MobileArkSDK 开发手册 iOS 版

南京烽火星空通信发展有限公司

目　录

1 概述

1.1 适用环境

适用于 IOS 5 及以上，支持 Armv7 与 Arma7s 平台。

1.2 主要组成

下载好的 zip 包中包括三个文件：

1、 MobileArkSDK 文件夹，内部包括 libMobileArkSDK.a、MobileArkAgent.h、MobileArkPSA.h、FHSSO.h 四个文件。

2、 demo:包含一个 SDK 调用实例。

3、 MobileArkSDK 开发手册（ios）.doc:开发手册文件。

1.3 使用前提条件

设备应已经安装 MobileArk 客户端，在 MobileArk 服务器端注册相关应用的包名。

MobileArk 客户端版本要求 2.0 版本及以上。

MobileArk 服务器端版本要求 2.0 版本及以上。

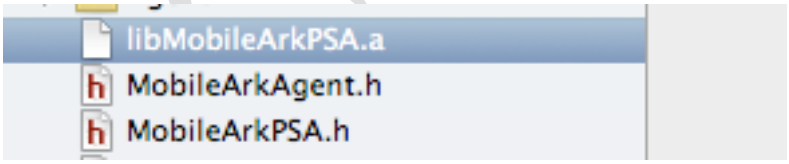
2 SDK 导入与工程设置

2.1 下载 SDK

下载 SDK 包，解压后得到四个文件：libMobileArkSDK.a、MobileArkAgent.h、MobileArkPSA.h、FHSSO.h。

2.2 导入静态库

将三个文件添加到工程文件目录内，右键选择 Add->Existing Files…，选择文件导入工程。或者将这三个文件拖入 XCode 工程目录结构中，在弹出的界面中勾选 Copy items into destination group's folder(if needed)，并确保 Add To Targets 勾选相应的 target。



添加依赖框架（FrameWork）和编译选项

TARGETS-->Build Phases-->Link Binary With Libraries--> + -->libsqlite3.dylib Required

TARGETS-->Build Phases-->Link Binary With Libraries--> + -->CoreTelephony.framework Optional

具体操作步骤如图：

选择左侧工程后选择你需要添加 sdk 的 target，然后选择 Build phases，点击 Link Binary With Libraries 下面的+号

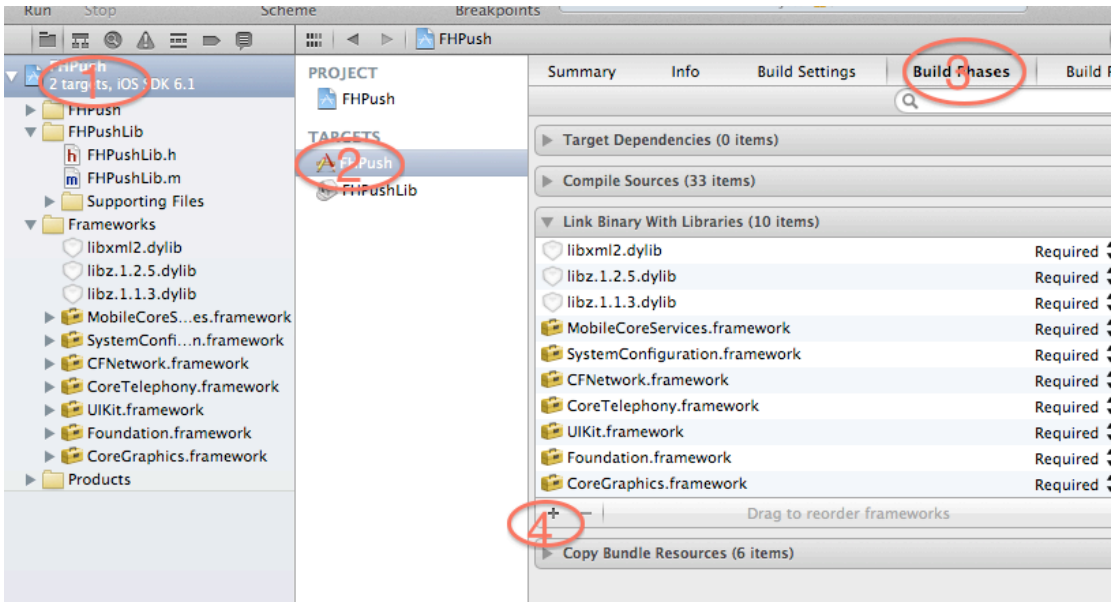


图 2-1

进入如下界面，在搜索栏里输入你库名的前几个字母，如本文数据 coretele，将搜索到如图的库名，选择库，点击 add，则会讲该 FrameWork 添加入工程，

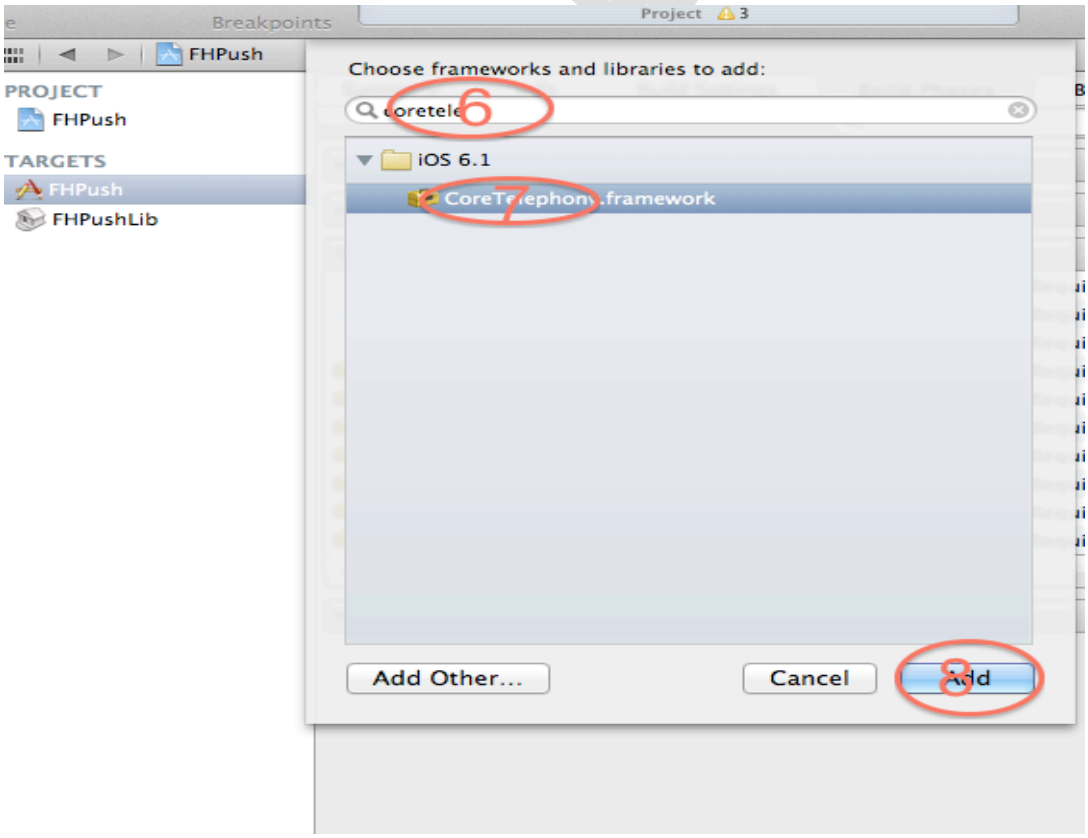


图 2-2

如图 2-3 显示了 coreTelephony.frameWork 已经添加入工程。

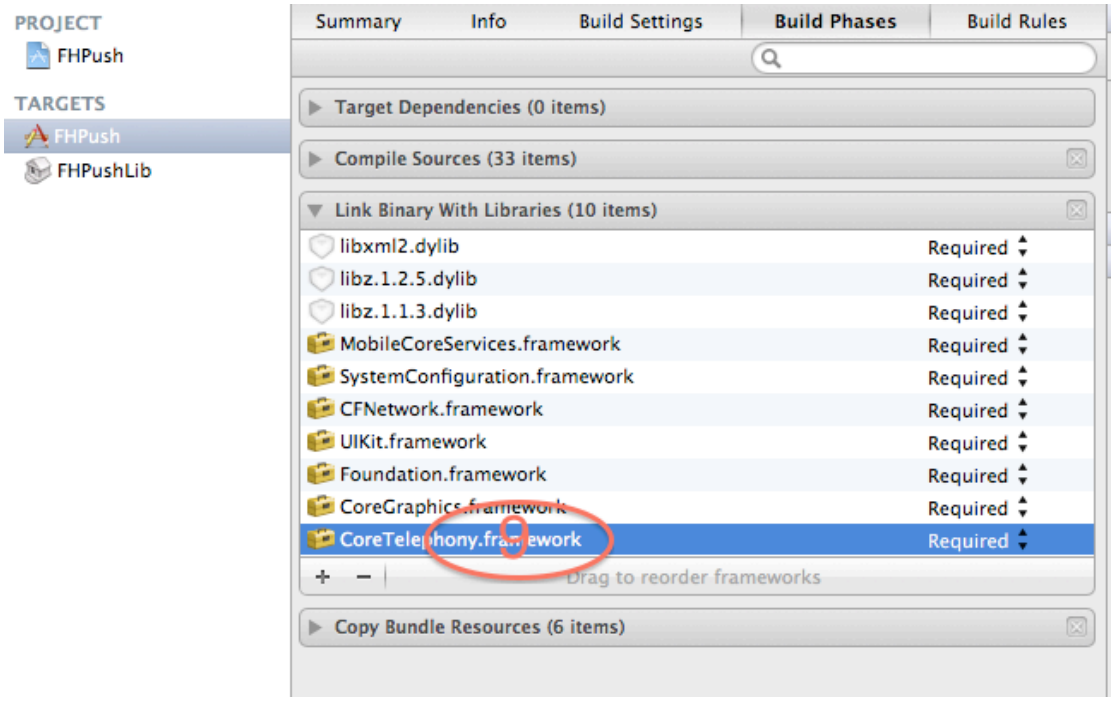


图 2-3

相同的方法添加 SystemConfiguration.framework、MobileCoreServices.framework、CoreGraphics.framework、CFNetwork.framwork、libxml2.dylib 和 libz.1.2.5.dylib(若 IOS5

以前的版本则为 libz.1.2.3.dylib)

设置：

TARGET→Build Settings→Search Paths →Header Search Paths 添加值：/usr/include/libxml2 如图 2-4

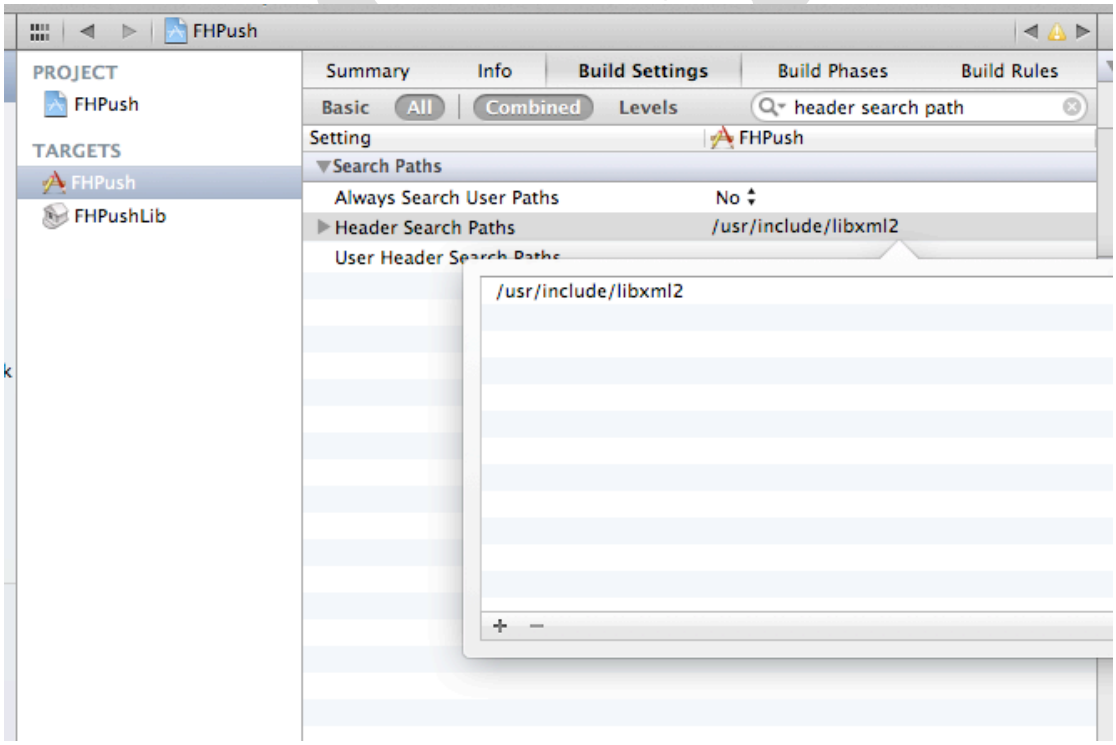


图 2-4

TARGET→Build Settings →Linking → Other Linker Flags 添加：-lxml2 如图 2-5

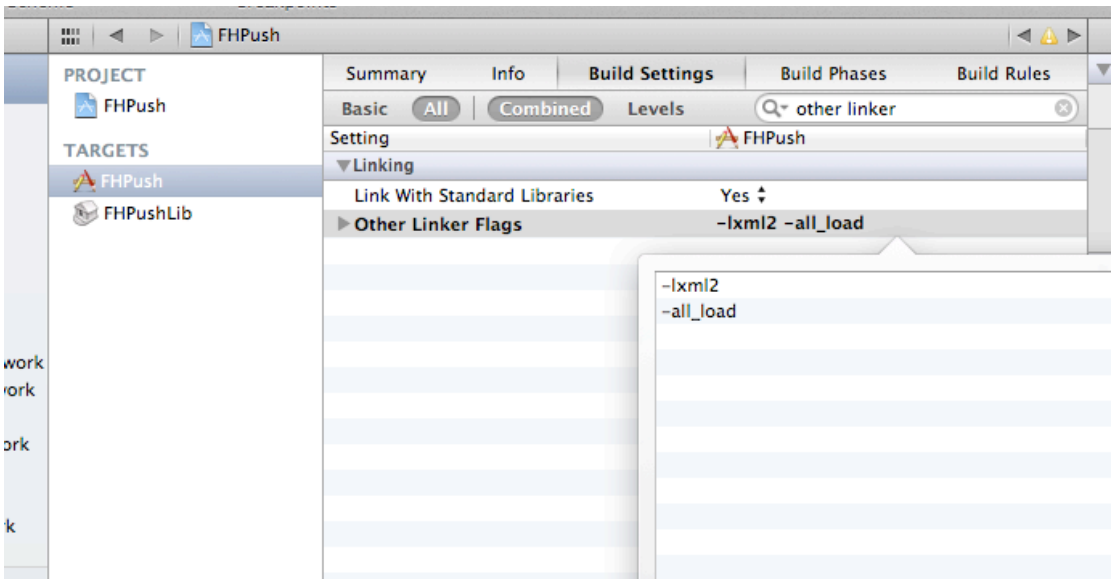


图 2-5

2.3 新版单点登录需要配置的选项

需要导入以下库文件

CoreTelephony.framework,Foundation.framework,SystemConfiguration.framework,MobileCoreServices.framework,CFNetwork.framework,libz.1.dylib,libsqlite3.dylib,Security.framework,

创建KeychainItemWrapper.plist,添加到所需要的工程去,添加mplus的teamID以及bundleID到KeychainItemWrapper.plist文件中

去.KeychainItemWrapper.plist要添加到工程文件Build Settings->Code Signing Entitlements,见图2-6.

KeychainItemWrapper.plist文件结构见图2-7.

KeychainItemWrapper.plist需要添加在一个TEAMID下的证书.

MAResource.zip

需要导入图片资源 ,图片资源必须导入工程,手势密码需要这些资源.

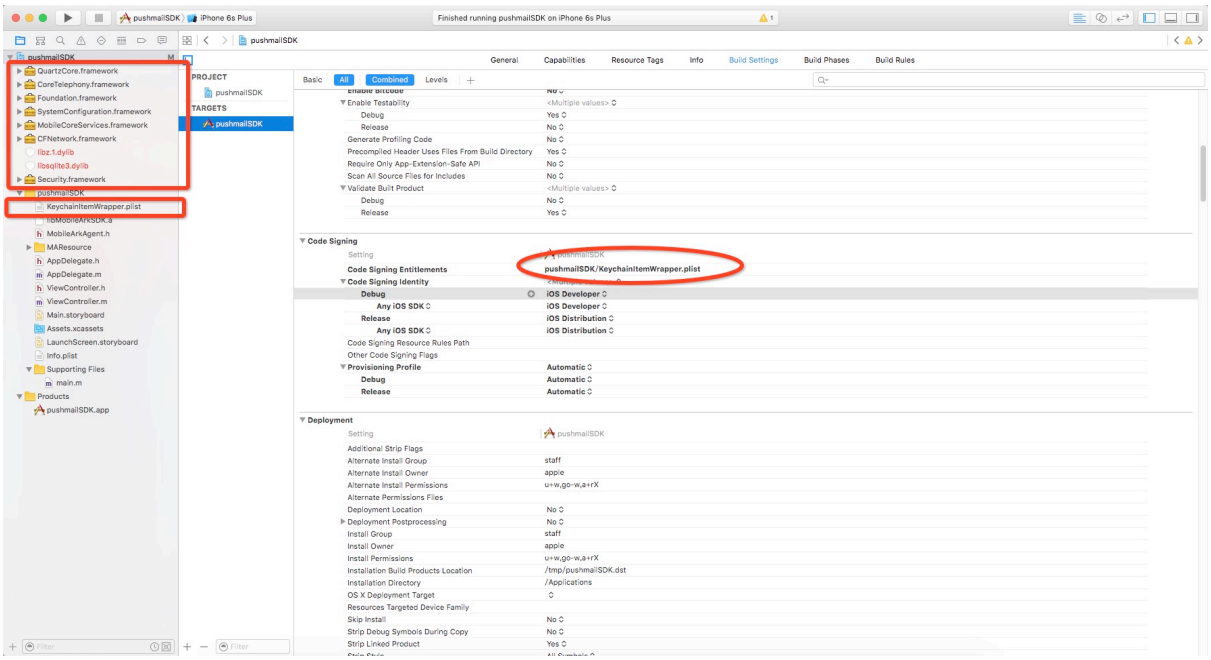


图 2-6

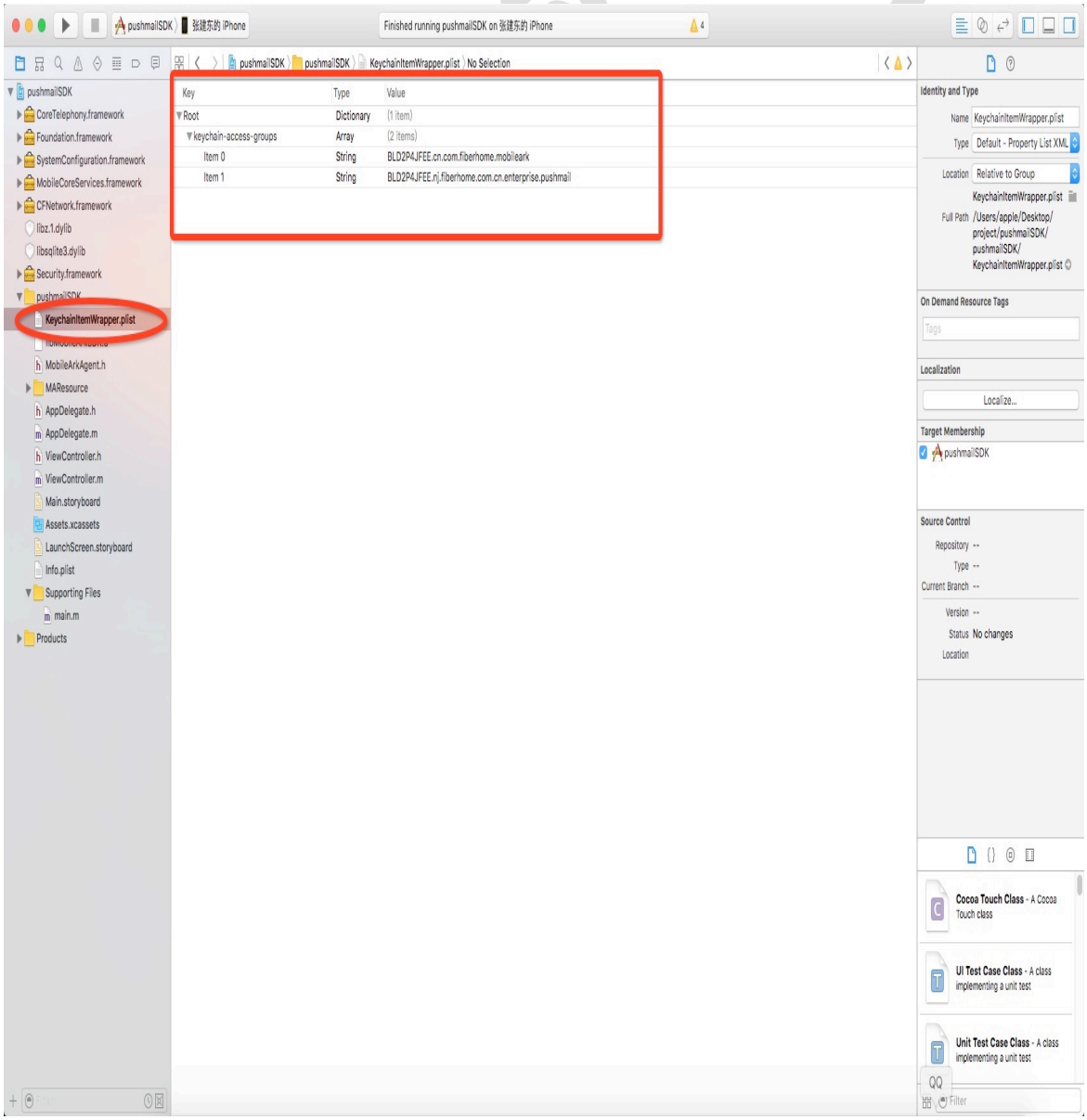

pushmailSDK
张建东的 iPhone
Finished running pushmailSDK on 张建东的 iPhone
pushmailSDK
pushmailSDK
KeychainItemWrapper.plist
No Selection
Key
Type
Value
Root
Dictionary
(1 item)
keychain-access-groups
Array
(2 items)
Item 0
String
BLD2P4JFEE.cn.com.fiberhome.mobileark
Item 1
String
BLD2P4JFEE.nj.fiberhome.com.cn.enterprise.pushmail
KeychainItemWrapper.plist
MobileArkAgent.h
MAResource
AppDelegate.h
AppDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Supporting Files
main.m
Products
Identity and Type
Name KeychainItemWrapper.plist
Type Default - Property List XML
Location Relative to Group
KeychainItemWrapper.plist
Full Path /Users/apple/Desktop/project/pushmailSDK/pushmailSDK/KeychainItemWrapper.plist
On Demand Resource Tags
Localization
Localize...
Target Membership
pushmailSDK
Source Control
Repository --
Type --
Current Branch --
Version --
Status No changes
Location
Cocoa Touch Class - A Cocoa Touch class
UI Test Case Class - A class implementing a unit test
Unit Test Case Class - A class implementing a unit test
Filter

图 2-7

3 SDK 使用方法

3.1 获取版本号

集成 SDK 后开发者可以通过如下方法获取当前 SDK 版本号：

```
NSString * version =[MobileArkPSA getVersion];
```

3.2 单点登录 V1

3.2.1 设计思想

APP 之间可以通过 OpenUrl 方式传递参数，在实现单点登录的过程中，MobileArk 可以跟 Native APP 在实现单点登录的过程中传递 secretToken 等参数，以实现认证信息的交互传递。如图 3-2 所示，APP 在登录的时候通过 OpenUrl 启动 MobileArk 进行登录，MobileArk 登录后通过 OpenUrl 方式想 APP 传递登录参数，实现单点登录。

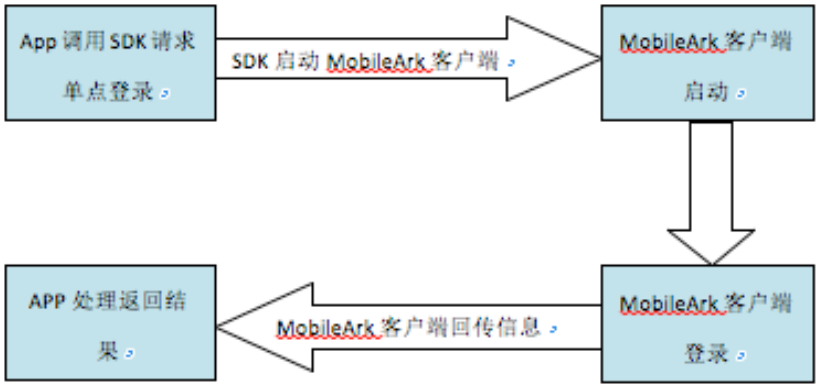


图 3-2

3.2.2 适用场景

应用支持 SSO，是为了统一认证，为用户提供安全友好的登录体验。

多个应用共享登录认证结果，不需要单独进行登录认证，减少了认证服务器的操作，提高用户体验。

3.2.3 基本使用

单点登录的过程中通过 OpenUrl 方式传递参数，所以要设置应用注册的 URL 前缀如下，应用的 urlscheme 一般为应用的 BundleID。

URL types	Array	(1 item)
Item 0	Dictionary	(2 items)
URL identifier	String	cn.com.fiberhome.nj.SDKSample
URL Schemes	Array	(1 item)
Item 0	String	cn.com.fiberhome.nj.SDKSample

3.2.3.1 登录

当您需要进行单点登录的时候调用函数：

```
[FHSSO  startConnectSSO];
```

3.2.3.2 获取单点登录回传信息

当 MobielArk 向本 APP 传送数据时，我们需要在*APPDelegate.h 文件中嵌入代码获取登录认证信息：

```
-(BOOL)application:(UIApplication *)application
handleOpenURL:(NSURL *)url
```

在以上方法中加入代码：

```
[FHSSO  handleConnectApp:url];
```

在登录后回传处理会自动保存相应的secretToken等数据，当需要数据时可以调用相应的接口获取。

3.2.3.3 应用更新检测

通过 SDK 获取当前应用是否要更新,需要传入 appid 和 appversion

+(BOOL)checkAppVersion:(NSString*)appID andAppVersion:(NSString*)appVersion;

+(NSDictionary*)appInstallForIOS:(NSString*)appID andAppVersion:(NSString*)appVersion;

E.g:

BOOL ret =[MobileArkAgent checkAppVersion:appid andAppversion:appversion];

如果 ret 返回的是 YES,需要用户调用应用安装的请求

NSDictionary* dic =[MobileArkAgent appInstallForIOS:appid andAppVersion:appVersion];

返回的字典里面包含 resultcode(0 表示成功 -1 表示其他错误)和 resultmessage,

3.2.3.4 应用启动手势密码

用户切换前后台看是否启动手势密码

#pragma 切换到后台
+(void)enterBackground;
#pragma 手势密码页面是addsubview上去
#pragma 切换到前台
+(void)willBecomeActive;
#pragma 展示手势密码
+(BOOL)showGestureView;
返回值为YES表示有手势密码，返回NO表示无手势密码，需要第三方唤起门户到登录页面，重新输入登录设置手势密码
#pragma 第三方基于nav pushview手势密码的方式
+(BOOL)showNavGestureView:(UIViewController*)view;
返回值为YES表示有手势密码，返回NO表示无手势密码，需要第三方唤起门户到登录页面，重新输入登录设置手势密码

+(void)willNavBecomeActive:(UIViewController*)view;

#pragma 手势密码页面present上去
+(void)willPresentBecomeActive;

static NSString *const NOTIFICATION_GESTURE_IS_WRONG =@"notification fhlock gesture is wrong";//手势密码输入错误

static NSString *const NOTIFICATION_CHECK_GESTURE_SUCCESS =@"notification fhlock check gesture";//手势密码验证成功

切换前后台

1:判断有没有手势密码,如果没有密码是不会弹出手势密码页面,有手势密码页面会弹出

2.输入手势密码成功或者失败需要用户在页面注入上面两个通知,

NOTIFICATION_GESTURE_IS_WRONG 这个表示输入密码五次错误,弹出 alert,然后会通知用户,用户填入自己所需操作; NOTIFICATION_CHECK_GESTURE_SUCCESS 这个表示输入密码成功, 用户填入自己所需操作.

3.如果用户需要拉起 MPlus 客户端,需要添加如下代码:

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:[NSString stringWithFormat:@"MobileArk://"]]];

FAQ:需要第三方手动添加延迟 5 秒, 再拉起 MPlus 客户端

认证信息:

```
-(void)  applicationDidEnterBackground:(UIApplication*)application
{
    [MobileArkAgent  enterBackground];
}

-  (void)  applicationWillEnterForeground:(UIApplication*)application
{
    [MobileArkAgent  willBecomeActive];
}
```

3.3 单点登录 V2

3.3.1.1 新版单点登录,自动刷新 token

当 token 即将过期,更新 token

E.g:NSString *str = [MobileArkAgent getToken];

3.3.1.2 获取用户名,密码,ECID 以及 ESN

当获取用户名,密码,ECID 以及 ESN,需要用户传入 key 值

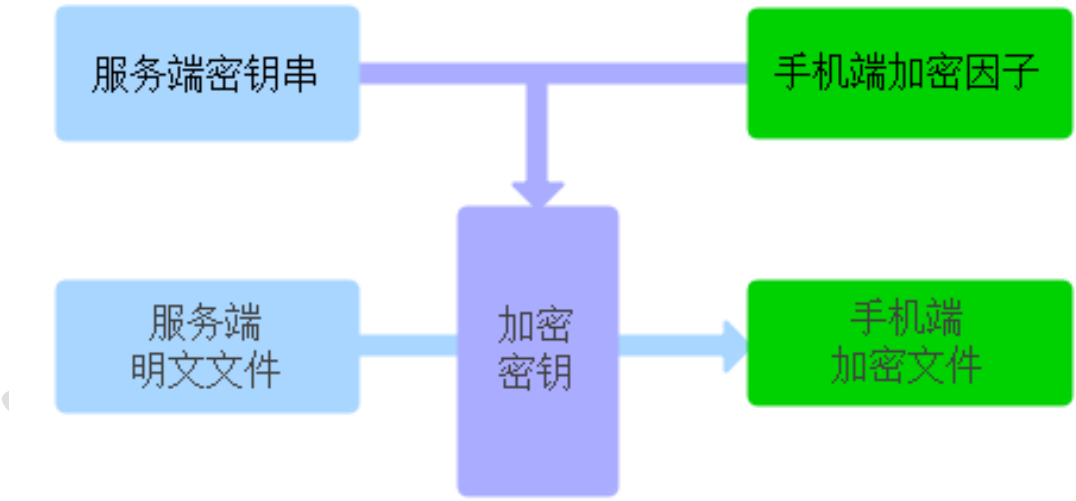
Key 值分别是: loginname, password, ecid 以及 esn

E.g:NSString *str = [MobileArkAgent getParam:@"loginname"];

3.4 安全存储

3.4.1 设计思想

将从服务端获取到的在线密钥串和客户端的加密因子合并，通过算法生成一个加密秘钥，并用该秘钥加密存储所有客户端从服务端下载的文件。



使用该模块，应用不再需要单独实现加密功能。就可实现文网络上缓存的数据，下载的数据进行加密存储，极大地提高了效率和安全性。

3.4.2 适用场景

该模块适用于对安全要求较高的文件加密。对应用提供了统一的，全面的而且可靠的加密方式。

3.4.3 基本使用

3.4.3.1 用户认证

获取服务端的加密密钥串。该请求需要向认证服务器发起 https 请求，mobileark 服务器默认的请求 uri 为/dataanth。

客户端请求，参数:用户名、密码

```
<?xml version='1.0' encoding='UTF-8'?>
        <auth_interface>
        <username>zhangsan</username>
                <!-- 安全令牌和用户密码 2 选 1 传入 用户密码用于应用单独认证的情况、安全令牌用于单点登录的场景 -->
                 <password>用户密码</password >
                 <token>安全令牌</token>
            <version>1.0</version>
</auth_interface>
```

服务器返回， resultcode：1 成功;0 失败;

```
<?xml version=’1.0’ encoding=’UTF-8’?>
       < auth_interface>
```

```
                <resultcode>1</resultcode>
                 <resultdesc>登录成功！</resultdesc>
                 <timestamp>20130105222222</timestamp>
          <userId>11desqdadzcasq</userId>
         <encrykey>12345678</encrykey>
                <version>1.0</version>
</auth_interface>
```

3.4.3.2 设置加密秘钥

将服务器端获取的加密秘钥设置到 SDK 中和客户端的加密因子合并，在 SDK 内生成加密秘钥

```
[CryptoSDKManager    setOnlineKey:(NSString *) key];
```

3.4.3.3 加密文件

直接加密文件。

```
[CryptoSDKManager    encryptContentsOfFile:(NSString *)file    toFile:(NSString *)toFile] ;
```

file：传入需要加密的文件路径。 toFile：加密后文件存储路径。

3.4.3.4 加密 NSData

将网络返回的 NSData 加密，并直接将加密后的数据写入文件

```
[CryptoSDKManager    encryptData:(NSData*)data toFile:(NSString *)filePath] ;
```

data:网络返回的 NSData。 filePath: 加密后写入的文件位置。

3.4.3.5 解密文件

将文件内容解密后存入某个位置。

```
[CryptoSDKManager    decryptContentFile:(NSString*)file    toFile:(NSString *)toFile] ;
```

file：需要解密的文件路径。 toFile：解密后文件存储路径。

将文件内容解密后返回 NSData。

```
[CryptoSDKManager    dencryptDataFromFile:(NSString*)file] ;
```

file：需要解密的文件路径。该函数返回解密后的 NSData。

3.5 MAM 接口

3.5.1.1 应用更新检测

当需要获取当前应用是否要更新,需要传入 appid 和 appversion

BOOL ret =[MobileArkAgent checkAppVersion:appid andAppversion:appversion];

如果 ret 返回的是 YES,需要用户调用应用安装的请求

NSDictionary* dic =[MobileArkAgent appInstallForIOS:appid andAppVersion:appVersion];

返回的字典里面包含 resultcode(0 表示成功 -1 表示其他错误)和 resultmessage

3.6 应用启动手势密码

用户切换前后台看是否启动手势密码

需要在*APPDelegate.h 文件中嵌入代码

```
-(void) applicationDidEnterBackground:(UIApplication*)application
{
    [MobileArkAgent enterBackground];
}

- (void) applicationWillEnterForeground:(UIApplication*)application
{
    [MobileArkAgent willBecomeActive];
    或者
    [MobileArkAgent willPresentBecomeActive];
    或者
    [MobileArkAgent willNavBecomeActive:(UIViewController)];

}
```

4 接口说明

4.1 获取版本号接口

+(NSString *)getVersion

说明	获取当前集成的 SDK 版本号
参数	无
返回值	版本号字符串

4.2 单点登录 V1 接口

4.2.1 登录

+(void)startConnectSSO

说明	启动 MobileArk 并通过传参通知 MobileArk 登录
参数	无
返回值	无

4.2.2 解析启动所传入参数

+(void)handleConnectApp:(NSURL *)url;

说明	处理 MobileArk 进行登录认证后传递的参数
参数	url：MobileArk 启动 Native APP 所传递 URL
返回值	无

4.3 单点登录 V2 接口

4.3.1 新版单点登录,自动刷新 token

+(NSString*)getToken;

说明	获取 token
参数	无
返回值	返回字典格式,需要第三方自己解析 成功:

	{
	resultcode = 0;
	resultmessage = success;
	secrettoken = "XXX";
	}
	失败:
	{
	resultcode = 1;
	resultmessage =账号密码错误;
	}

备注	成功时返回有效 toekn,失败时响应为 json，例如：{"resultcode":"1/-1","resultmessage":""} 1：账号密码错误,需要第三方唤起 mplus,会跳到登陆界面；-1：其他错误 如果是网络问题则 json 返回 null，使用的时候最好判断 null 和空字符串

4.3.2 获取用户名户密码以及 ECID

+(NSString*)getParam:(NSString*)key;

说明	获取参数
参数	key：@"loginname" @"password" @"ecid"
返回值	成功时返回有效的值,失败时返回空

4.4 安全存储接口

4.4.1 设置公钥

+(void)setOnlineKey:(NSString*)key;

说明	设置在线公钥
参数	key：服务端返回的公钥
返回值	无

4.4.2 加密数据

+(void) encryptData:(NSData *)data toFile:(NSString *)filePath;

说明	加密数据
参数	data：需要加密的 NSData toFile: 加密后写入的文件位置
返回值	无

+(void) encryptContentsOfFile: (NSString *)file toFile:(NSString *)toFile;

说明	加密文件
参数	file：需要加密的文件 toFile: 加密后写入的文件位置
返回值	无

4.4.3 解密数据

+(NSData*) dencryptDataFromFile:(NSString *)filePath;

说明	解密数据并写入文件
参数	toFile: 加密后写入的文件位置
返回值	data：需要加密的 NSData

+(void) decryptContentFile:(NSString *) file toFile:(NSString *)filePath;

说明	解密文件并写入文件
参数	file：需要加密的文件 toFile: 加密后写入的文件位置
返回值	无

4.5 MCM 接口

所属类：

MAMCMRequest

4.5.1 getDocumentList 获取个人文档列表

方法名： -(void)getDocumentList

功能说明： 获取个人文档列表

参数说明：

参数名	类型	描述
version	string	协议版本
folderid	string	文件夹 id 当为-1 时，获取根企业目录

order	string	0 大小 1 时间 2 名称 默认 2
type	string	获得文档类型 1:企业文档 2:个人文档 3：共享文档,当无该字段时,默认为 1

返回值：无返回值

4.5.1.1 获取个人文档列表回调函数

方法名：

需实现方法
-(void)callbackNetworkRequestSuccessed:(NSDictionary*)rspData
-(void)callbackNetworkRequestFailed(NSDictionary*)error

功能说明：获取个人文档列表回调函数

方法说明：

参数名	类型	描述
resultcode	string	返回码 -1：系统正忙，请稍后再试！
resultmessage	String	返回码说明
documentlist	DocumentList[DocumentInfo]	用户文件列表
folderlist	FolderList[FolderInfo]	文件夹列表

返回值：无返回值

4.5.1.2 DocumentInfo 类

类说明：从服务端解析到的文档信息（字典类）

文件详情
属性说明：

属性名	类型	描述
documentname	String	文件名称
documentid	String	文件 id
type	String	jpg、png….
folderid	String	上级文件夹 ID
documenturl	String	文件下载地址

previewurl	String[]	预览地址
downloadflag	String	0 禁止下载 1 允许下载
preview	String	0 无法预览 1 可以预览
createtime	String	文件创建时间
size	String	文件大小，单位 K，客户端请自行转换
documentrole	string	文件权限(继承上级文件夹) 0 查看权限 1 管理权限（高级操作）
documentdesc	String	文件摘要
documentcreater	String	文件上传用户

4.5.1.3 FolderInfo 类

类说明：从服务端解析到的文件夹信息（字典类）

文件夹详情

属性说明：

属性名	类型	描述
folderid	String	文件夹 ID
foldername	String	文件夹名称
createtime	String	创建时间 YYYY-MM-DD HH-mm-ss
foldertype	string	文件夹类型 0 机构文件夹 1 普通文件夹
folderrole	string	文件夹权限 0 查看权限 1 管理权限（高级操作）
isshare	string	是否共享

4.5.2 getDocDownloadUrlOrPreviewUrl 获取文件下载地址或预览地址

方法名： -(void) getDocDownloadUrlOrPreviewUrl:(NSString*)docID

功能说明： 该请求由 System App 向 EMP Manager 发起，用户在 System App 中获取文件的下载地址或预览地址

参数说明：

参数名	类型	描述
version	string	协议版本
documentid	string	文档 ID
operatortype	string	操作类型，1-获取文档下载地址，2-获取文档预览地址
type	string	获得文档类型 1:企业文档 2:个人文档,当无该字段时,默认为 1

返回值：无返回值

4.5.2.1 获取文件下载地址或预览地址回调函数

方法名：

需实现方法

-(void)callbackNetworkRequestSuccessed:(NSDictionary*)rspData

-(void)callbackNetworkRequestFailed(NSDictionary*)error

功能说明：获取文件下载地址或预览地址回调函数

方法说明：

参数名	类型	描述
resultcode	string	返回码 -1：系统正忙，请稍后再试！
resultmessage	String	返回码说明
documenturl	String	文件下载地址
previewurl	String[]	预览地址

返回值：无返回值

4.5.3 uploadLocalFile 文件上传

方法名：-(void) uploadLocalFile

功能说明：该请求由 System App 向 EMP Manager 发起，用户在 System App 中获取文件的下载地址或预览地址

参数说明：

参数名	类型	描述

version	string	协议版本
folderid	string	文件夹 ID
foldertype	string	文件夹类型类型，1-企业文件，2-个人文件
filedepartment	string	文件部门

返回值：无返回值

4.5.3.1 文件上传回调函数

方法名：

需实现方法

-(void)callbackShowUploadProgress:(float)progress ----- 该回调函数用来返回上传进度

-(void)callbackUploadMCMFileSuccess ------ 上传成功回调函数

-(void)callbackUploadMCMFileFailed: (NSDictionary *)dic ----上传失败回调函数

功能说明：文件上传回调函数

返回值：无返回值

4.6 MAM 接口

4.6.1 应用更新检测

+(NSDictionary*)checkAppVersion:(NSString*)appID andAppVersion:(NSString*)appVersion;

说明	获取应用是否能更新
参数	appID:应用 ID　appVersion:应用版本号

返回值:

resultcode	返回码:0:成功 -1 其他错误
resultmessage	返回码说明(携带错误信息)
downloadurl	如果存在更高版本，则提供该信息，下载地址
filesize	文件大小 单位 K
updatescription	更新描述
updateflag	updateFlag 值为是存在更新（0 为非强制更新，1 为强制更新，2 为无需更新）
appversion	应用最新版本号

+(NSDictionary*)appInstallForIOS:(NSString*)appID andAppVersion:(NSString*)appVersion;

说明	获取应用是否能更新
参数	appID:应用 ID　appVersion:应用版本号

返回值:

resultcode	返回码:0:成功 -1 其他错误
resultmessage	返回码说明(携带错误信息)

4.7 应用启动手势密码

+(void)enterBackground;

说明	应用切换到后台需要添加此代码
参数	无
返回值	无

+(void)willBecomeActive;

看是否需要 SDK 弹出手势密码的界面

说明	应用切换到前台需要添加此代码
参数	无
返回值	无

5 技术支持

6 FAQ

6.1 手势密码使用注意事项

手势密码页面是强制竖屏

1.展示手势密码

+(BOOL)showGestureView;

基于[UIApplication sharedApplication].keyWindow上addSubview的方法

+(BOOL)showNavGestureView:(UIViewController*)view;

基于UIViewController上面加了navigationController上pushViewController的方法

返回值为YES表示有手势密码，返回NO表示无手势密码，需要第三方唤起门户到登录页面，重新输入登录设置手势密码

2.切换前台展示手势密码

+(void)willBecomeActive;

基于[UIApplication sharedApplication].keyWindow上addSubview的方法

+(void)willNavBecomeActive:(UIViewController*)view;

基于UIViewController上面加了navigationController上pushViewController的方法

+(void)willPresentBecomeActive;

获取[[UIApplication sharedApplication].keyWindow.rootViewController上的presentedViewController,然后手势密码就加载到最上面的View,通过presentViewController上去,以后作为标准方法提供.