint:CGPointMake(130, 100) name:@"yuebao"]
//Reset color.
[fontView setIconFontColor:[UIColor redColor]];
//Reset image.
[fontView setIconFontName:@"yuebao"];
[self.view addSubview:fontView];
```

@cnName SonicWaveNFC Component @priority 8

SonicWaveNFC Component

[TOC]

1 Usage and function

The sonic-wave near field communication technology enables terminals with speakers and microphones to achieve data communication with the sonic wave.

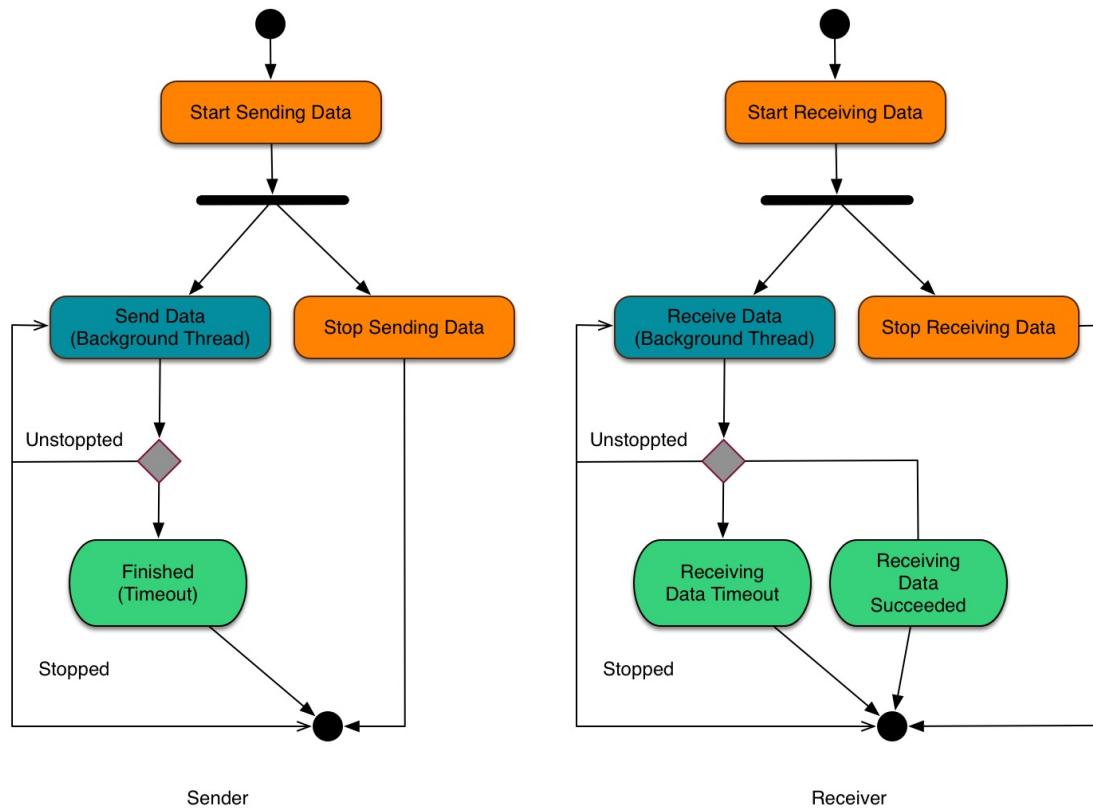
The technology does not require modification to the hardware. It is fully dependent on the software, featuring a low deployment cost, and is thus capable of rapid promotion. Its applicable scenarios include but are not limited to the following:

- Establishing dialogs between mobile phones to achieve face-to-face payment and collection.
- Establishing dialogs between mobile phones and cash register PCs (or tablets) to achieve face-to-face payment and collection.
- Establishing dialogs between mobile phones and PCs to pay for online goods on the mobile phones.
- Establishing dialogs between mobile phones and check-in PCs (or check-in modules) of the business for precise check-ins.
- Bluetooth pairing between mobile phones.
- Establishing dialogs between mobile phones to exchange name cards, photos, music files, etc.

SonicWaveNFC SDK (SDK for short hereinafter) is a tool for developing applications of the sonic-wave near field communication technology. It provides the best features for developing innovative mobile Internet applications and solutions.

2 Communication means

The application adopts a 'half-duplex' means for communication which allows two-way data transmission between two terminals. The two terminals cannot send data at the same time. When one terminal is sending data, the other can only wait until the first terminal finishes the sending before the second terminal is able to send data. The figure below illustrates the processing procedure of the application during a communication:



The sender is terminal using the speaker to send acoustic signals, and the recipient is the terminal using the microphone to record the acoustic signals.

The procedure of the sender is as follows:

1. The application system calls the method to start sending data;
2. The application will process data sending in the background thread in loop execution;
3. The application system calls the method to stop sending data in the event of a timeout or other application events;
4. The application receives the instruction for stopping sending data in the background thread, and stops execution;
5. The data transmission process is stopped.

The procedure of the receiver is as follows:

1. The application system calls the method to start receiving data;
2. The application will process data receiving in the background thread in loop execution;
3. If the data is successfully received when the application is executing the data receiving procedure, the execution will stop. The application invokes an event callback for successful data receiving, and the data receiving process is ended;
4. If the data receiving action times out when the application is executing the data receiving procedure, the execution will stop. The application invokes an event callback for data receiving timeout, and the data receiving process is ended;
5. The application system calls the method to stop receiving data in the event of other application events;
6. The application receives the instruction for stopping receiving data in the background thread, and stops execution;
7. The data receiving procedure is ended.

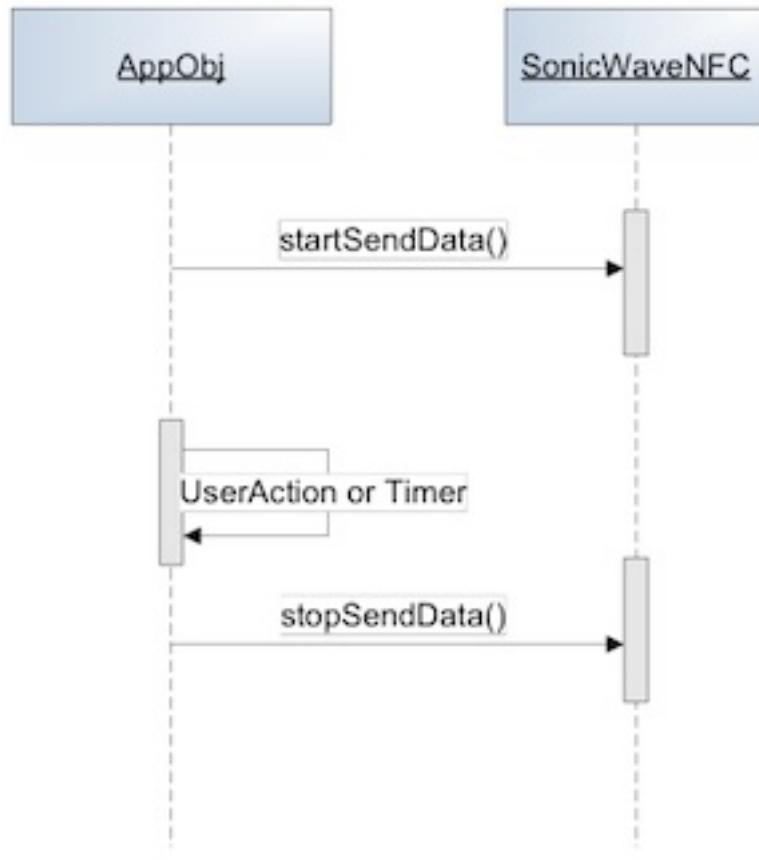
3 How to use

The sonic-wave communication component is MPSonicWaveNFC. Check it during packaging for generating the mPaas.framework. You also need to add mPaas.bundle to the project as it contains the default sound effect files.



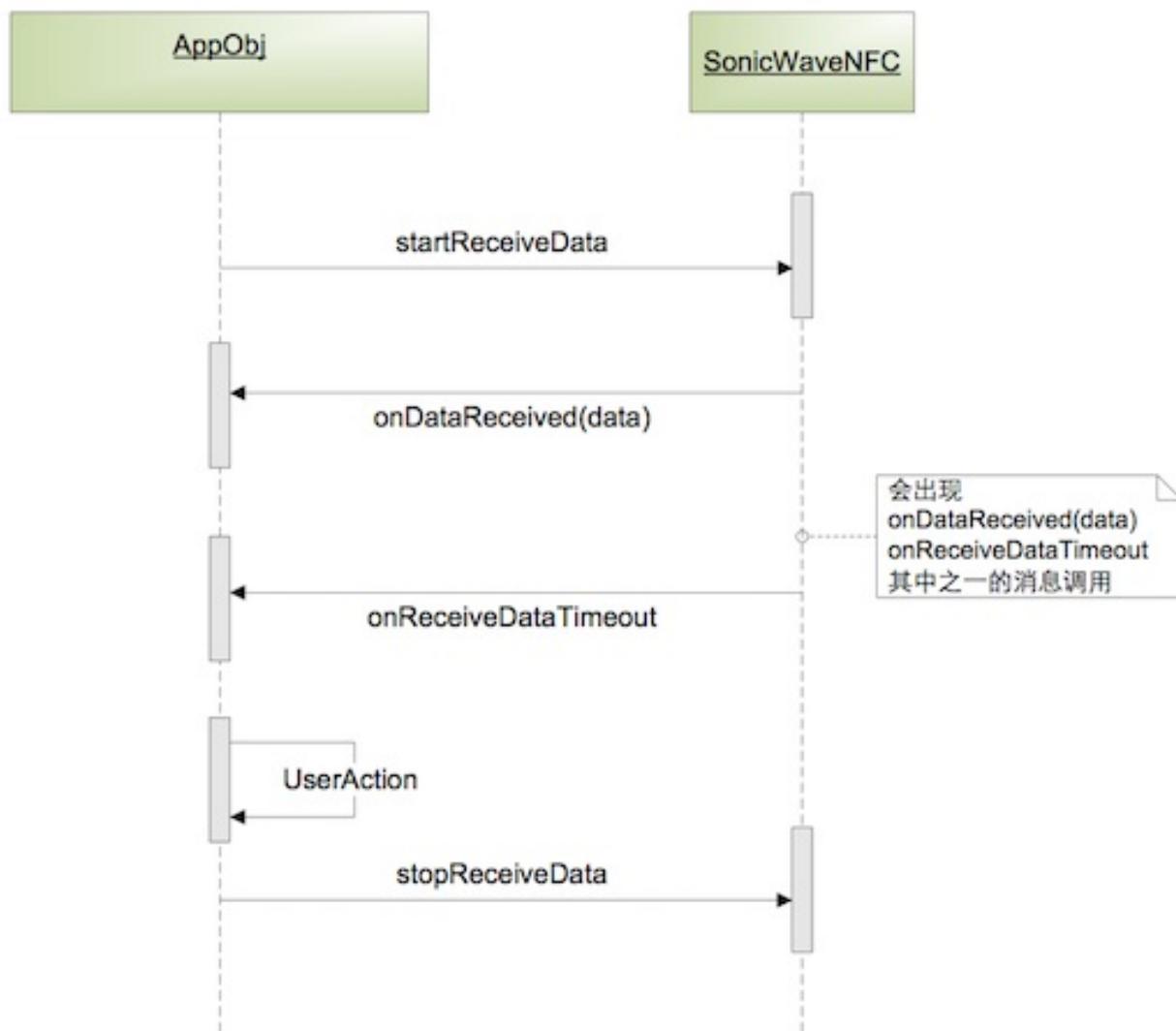
3.1 Sending data

Among them, the AppObj is the instance of a container class of the application system or a page object.



3.2 Receiving data

Among them, the AppObj is the instance of a container class of the application system or a page object.



4 Interface description

4.1 SonicWaveNFC

hasHeadset

Description: It judges whether the headset has been inserted.

Return: Whether the headset has been inserted.

startSendData

Description: Starts to send data.

Argument:

strData The data to be sent, 64 in number at most. When this value is

iTimeoutSeconds The timeout seconds for data sending. Recommended value: 10.

iSoundType The sound type when data is being sent.

NFC_SENDDATA_SOUNDTYPE_SILENT 0 Silent

NFC_SENDDATA_SOUNDTYPE_NOISY 1 Sound

NFC_SENDDATA_SOUNDTYPE_MIX 2 Mixed background prompt tones

iVolume Sound volume for data sending. Valid value: 0-100. Recommended value: 50.

handler The callback handler of SonicWaveNFC and the implementation of the `onSendData` event.

Return: Whether the start of data sending is normal.

Prompt: When the data sent are all digits or lower-case letters, the function returns true; otherwise, it returns false.

stopSendData

Description: Stops sending data.

setBkSoundWave4MixFromRes

Description: Sets the mixed background prompt tones used.

Argument:

resName Resource name. See the resName argument of pathForResource.

ofType Resource type (extension). See the ofType argument of pathForResource.

Return: Whether the settings are successful.

startReceiveData

Description: Starts to receive data.

Argument:

iTimeoutSeconds The timeout seconds for receiving data.

iMinAmplitude The minimal threshold value of amplitude for receiving data.

handler The callback handler of SonicWaveNFC and the implementation of the interface.

Return: Whether the start of data receiving is normal.

stopReceiveData

Description: Stops receiving data.

4.2 SonicWaveNFCHandler

onSendDataStarted

Description: Starts data sending normally.

onSendDataTimeout

Description: Data sending times out.

onSendDataFailed

Description: Data sending failed.

Argument:

iErrorCode The reason code for the failure.

onReceiveDataStarted

Description: Starts data receiving normally.

onDataReceived

Description: Data received.

Argument:

strData Data received.

onReceiveDataTimeout

Description: Data receiving times out.

onReceiveDataFailed

```Description: Data receiving failed.

Argument: iErrorCode The reason code for the failure.

```
onHeadsetOn
```

Description: The headset is inserted while data sending or receiving is in process.

```
onHeadsetOff
```

Description: The headset is pulled out while data sending or receiving is in process.

```

Multilingual Component

[TOC]

1 Introduction

MPLanguageSetting is used for switching the language environments in the application. It is independent from the system language settings. It has the following features:

1. It switches the constant character languages automatically and the caller does not need to care about it.
2. For parameterized strings, it offers callback methods to the caller.
3. With further special demands, you can monitor the notifications issued by mPaaS at language switching.

2 How to use

2.1 Switching monolingual application to monolingual one

1. Import mPaas.framework and MPLanguage.bundle with the resource files into the project. This bundle is located in the mPaas.framework directory and should be copied from the application.
2. Use NSLocalizedString macro for compiling the strings to be localized in the codes so that the .strings file can be generated, for example:

```
// The recommended key value should be in the "module.content" format

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    CustomTableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"CustomCell"];
    cell.titleLabel.text = NSLocalizedString(@"app1.table.cell.title");
    cell subTitleLabel.text = NSLocalizedString(@"app1.table.cell.subTitle");
    cell.textField.placeholder = NSLocalizedString(@"app1.table.cell.textFieldPlaceholder");
    return cell;
}
```

Extract texts to be localized with the genstrings method, and generate the .strings file.

```
// Open the terminal, find the project root directory and execute:
find ./ -name "*.m" -print0 | xargs -0 genstrings -o .strings
```

3. After the execution is done, the Localizable.strings file will be generated in the current directory. The content is similar to the following:

```
/* comment here. */
"app1.table.cell.subTitle" = "app1.table.cell.subTitle";

/* comment here. */
"app1.table.cell.textField" = "app1.table.cell.textField";

/* comment here. */
"app1.table.cell.title" = "app1.table.cell.title";
```

4.Copy the content in Localizable.strings to the .strings file at MPLanguage.bundle/, and translate the content into the correct language. For example, copy the content in Localizable.strings to MPLanguage.bundle/zh_CN.strings (we will provide tools to help you with this step), and translate it into the Chinese language:

```
// Filename: zh_CN.strings

/* comment here. */
"app1.table.cell.subTitle" = "Subtitle";

/* comment here. */
"app1.table.cell.textfield" = "Click and enter";

/* comment here. */
"app1.table.cell.title" = "Title";
```

5.Evoke the language settings page. There are two ways to achieve this: (1) Using the startApp method. (2) Using the langauge settings VC provided by MPLanguageSetting.

(1) Using the startApp method:

Add 'MPLanguageSettingrAppDelegate' in MobileRuntime.plist and set the AppID.

LanguageSetting > LanguageSetting > Supporting Files > MobileRuntime.plist > No Selection		
Key	Type	Value
Root	Dictionary	(3 items)
Launcher	String	20000001
Services	Array	(1 item)
Applications	Array	(3 items)
Item 0	Dictionary	(3 items)
delegate	String	DemoAppDelegate
description	String	Demo
name	String	20000002
Item 1	Dictionary	(3 items)
delegate	String	LauncherAppDelegate
description	String	启动窗
name	String	20000001
Item 2	Dictionary	(3 items)
delegate	String	MPLanguageSettingrAppDelegate
description	String	LanguageSetting
name	String	20000003

Call the method at where you want to evoke the language settings page. [DTContextGet() startApplication:@"20000003" params:@{} animated:YES];

(2) Using the 'MPLanguageManageController':

```
MPLanguageManageController *settingVC = [[MPLanguageManageController alloc] init];
// push settingVC Or present settingVC
```

At this point, the project is already able to handle the language switching for constant strings.

3 Advanced

1. Get the text content in the current language setting environment:

```
//Use the macro __TEXT(key, comment) provided by MPLanguageSetting.h  
  
- (void)tabBarController:(UITabBarController *)tabBarController didChangeLanguageSetting:(NSDictionary *)info NS_REQUIRE_SUPER ;  
  
{  
    self.title = __TEXT(viewController.title,@ ""); // Use the view controller's title  
}
```

2. Implementation - (void)languageSettingDidChange:(NSDictionary *)info NS_REQUIRE_SUPER ; .

```
//The DTViewController class realizes the monitoring of notifications  
  
// DemoAppViewController.m  
  
- (void)languageSettingDidChange:(NSDictionary *)info  
{  
    [super languageSettingDidChange:info];  
    if (![info[APLanguageSettingInfoNewKey] isEqualToString:info[APLanguageSettingInfoOldKey]])  
        self.title = __TEXT(@"app.sub.app1", @"Sub application");  
        // update constraints  
        // update image resources , etc.  
    }  
    NSLog(@"%@", @"DemoAppVC");  
}
```

3. Monitor notifications on language switching:

Field	Implication	Remark
APLanguageSettingDidChangeNotification	Name of the notification on language switching	
APLanguageSettingInfoOldKey	The acquired key of the old language in userInfo in the notification	
APLanguageSettingInfoNewKey	The acquired key of the new language in userInfo in the notification	The new language name may be the same as the old one

@cnName Code Scan Component @priority 10

Code Scan Component

[TOC]

1 Introduction

MPSCodeGen module integrates the functions of scanning the bar code, QR code, credit card, etc. It offers quick and low-CPU-consuming code scan service. All the functions are covered in one interface, and the application can select the code scan methods they want to enable.

2 How to use

We recommend mPaaS client framework for creating a micro application for the code scan function. The framework interfaces are also recommended for starting the code scan services.

```
@interface ScanCodeDelegate () <BumpControllerDelegate>

@property(nonatomic, strong) UIViewController *rootController;

@end

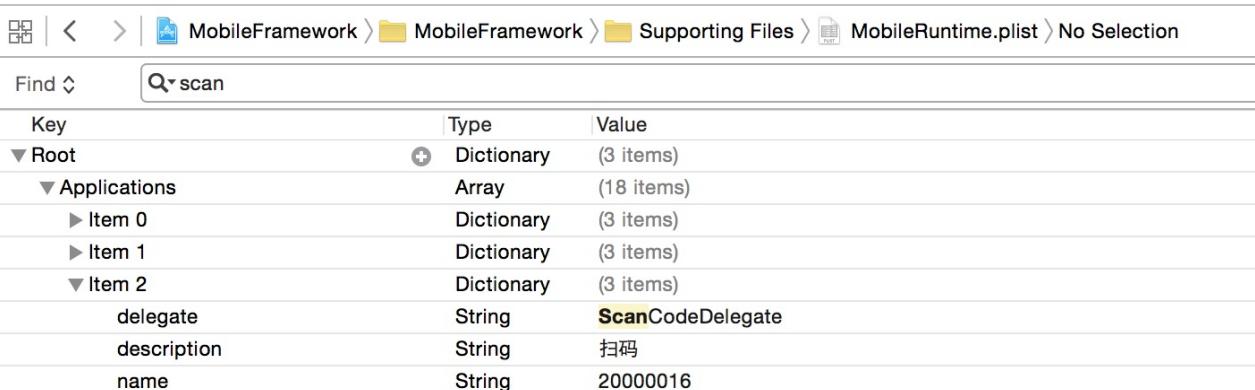
@implementation ScanCodeDelegate

- (UIViewController *)rootControllerInApplication:(DTMicroApplication *)
{
    return self.rootController;
}

- (void)application:(DTMicroApplication *)application willStartLaunch:
{
    self.rootController = [[ScanCodeViewController alloc] initWithNibName:
        ((ScanCodeViewController*)self.rootController).bumpDelegate = self;
}

@end
```

Configure MobileRuntime.plist.



The screenshot shows the Xcode plist editor interface. The navigation bar at the top displays the file path: MobileFramework > MobileFramework > Supporting Files > MobileRuntime.plist > No Selection. Below the navigation bar is a search bar with the text "scan". The main area is a table view showing the contents of the plist file:

Key	Type	Value
Root	Dictionary	(3 items)
Applications	Array	(18 items)
Item 0	Dictionary	(3 items)
Item 1	Dictionary	(3 items)
Item 2	Dictionary	(3 items)
delegate	String	ScanCodeDelegate
description	String	扫码
name	String	20000016

Start the scanning.

```
[DTContextGet() startApplication:@"20000016" params:nil animated:YES].
```

@cnName 1 Overview @priority 1

1 Overview

Mobile developer Tools include a suite of components, such as command line tools, Xcode plug-in, etc. to assist developers.

To install Developer Tools, please execute the following command on a terminal

```
sh
```

```
Last login: Wed Oct 14 20:57:17 on ttys000
shenmo — shenmo@shenmoMacBookPro: ~ — zsh — 99x44
sh <(curl -s http://code.taobao.org/svn/mpaaskit/trunk/install.sh)
mpaaskit_installer
* 欢迎使用mpaaskit (海纳平台iOS开发组件)
- 1. 执行环境检测
- 检查 git: git version 2.3.8 (Apple Git-58)
done.
- 2. 安装mpaaskit
- 最新版本为0.0.1
% Total    % Received % Xferd  Average Speed   Time   Time     Time  Current
          Dload  Upload   Total Spent  Left  Speed
100  3153  100  3153    0      0  5707      0 --:--:-- --:--:-- --:--:--  5711
/Users/shenmo
mpaaskit_install_path: /Users/shenmo/.mpaaskit
http://code.taobao.org/svn/mpaaskit/tags/0.0.1.zip
% Total    % Received % Xferd  Average Speed   Time   Time     Time  Current
          Dload  Upload   Total Spent  Left  Speed
100 10.4M  100 10.4M    0      0  552k      0  0:00:19  0:00:19 --:--:--  733k
- install mPaaS templates....
- install mPaaS plugins
```

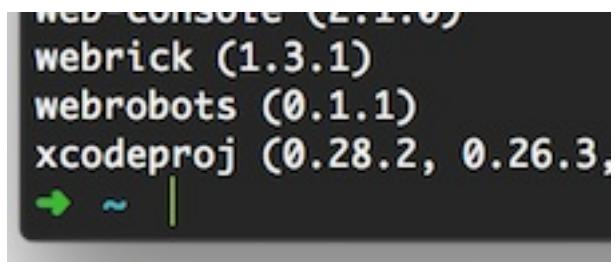
Please run the following command also to update Ruby Gems

```
sudo gem update
```

To edit the newest Xcode project files, please ensure xcodeproj is a relatively new version(the newest version is 0.28.2 in October, 2015).

Execute command:

```
gem list
```



```
web-console (2.1.0)
webrick (1.3.1)
webrobots (0.1.1)
xcdeproj (0.28.2, 0.26.3,
➔ ~ |
```

Besides, if you use the Ruby source provided by taobao.org, please make sure to use HTTPS rather than HTTP which is not supported any longer.

@cnName 2 Xcode Project Template @priority 2

2 Xcode Project Template

Xcode Project Template will be installed automatically along with the Developer Tools. Open New Project dialog from Xcode->File->New Project menu, then enter the project name to create a Xcode project based on mobile framework.

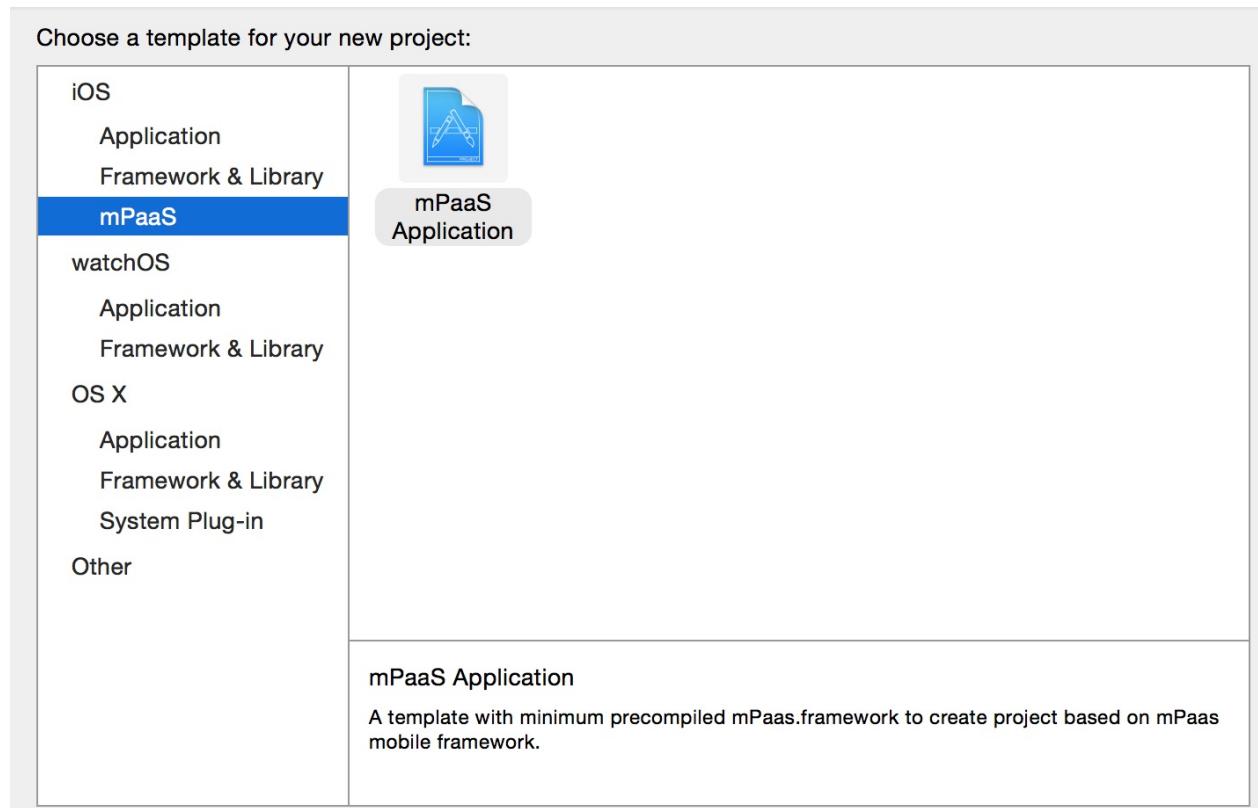


Table of Contents |

Choose options for your new project:

Product Name:

Organization Name:

Organization Identifier:

Bundle Identifier:

mPaaS Application Key:

[Cancel](#)

[Previous](#)

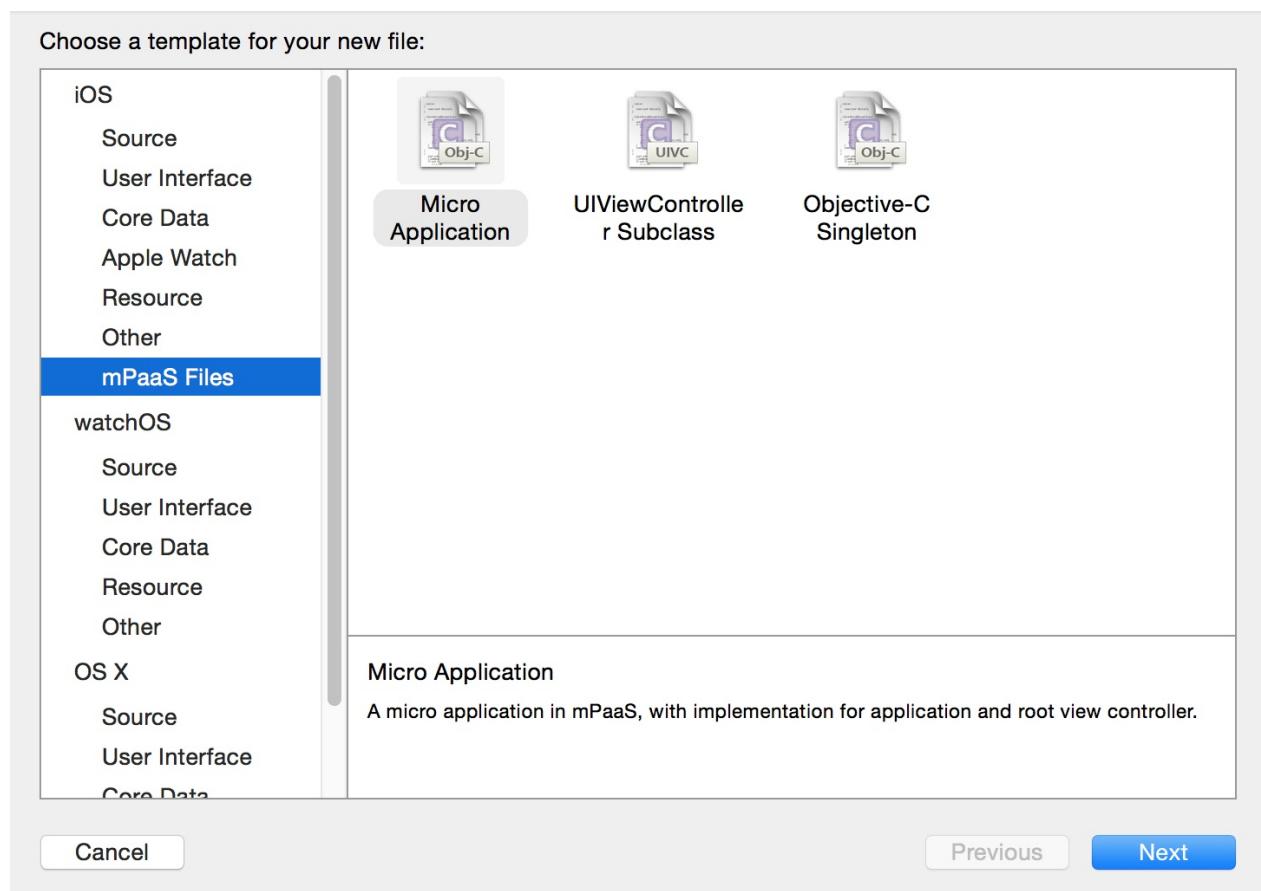
[Next](#)

@cnName 3 Xcode File Template @priority 3

3 Xcode File Template

Xcode File Template will be installed automatically along with the Developer Tools. Open New File dialog from Xcode->File->New File, then select a file template and enter the name. Currently three templates are provided as follows:

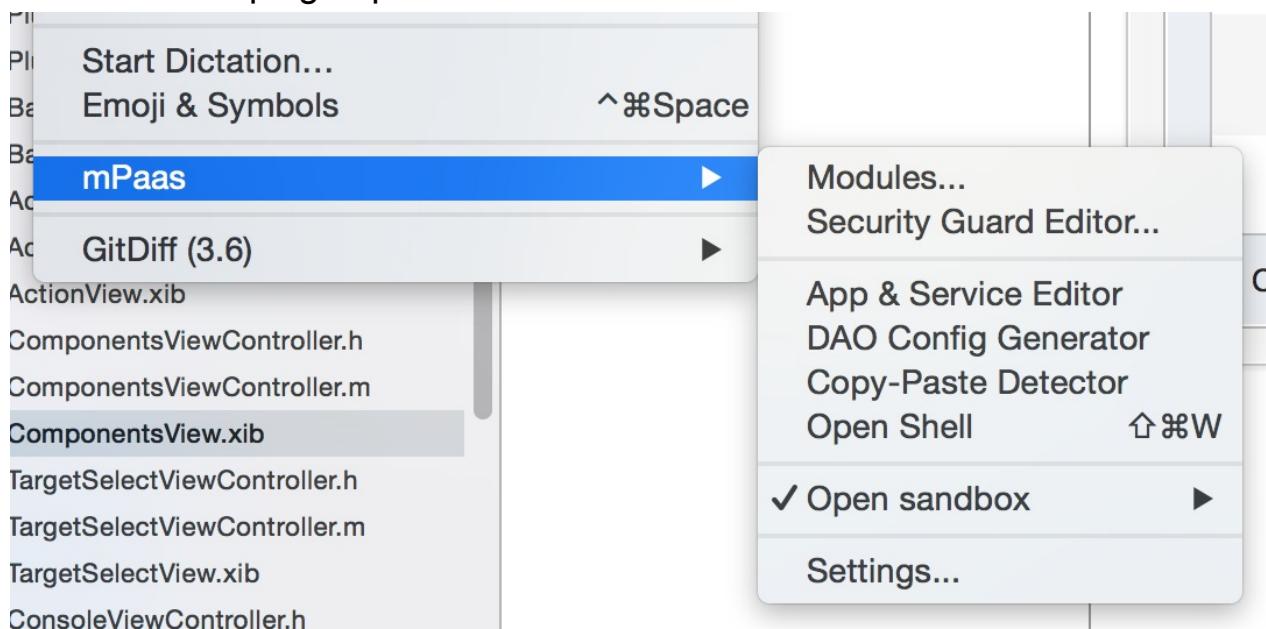
1. Micro application
2. UIViewController Subclass
3. Objective-C Singleton



@cnName 4 Xcode Plug-in @priority 4

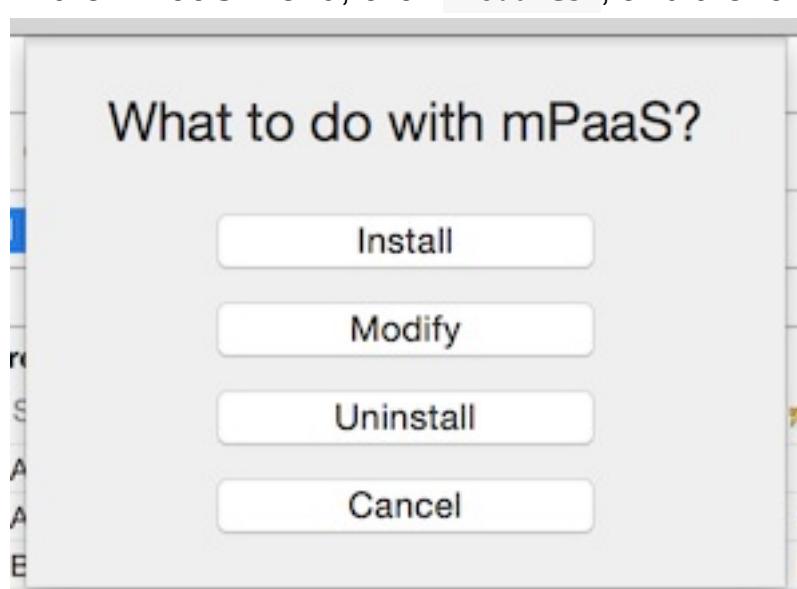
4 Xcode Plug-in

Developer tools contain Xcode Plug-in which appears in Xcode-Edit after installed. This plug-in provides functions as shown below.



4.1 mPaaS Modules Manager

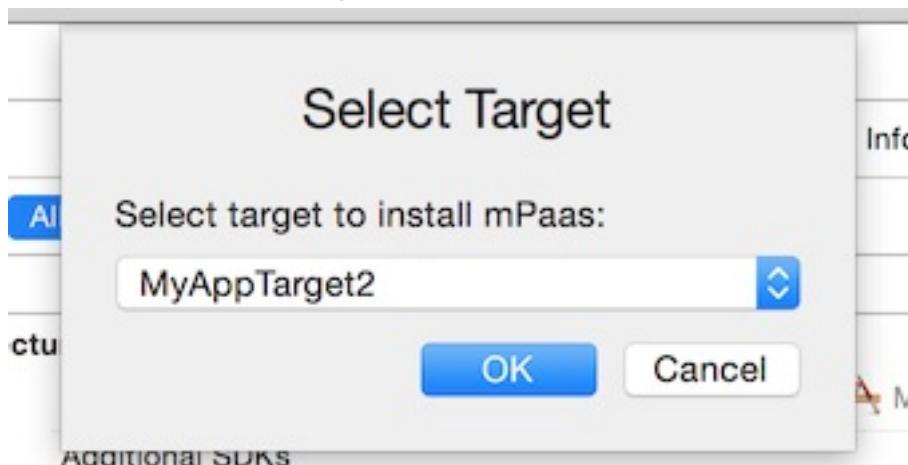
In the mPaaS menu, click `Modules`, and the following interface appears.



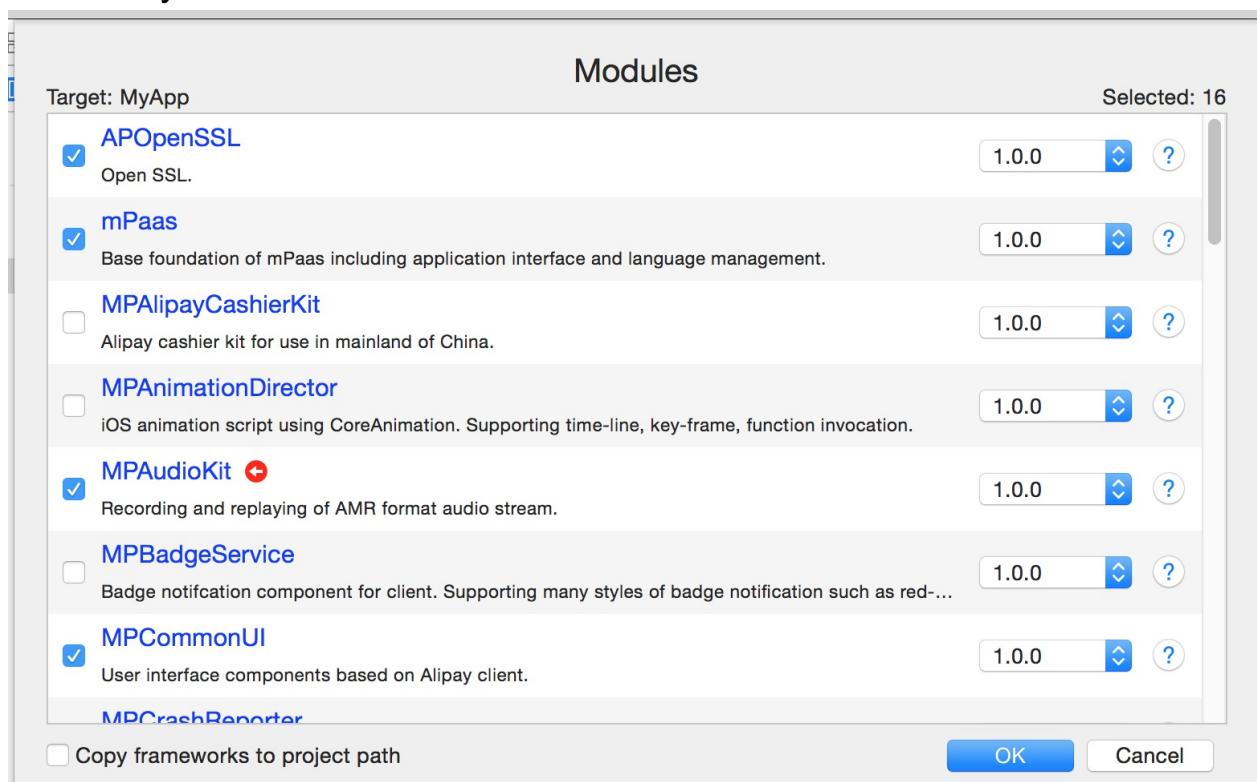
- Install: used to install the mPaaS module.

- **Modify:** used to change the mPaaS module installed in a target.
- **Uninstall:** used to uninstall the installed mPaaS module.

An Xcode project may contain several targets, and this plug-in can be set to each target separately. When a project contains multiple targets, a selection box like the following appears.



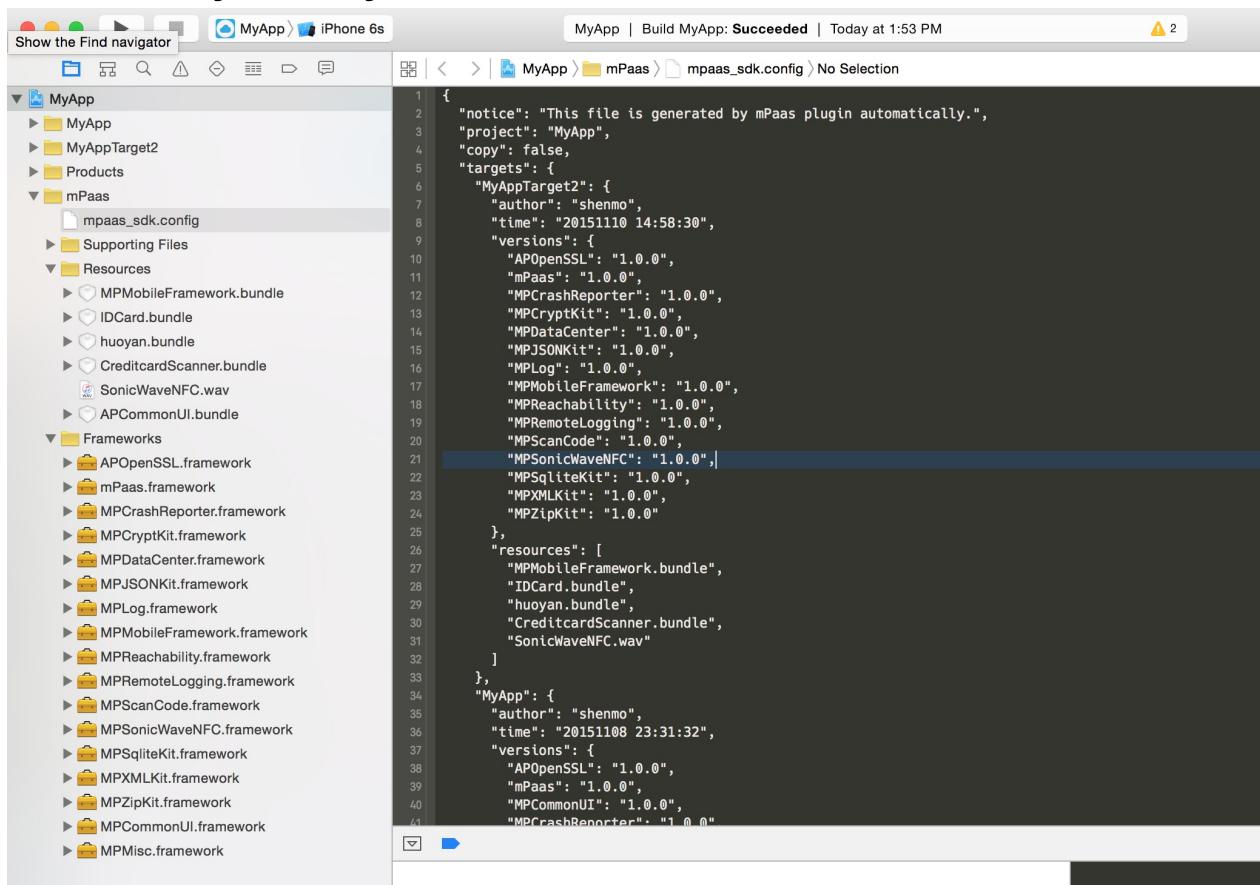
The Modules window appears after you select the target. Check the modules you want and select the versions.



- Red Arrows marks the modules with changes.
- Question Mark Buttons displays the module version and upgrade information.

- Copy frameworks to project path :
 - The mPaaS SDK will be installed in a shared directory of the user's machine.
 - If you leave this option unchecked, the SDK package in the shared directory will be used. Thus the project size is reduced and users only need to care about their own codes. When the project is run in other machines without mobile SDK, as long as the machine has Mobile plug-in installed, the plug-in will download the mobile SDK to ensure the project run normally.
 - If you check this option, the mPaaS SDK will be installed in the `mPaaS` directory of the project and will be taken as a part of the project.

The mPaaS directory exists in the project. Do not make any change in the directory manually.



The screenshot shows the Xcode interface with the project 'MyApp' selected. In the left sidebar, under the 'MyApp' group, there is a folder named 'mPaaS' which contains a file named 'mpaas_sdk.config'. The main editor area displays the contents of this configuration file:

```

1  {
2    "notice": "This file is generated by mPaaS plugin automatically.",
3    "project": "MyApp",
4    "copy": false,
5    "targets": {
6      "MyAppTarget2": {
7        "author": "shemo",
8        "time": "20151110 14:58:30",
9        "versions": {
10          "APOpenSSL": "1.0.0",
11          "mPaaS": "1.0.0",
12          "MPCrashReporter": "1.0.0",
13          "MPCryptKit": "1.0.0",
14          "MPDataCenter": "1.0.0",
15          "MPJSONKit": "1.0.0",
16          "MPLog": "1.0.0",
17          "MPMobileFramework": "1.0.0",
18          "MPReachability": "1.0.0",
19          "MPRemoteLogging": "1.0.0",
20          "MPScanCode": "1.0.0",
21          "MPSonicWaveNFC": "1.0.0",
22          "MPSqliteKit": "1.0.0",
23          "MPXMLKit": "1.0.0",
24          "MPZipKit": "1.0.0"
25        },
26        "resources": [
27          "MPMobileFramework.bundle",
28          "IDCard.bundle",
29          "huayan.bundle",
30          "CreditcardScanner.bundle",
31          "SonicWaveNFC.wav"
32        ]
33      },
34      "MyApp": {
35        "author": "shemo",
36        "time": "20151108 23:31:32",
37        "versions": {
38          "APOpenSSL": "1.0.0",
39          "mPaaS": "1.0.0",
40          "MPCommonUI": "1.0.0",
41          "MPCrashReporter": "1.0.0"
42        }
43      }
44    }
45  }

```

Table of Contents |

If the following errors appear, it means no Ruby Gem of `xcodeproj` is installed in your machine or the version is too old. Please refer to [Overview](#) for installation.

```
mPaaS Kit Output

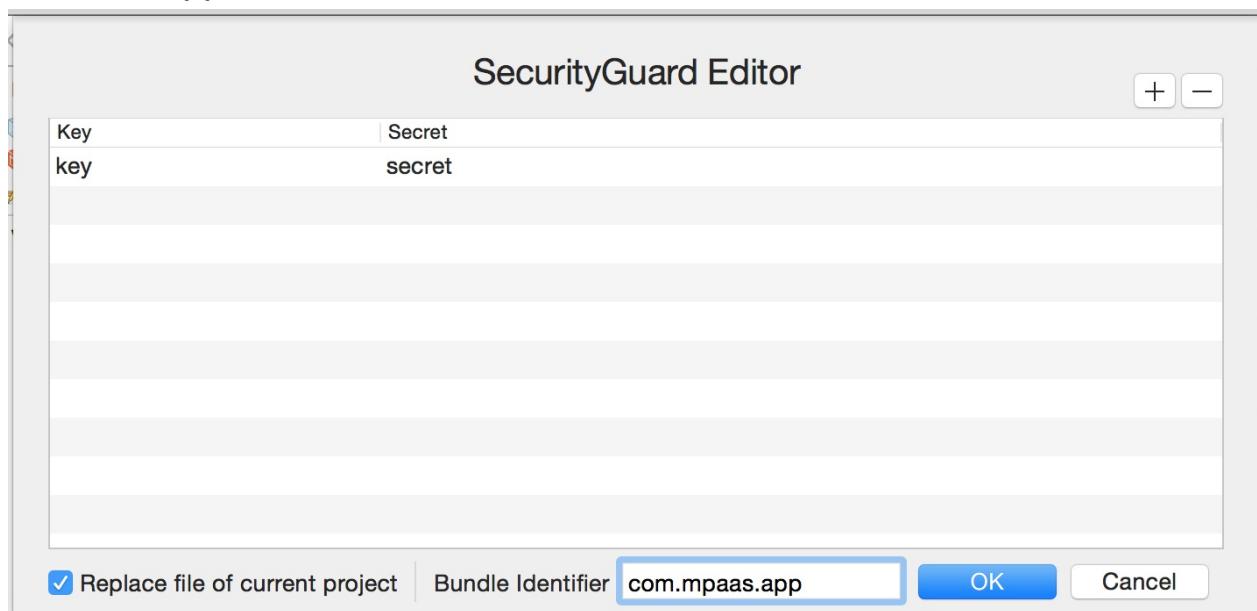
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileutils.rb:93: warning: already initialized constant FileUtils::OPT_TABLE
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileUtils.rb:93: warning: previous definition of OPT_TABLE was here
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileutils.rb:1272: warning: already initialized constant FileUtils::Entry_::S_IF_DOOR
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileUtils.rb:1272: warning: previous definition of S_IF_DOOR was here
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileutils.rb:1535: warning: already initialized constant FileUtils::Entry_::DIRECTORY_TERM
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileUtils.rb:1535: warning: previous definition of DIRECTORY_TERM was here
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileutils.rb:1537: warning: already initialized constant FileUtils::Entry_::SYSCASE
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileUtils.rb:1537: warning: previous definition of SYSCASE was here
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileutils.rb:1656: warning: already initialized constant FileUtils::LOW_METHODS
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileUtils.rb:1656: warning: previous definition of LOW_METHODS was here
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileutils.rb:1662: warning: already initialized constant FileUtils::METHODDS
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/fileUtils.rb:1662: warning: previous definition of METHODS was here
mPaaS SDK command: [/Users/kyao/Desktop/aaaa/aaaa.xcodeproj] [-target "aaaa" -modules "APOpenSSL:1.0.0.mPaaS:1.0.0.MPCommonUI:1.0.0.MPCrashReporter:1.0.0.MPDatadCenter:1.0.0.MPFeedbackKit:1.0.0.MPHtppClient:1.0.0.MPJSONKit:1.0.0.MPLog:1.0.0.MPMisc:1.0.0.MPMobileFramework:1.0.0.MPMonitor:1.0.0.MPReachability:1.0.0.MPRemoteLogging:1.0.0.MPScanCode:1.0.0.MPSecurityUtility:1.0.0.MPSqliteKit:1.0.0.MPTHreadManager:1.0.0.MPXMLKit:1.0.0.MPZipKit:1.0.0.SecurityGuardSDK:3.0.9,UTDID:1.0.2"]
Backup project.

Uninstall mPaaS.
2015-11-12 10:34:06.339 ruby[14498:81134] [MT] DVTAAssertions: ASSERTION FAILURE in /Library/Caches/com.apple.xbs/Sources/IDEFrameworks/IDEFrameworks-9079/IDEFoundation/Initialization/IDEInitialization.m:590
Details: Assertion failed: _initializationCompletedSuccessfully
Function: BOOL IDEIsInitializedForUserInteraction()
Thread: <NSThread: 0x7ccb5aa79b00>-(number = 1, name = main)
Hints: None
Backtrace:
0 0x00000000101c2bf6 _DVTAAssertionHandler handleFailureInFunction:fileName:lineNumber:assertionSignature:messageFormat:arguments:] (in DVTFoundation)
1 0x00000000101c2b723 _DVTAAssertionHandler (in DVTFoundation)
2 0x00000000101c2b89f _DVTAAssertionFailureHandler (in DVTFoundation)
3 0x00000000101c2b8f1 _DVTAAssertionFailureHandler (in DVTFoundation)
4 0x000000001032456fd IDEIsInitializedForUserInteraction (in IDEFoundation)
5 0x00000000105a78081 +[PBXProject projectWithFile:errorHandler:readOnly:] (in DevToolsCore)
6 0x00000000105a79c06 +[PBXProject projectWithFile:errorHandler:] (in DevToolsCore)
7 0x000007ff8ec50f44 ffi__call_unix64 (in libffi.dylib)

OK
```

4.2 Key Image Generator of Mobile Security Guard

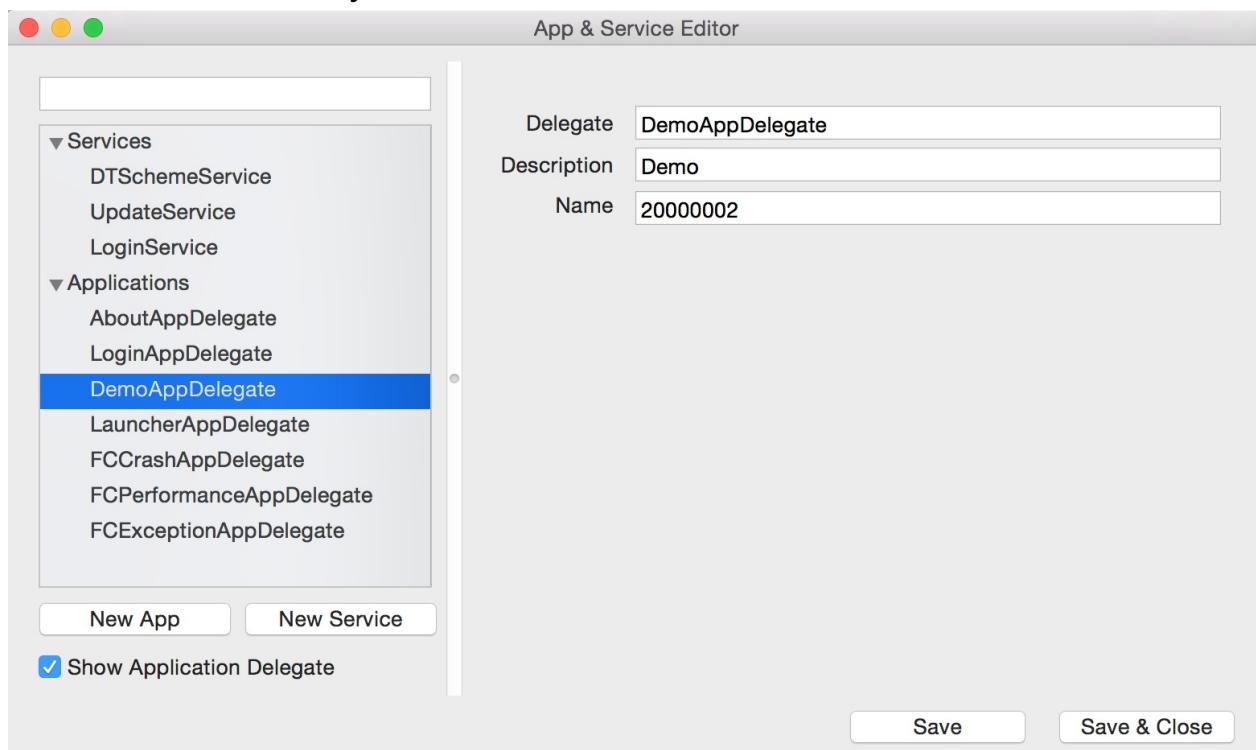
In the mPaaS menu, click `Security Guard Editor` and the following interface appears.



Click + or - on the upper right to add or delete a key pair. You can edit the content in the list. The key image must match the Bundle Identifier of the application. If you check Replace file of current project, the generated image will replace the yw_1222.jpg image of the project.

4.3 App & Service Editor

Select mPaaS > App & Service Editor, then click the project using mobile client SDK, the App & Service Editor will be opened. From this editor developers can view the registered micro-applications or micro-services, and edit them visually.



4.4 DAO Config Generator

In the Editor area of Xcode, select the class declarations that needs to generate DAO configuration files, and then click DAO Config Generator in the mPaaS menu to generate DAO configuration files automatically. The

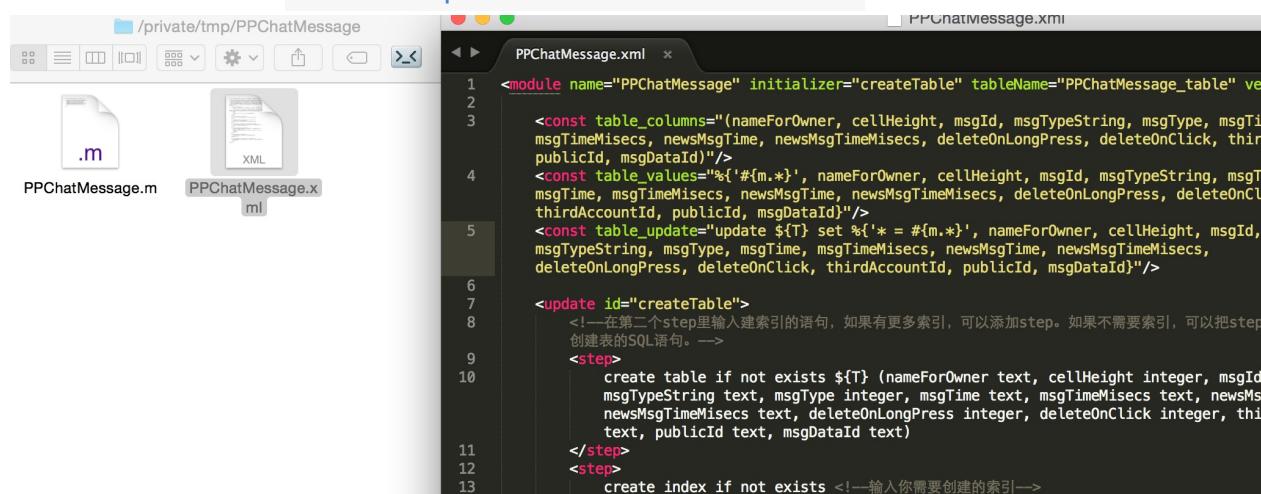
files will be created in the temporary directory. You can copy them to the project directory and add it to the project.

```

39     curData ? curData : @ ""
40
41 @class PubChatMessage;
42
43 /**
44 * 消息基类
45 */
46 @interface PPChatMessage : NSObject <NSCoding>
47
48 @property(nonatomic, strong) NSString *nameForOwner;//消息所属方名字
49 @property(nonatomic, assign) int cellHeight;//保存动态计算的高度，增加加载速度
50 @property(nonatomic, strong) NSString *msgId;
51 @property(nonatomic, strong) NSString *msgTimeString;
52 @property(nonatomic, assign) MessageType msgType;
53 @property(nonatomic, strong) NSString *msgTime;
54 @property(nonatomic, strong) NSString *msgTimeMisecs;
55 @property(nonatomic, strong) NSString *newsMsgTime;
56 @property(nonatomic, strong) NSString *newsMsgTimeMisecs;
57 @property(nonatomic, assign) BOOL deleteOnLongPress;
58 @property(nonatomic, assign) BOOL deleteOnClick;
59 @property(nonatomic, strong) NSString *thirdAccountId;
60 @property(nonatomic, strong) NSString *publicId;
61 @property(nonatomic, strong) NSString *msgDataId;
62
63 @end
64

```

You can refer to [Tool Component Set->Data Center](#) for how to use DAO.



4.5 Other Plug-ins

4.5.1 Open Sandbox

In the mPaaS menu, click [Open Sandbox](#) to quickly open the application simulator sandbox.

4.5.2 Copy-Paste Detector

In the mPaaS menu, click `Copy-Paste Detector` to detect the duplication of CPD-based code.

4.5.3 Open Shell

In the mPaaS menu, click `Open Shell` to quickly open the client and locate the path of the current project.

@cnName 5 Command Line Tool @priority 5

5 Command Line Tools

[TOC]

After Developer Tools installed, there is a set of Command Line Tools beside Project Template, File Template and Xcode Plug-in.

5.1 Update or Re-install Developer Tools

Run the command below on the terminal to re-install the latest Developer Tools.

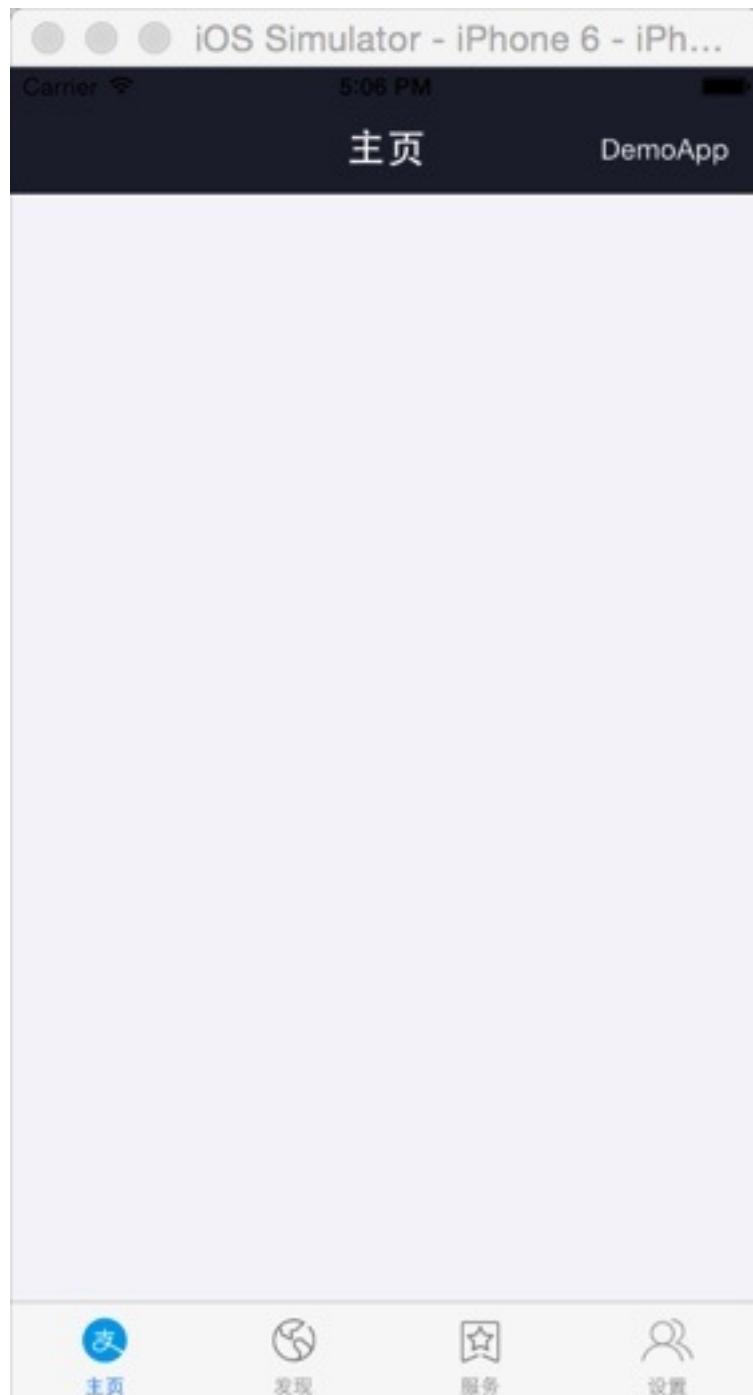
```
mpaas kitupdate
```

5.2 Create APP

Enter the target directory and execute the command below to create a project named XXXX

```
mpaas createapp XXXX
```

You can directly run this project which includes a Demo APP and service instance based on multi-tab. You can develop APPs on base of this demo and change the modules using Xcode Plug-in.

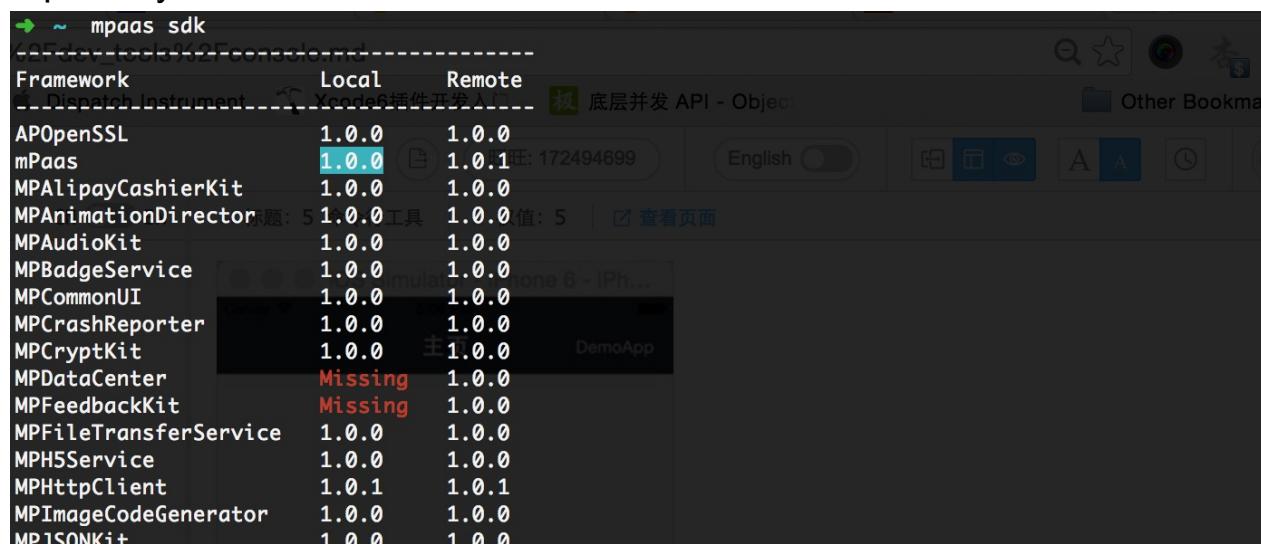


5.3 Check SDK Version Information

Run the command below on the terminal to print out the module information of local SDK.

```
mpaas sdk
```

In the result, `Local` indicates the version of local module; `Remote` indicates the latest version in repository; `Missing` means this module hasn't been downloaded to local machine; and the information in cyan background means the version of module is inconsistent with that in repository.



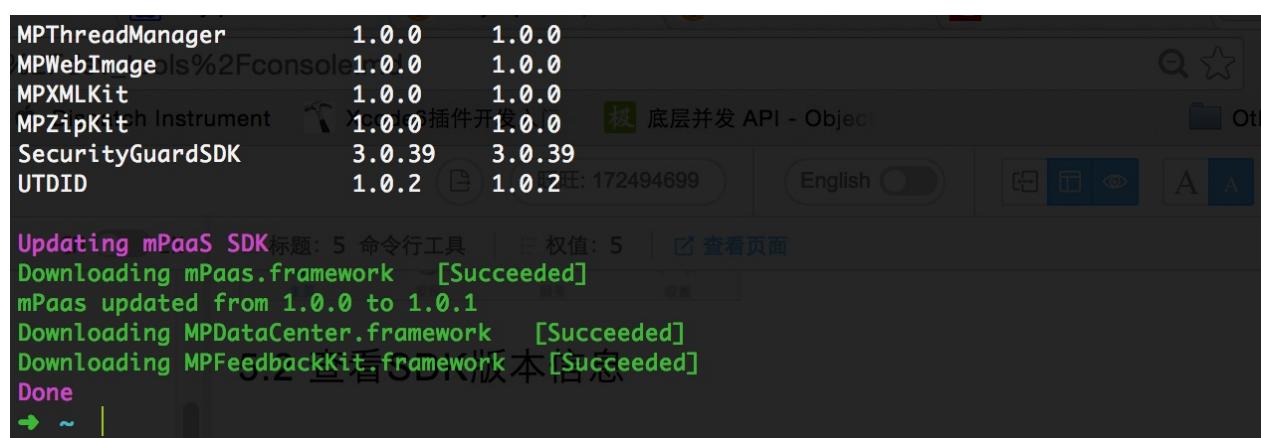
Framework	Local	Remote
APOpenSSL	1.0.0	1.0.0
mPaaS	1.0.0	1.0.1
MPAlipayCashierKit	1.0.0	1.0.0
MPAnimationDirector	1.0.0	1.0.0
MPAudioKit	1.0.0	1.0.0
MPBadgeService	1.0.0	1.0.0
MPCommonUI	1.0.0	1.0.0
MPCrashReporter	1.0.0	1.0.0
MPCryptKit	1.0.0	1.0.0
MPDataCenter	Missing	1.0.0
MPFeedbackKit	Missing	1.0.0
MPFileTransferService	1.0.0	1.0.0
MPH5Service	1.0.0	1.0.0
MPHttpClient	1.0.1	1.0.1
MPIImageCodeGenerator	1.0.0	1.0.0
MPJSONKit	1.0.0	1.0.0

5.4 Update Local SDK

Run the command below on the terminal

```
mpaas sdk update
```

Then local SDK will be synchronized with that in repository, i.e. the missing modules will be downloaded and the existing modules will be updated to the latest version.



```

MPThreadManager      1.0.0    1.0.0
MPWebImage          1.0.0    1.0.0
MPXMLKit            1.0.0    1.0.0
MPZipKit            1.0.0    1.0.0
SecurityGuardSDK    3.0.39   3.0.39
UTDID               1.0.2    1.0.2

Updating mPaaS SDK [Succeeded]
mPaaS updated from 1.0.0 to 1.0.1
Downloading MPDataCenter.framework [Succeeded]
Downloading MPFeedbackKit.framework [Succeeded]
Done

```

5.5 Check Local Version and Update Information of a Certain Module

Run the command below on the terminal, replace xxxx with the module name.

```
mpaas sdk XXXX
```

```
➔ ~ mpaas sdk MPHttpClient
MPHttpClient.framework is 1.0.1 in local mPaaS SDK.

对NSURLSession的封装，提供下载、上传，RPC调用等功能。
Encapsulation of NSURLRequest providing uploading, downloading, RPC functionality.

Versions:

1.0.0
    初始版本
    Initial version.

    Dependencies: mPaas, MPZipKit, MPJSONKit, MPCryptKit, MPLog, MPReachability, SecurityGuardSDK, UT DID

1.0.1
    RPC模块可以指定请求加签名时使用的密钥。
    You can specify a request signing key for RPC.

    Dependencies: mPaas, MPZipKit, MPJSONKit, MPCryptKit, MPLog, MPReachability, SecurityGuardSDK, UT DID
```

5.6 Update a Certain module to the Latest Version

Run the command below on the terminal, replace xxxx with the module name. The module will be re-downloaded regardless of current version.

```
mpaas sdk update XXXX
```

```
➔ ~ mpaas sdk update mPaas
Downloading mPaas.framework [Succeeded]
mPaas updated from 1.0.1 to 1.0.1
```

5.7 Download or Re-download a Certain Module

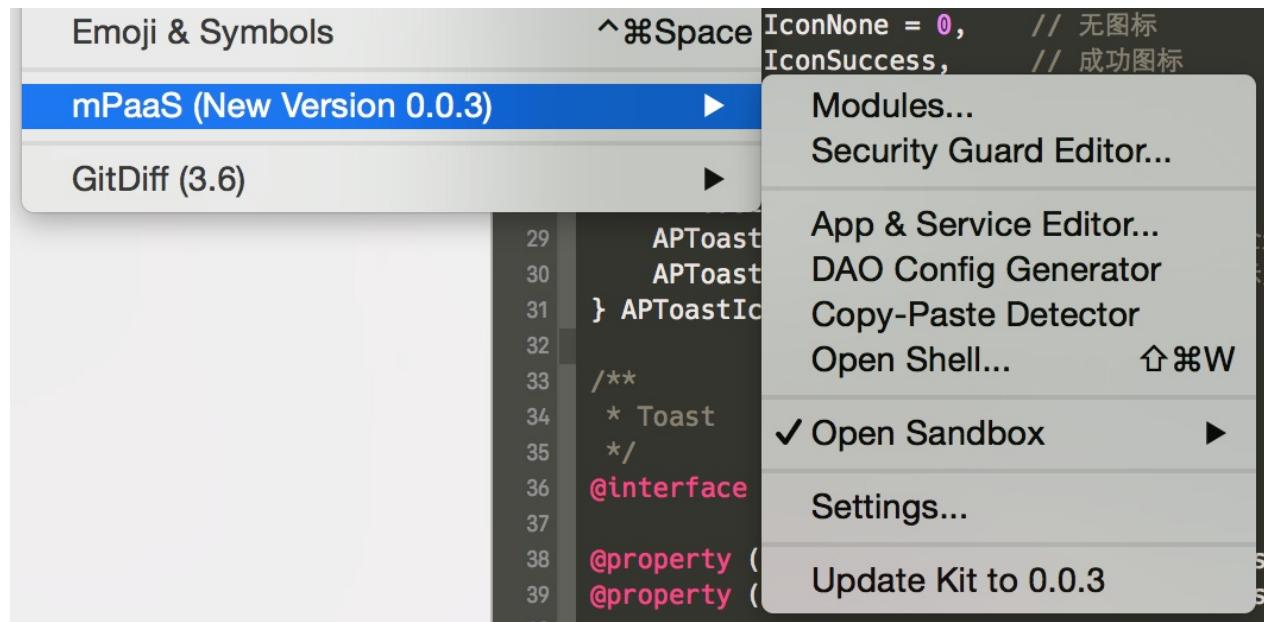
Run the command below on the terminal, replace xxxx with the module name. The module will be re-downloaded even it already exists in local.

```
mpaas sdk download XXXX
```



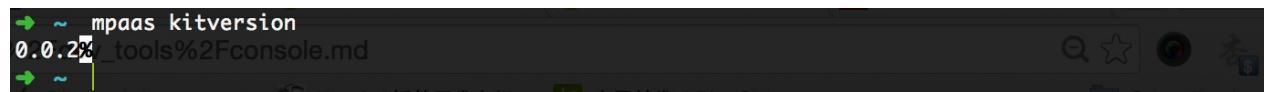
5.8 Check Version of Developer Tools

The version of Developer Tools can be found in mPaaS menu of Xcode; if update is available, a prompt will appear.



Current version of Developer Tools can also be found by running the command below.

```
mpaas kitversion
```



@cnName 6 Release History @priority 6

6 Release History

Appendix

1 Usage of open-source code

The following open-source codes are referenced or modified in the mPaaS iOS client codes

Open-source project	Role
OpenCore	MPAudioKit
MBProgressHUD	MPCommonUI
ODRefreshControl	MPCommonUI
TTTAttributedLabel	MPCommonUI
EGOResfreshTableHeaderView	MPCommonUI
PLCrashReport	MPCrashReporter
libpng	MPIImageCodeGenerator
Barcode	MPIImageCodeGenerator
QR_Encode	MPIImageCodeGenerator
Reachability	MPReachability
FMDB	MPSqliteKit
SDWebImage	MPWebImage
KissXML	MPXMLKit
MiniZip	MPZipKit

The codes of the following open-source plug-in are adopted in mPaaS plug-ins

Open-source project
Lin
VVDocumenter