移动应用框架集成框架(HiFrame)使用说明

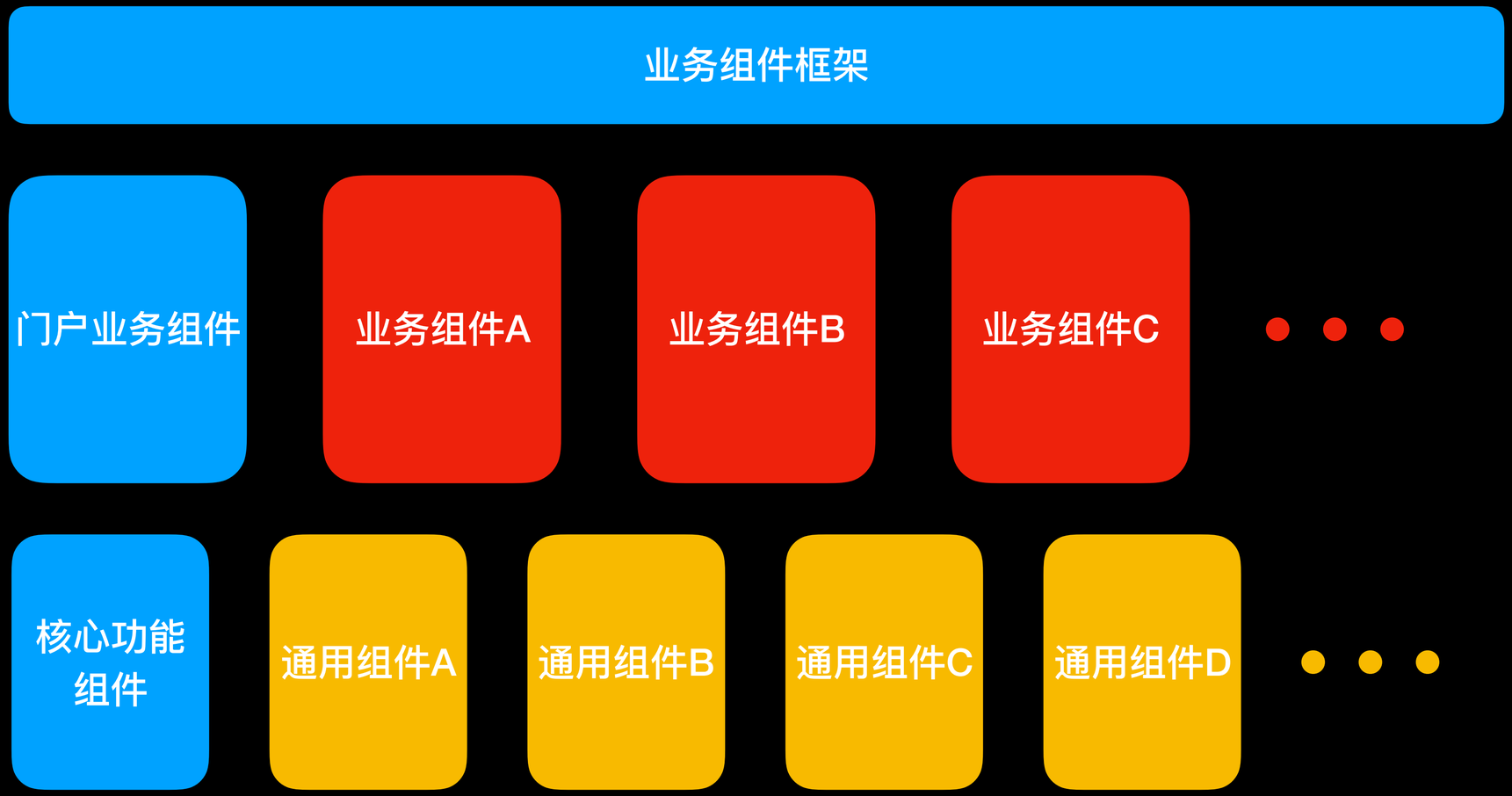
1.HiFrame框架结构介绍

2.iOS 如何通过 HiFrame 开发业务组件

3.核心功能组件主要功能介绍

HiFrame

框架结构



HiFrame

框架结构

HiFrame

框架结构

ModuleFramework

- ModuleManager // 加载和管理业务组件
- MenuManager // 注册与发现菜单，管理排序与加载
- Menu // 菜单项
- CellManager // 注册与发现Cell
- ApplicationDelegateProxy // 系统生命周期事件分发
- IApplicationDelegate // 生命周期事件协议
- IMenuDelegate // 菜单协议
- ICellDelegate // Cell协议

可见 不可见 协议

HiFrame

框架结构

HiFrame

框架结构

CoreFunctionModule

- AppUpgradeCheck // App版本监测
- Config // App配置项设置，如换肤主题
- Logger // 日志管理
- LoginBusiness // 登录管理
- MultiMediaManager // 多媒体管理
- NotificationReceiverBusiness // 推送接收管理
- TrafficStatisticsManager // 流量监控
- LoginProtocol // 自定义登录协议
- NotificationReceiverConnectionProtocol // 自定义消息推送协议协议

可见 协议

HiFrame

框架结构

HiFrame

框架结构

PortalModule

- LoginViewController // 登录门户
- MenuViewController // 菜单门户
- MineViewController // 我的门户
- SettingViewController //设置门户
- MediaViewController // 多媒体门户
- ICustomLoginDelegate // 生命周期事件协议
- IMineAccountCellDelegate // Cell协议
- ……

不可见 协议

1.HiFrame框架结构介绍

2.iOS 如何通过 HiFrame 开发业务组件

3.核心功能组件主要功能介绍

HiFrame

业务组件开发

1.通过模板创建业务组件

```
mayiming$ pod lib create B_OS_Play --template-url=https://git.hikvision.com.cn/scm/hik-system/moduletemplate.git
```

组件类型_事业部缩写_组件名称

B:业务组件　如海外为OS　如视频业务组件为Play

C:通用组件

模板地址

HiFrame

业务组件开发

1.通过模板创建业务组件

```
mayiming$ pod lib create B_OS_Play --template-url=https://git.hikvision.com.cn/scm/hik-system/moduletemplate.git
```

```
请输入所在事业部的缩写，例如[海外]则输入OS
 > OS
```

```
请输入事业部级GIT仓库地址(事业部cocoapods仓库地址)
 > https://git.hikvision.com.cn/scm/~xuchunyu/os-dev-ios-repo.git
```

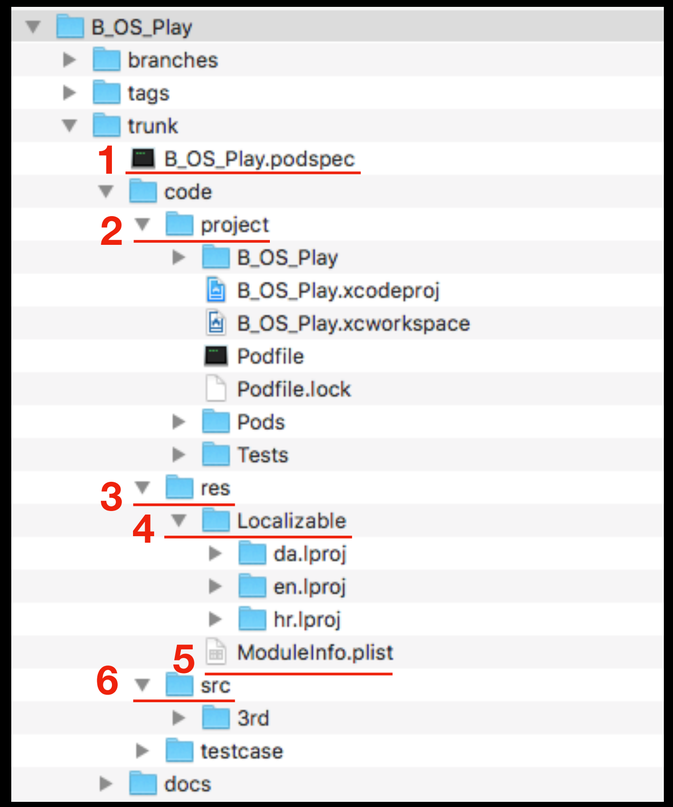
```
请输入你的名字作为你创建组件的作者
 > mayiming

请输入你的邮件，作为联系方式
 > mayiming@hikvision.com.cn
```

```
则请输入PodRoot的SVN地址全路径
 > https://192.0.0.140/APP-Client/iVMS5260/trunk/MAIF/iOS-Repos/OS-Dev-iOS-Repo
```

HiFrame

业务组件开发

业务组件目录结构



1.组件Pod描述文件

2.组件工程目录

3.组件资源目录

4.多语言文件

5.ModuleInfo文件

6.组件实现文件目录

Demo

HiFrame

业务组件开发

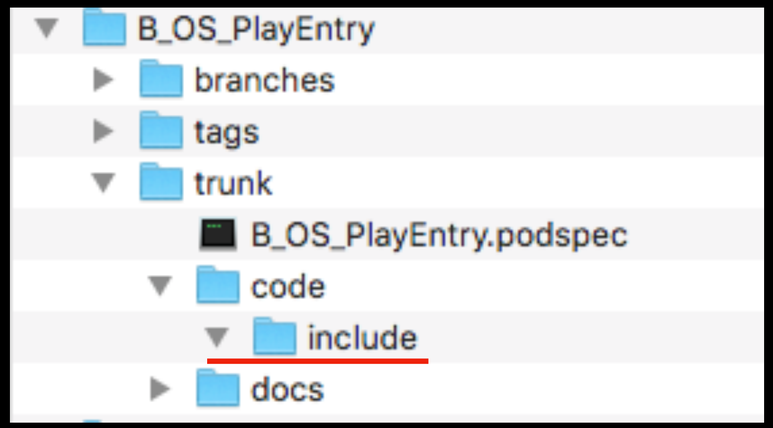
2.通过模板创建业务组件接口描述组件

```
mayiming$ pod lib create B_OS_PlayEntry --template-url=https://git.hikvision.com.cn/scm/hik-system/moduletemplate.git
```

组件类型_事业部缩写_(组件名称 + Entry)

模板地址

删除工程目录等无用文件（后续版本模板会支持直接创建接口描述组件）



Demo

HiFrame

业务组件开发

3.创建业务组件接口描述组件头文件

OSIPlayEntry.h

```objc
//
//  OSIPlayEntry.h
//  Pods
//
//  Created by mayiming on 2017/7/31.
//
//

#import <Foundation/Foundation.h>

@protocol OSIPlayEntry <NSObject>

- (BOOL)playWithUrl:(NSString *)urlStr;

@end
```

4.发布业务组件接口描述组件

HiFrame

业务组件开发

5.业务组件依赖业务组件接口描述组件

B_OS_Play.podspec

```
Pod::Spec.new do |s|

  ...

  s.subspec 'dev_repo' do |devs|
    #源码模式
    if ENV['mode'].to_s == "source" || ENV['mode'].to_s == "B_OS_Play"

      ...

      devs.ios.dependency 'B_OS_PlayEntry'

    ...

end
```

HiFrame

业务组件开发

6.业务组件实现接口协议

OSPlayEntry.m

```
#import "OSPlayEntry.h"
#import <B_OS_PlayEntry/OSIPlayEntry.h>

@interface OSPlayEntry () <OSIPlayEntry>

@end

@implementation OSPlayEntry

- (BOOL)playWithUrl:(NSString *)urlStr
{
    // 处理业务逻辑
}

@end
```

HiFrame

业务组件开发

7.定义ModuleInfo.plist

Key	Type	Value
▼ Information Property List	Dictionary	(1 item)
▼ MenuInfo	Array	(2 items)
▼ Item 0	Dictionary	(2 items)
MenuNameKey	String	OS_Play_Menu_Name_Preview
MenuImageKey	String	OS_Play_Menu_Image_Preview
▼ Item 1	Dictionary	(2 items)
MenuNameKey	String	OS_Play_Menu_Name_Playback
MenuImageKey	String	OS_Play_Menu_Image_Playback

事业部_组件名_Menu_Type(Name\Image)_菜单名

HiFrame

业务组件开发

8.实现IApplicationDelegate

HiIApplicationDelegate.h

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

@protocol HiIApplicationDelegate <NSObject>

@optional

// Module life cycle
- (void)applicationDidLoad;

// UIApplicationDelegate
- (void)applicationWillResignActive:(UIApplication *)application;
- (void)applicationDidEnterBackground:(UIApplication *)application;
- (void)applicationWillEnterForeground:(UIApplication *)application;
- (void)applicationDidBecomeActive:(UIApplication *)application;
- (void)applicationWillTerminate:(UIApplication *)application;
- (void)applicationDidReceiveMemoryWarning:(UIApplication *)application;

- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo fetchCompletionHandler:(void (^)(UIBackgroundFetchResult result))completionHandler;
- (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken;
- (void)application:(UIApplication *)application didFailToRegisterForRemoteNotificationsWithError:(NSError *)error;

@end
```

HiFrame

业务组件开发

8.实现IApplicationDelegate

HilApplicationDelegate.h

业务组件初始化

系统App生命周期事件，通知事件等

```
 8 // Module life cycle
 9 - (void)applicationDidLoad;
10
11 // UIApplicationDelegate
12 - (void)applicationWillResignActive:(UIApplication *)application;
13 - (void)applicationDidEnterBackground:(UIApplication *)application;
14 - (void)applicationWillEnterForeground:(UIApplication *)application;
15 - (void)applicationDidBecomeActive:(UIApplication *)application;
16 - (void)applicationWillTerminate:(UIApplication *)application;
17 - (void)applicationDidReceiveMemoryWarning:(UIApplication *)application;
18
    ...
```

HiFrame

业务组件开发

9.实现菜单加载、跳转（发现模式）

HilMenuDelegate.h

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

NS_ASSUME_NONNULL_BEGIN

@protocol HiIMenuDelegate <NSObject>

- (nullable UIViewController *)viewControllerWithMenuNameKey:(NSString *)menuNameKey;

@end

NS_ASSUME_NONNULL_END
```

HiFrame

业务组件开发

9.实现菜单加载、跳转（注册模式）

HilMenuManager.h

```objc
#import <Foundation/Foundation.h>

NS_ASSUME_NONNULL_BEGIN

@class HiMenu;

@interface HiMenuManager : NSObject

+ (instancetype)sharedInstance;

- (nullable NSArray<HiMenu *> *)allMenus;
- (void)filterMenusWithOrderedMenuNameKeys:(NSArray<NSString *> *)menuNameKeys;
- (void)filterMenusWithOrderedModuleNames:(NSArray<NSString *> *)moduleNames;

- (nullable UIViewController *)viewControllerWithMenuNameKey:(NSString *)menuNameKey;

// 注册模式，  注册成功返回YES，  失败返回NO
- (BOOL)registerMenuNameKey:(NSString *)menuKey
             viewController:(UIViewController *)viewController;

@end

NS_ASSUME_NONNULL_END
```

HiFrame

业务组件开发

9.实现菜单加载、跳转（注册模式）

HilMenuManager.h

```
// 注册模式，  注册成功返回YES，  失败返回NO
- (BOOL)registerMenuNameKey:(NSString *)menuKey
                 viewController:(UIViewController *)viewController;
```

```
[[HiMenuManager sharedInstance] registerMenuNameKey:@“OS_Play_Menu_Name_Preview”
                                     viewController:previewViewcontroller];
```

HiFrame

业务组件开发

10.设置项加载（发现模式）

HiICellDelegate.h

```
1  #import <Foundation/Foundation.h>
2  #import <UIKit/UIKit.h>
3
4  @protocol HiICellDelegate <NSObject>
5
6  - (NSArray<UITableViewCell *> *)cellArrayWithMenuNameKey:(NSString *)menuNameKey;
7
8  @end
9
10 @protocol HiISettingCellDelegate <HiICellDelegate>
11
12 @end
13
14 @protocol HiIMineCellDelegate <HiICellDelegate>
15
16 @end
```

设置页面设置项

我的页面设置项

HiFrame

业务组件开发

10.设置项加载（注册模式）

HiCellManager.h

```objc
#import <Foundation/Foundation.h>

NS_ASSUME_NONNULL_BEGIN

@interface HiCellManager : NSObject

+ (instancetype)sharedInstance;

- (nullable NSArray<UITableViewCell *> *)cellsForMenuNameKey:(NSString *)menuNameKey
                                    withCellDelegateProtocol:(Protocol *)cellDelegateProtocol;

// 注册模式， 注册成功返回YES， 失败返回NO
- (BOOL)registerCells:(NSArray<UITableViewCell *> *)cells forMenuNameKey:(NSString *)menuNameKey
                                              withCellDelegateProtocol:(Protocol *)cellDelegateProtocol;

@end

NS_ASSUME_NONNULL_END
```

HiFrame

业务组件开发

10.设置项加载（注册模式）

HiCellManager.h

```
12 // 注册模式，  注册成功返回YES，  失败返回NO
13 - (BOOL)registerCells:(NSArray<UITableViewCell *> *)cells
           forMenuNameKey:(NSString *)menuNameKey
withCellDelegateProtocol:(Protocol *)cellDelegateProtocol;
```

HiFrame

业务组件开发

10.设置项加载（注册模式）

```objc
// Mine cell
[[HiCellManager sharedInstance] registerCells:previewMineCells
                                forMenuNameKey:@"OS_Play_Menu_Name_Preview"
                       withCellDelegateProtocol:@protocol(HiIMineCellDelegate)];
[[HiCellManager sharedInstance] registerCells:playbackMineCells
                                forMenuNameKey:@"OS_Play_Menu_Name_Playback"
                       withCellDelegateProtocol:@protocol(HiIMineCellDelegate)];

// Setting cell
[[HiCellManager sharedInstance] registerCells:previewSettingCells
                                forMenuNameKey:@"OS_Play_Menu_Name_Preview"
                       withCellDelegateProtocol:@protocol(HiISettingCellDelegate)];
[[HiCellManager sharedInstance] registerCells:playbackSettingCells
                                forMenuNameKey:@"OS_Play_Menu_Name_Playback"
                       withCellDelegateProtocol:@protocol(HiISettingCellDelegate)];
```

HiFrame

业务组件开发

11.组件间方法调用

HiModuleManager.h

```objc
#import <Foundation/Foundation.h>

NS_ASSUME_NONNULL_BEGIN

@interface HiModuleManager : NSObject

+ (instancetype)sharedInstance;

- (void)loadModules;
- (nullable id)newObjectWithProtocol:(Protocol *)protocol;
- (nullable NSArray *)newObjectsWithProtocol:(Protocol *)protocol;
- (nullable NSString *)pathForCurrentModuleLibraryDirectory;
- (nullable NSString *)pathForCurrentModuleDocumentDirectory;

@end

NS_ASSUME_NONNULL_END
```

HiFrame

业务组件开发

11.组件间方法调用

获取找到的第一个实现该协议的类，创建新对象

```
10 - (nullable id)newObjectWithProtocol:(Protocol *)protocol;
11 - (nullable NSArray *)newObjectsWithProtocol:(Protocol *)protocol;
```

获取所有实现该协议的类，创建新对象

HiFrame

业务组件开发

11.组件间方法调用

业务组件B_OS_Park中的实现

```objc
#import <C_HI_Framework/HiModuleManager.h>
#import <B_OS_PlayEntry/OSIPlayEntry.h>

...

- (BOOL)playParkVideoWithUrl:(NSString *)url {
    id<OSIPlayEntry> playEntry = [[HiModuleManager sharedInstance]
                                  newObjectWithProtocol:@protocol(OSIPlayEntry)];
    return [playEntry playWithUrl:url];
}
```

HiFrame

业务组件开发

12.组件沙盒访问

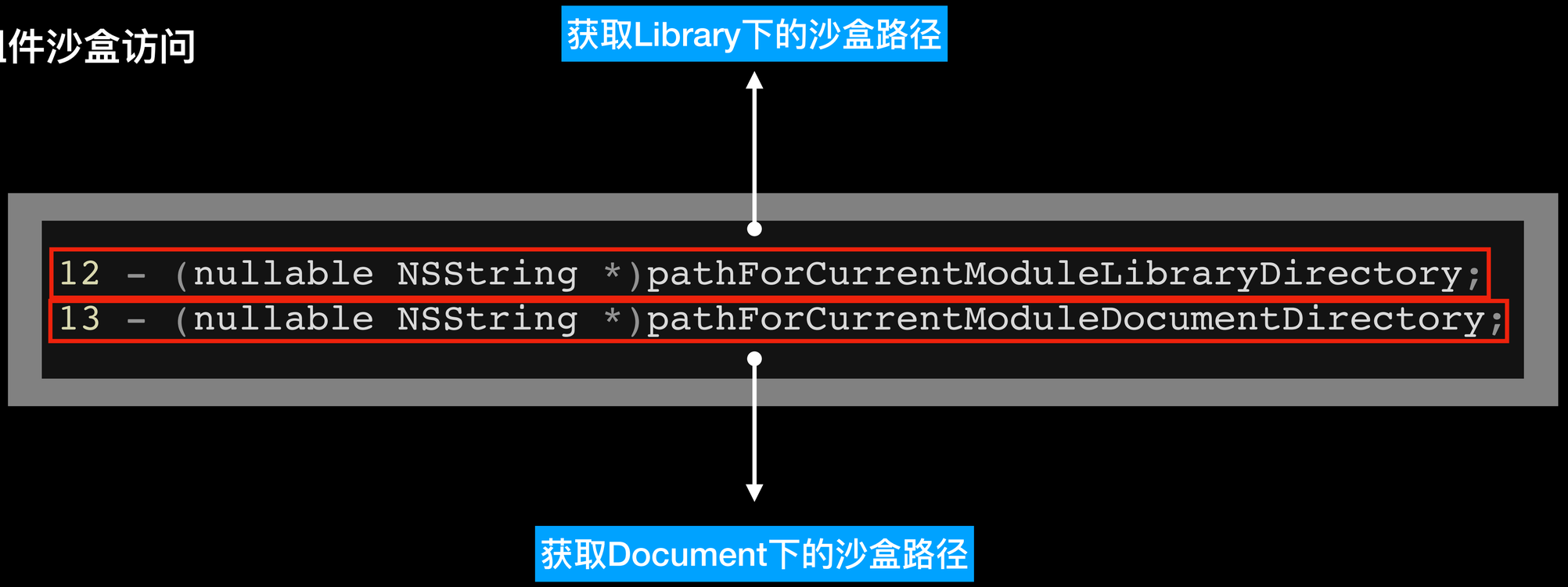
HiModuleManager.h

```objc
#import <Foundation/Foundation.h>

NS_ASSUME_NONNULL_BEGIN

@interface HiModuleManager : NSObject

+ (instancetype)sharedInstance;

- (void)loadModules;
- (nullable id)newObjectWithProtocol:(Protocol *)protocol;
- (nullable NSArray *)newObjectsWithProtocol:(Protocol *)protocol;
- (nullable NSString *)pathForCurrentModuleLibraryDirectory;
- (nullable NSString *)pathForCurrentModuleDocumentDirectory;

@end

NS_ASSUME_NONNULL_END
```

HiFrame

业务组件开发

12.组件沙盒访问



PS:iOS 还会提供一个获取Temporary目录沙盒路径的接口

HiFrame

业务组件开发

13.错误码机制

HiErrorManager.h

```objc
#import <Foundation/Foundation.h>


NS_ASSUME_NONNULL_BEGIN

@interface HiError : NSObject

@property(nonatomic, copy) NSString *module;
@property(nonatomic, assign) int code;
@property(nonatomic, copy) NSString *description;

@end

@interface HiErrorManager : NSObject

+ (void)setLastError:(NSString *)module
                code:(int)code
         description:(NSString *)description;

+ (nullable HiError *)getLastError;

@end

NS_ASSUME_NONNULL_END
```

1.HiFrame框架结构介绍

2.iOS 如何通过 HiFrame 开发业务组件

3.核心功能组件主要功能介绍

HiFrame

业务组件开发

1.登录信息管理

```objc
@protocol HiLoginBusinessDelegate <NSObject>

- (void)beforeLogin;
- (void)loginFinishedFaild;
- (void)loginFinishedSuccess;
- (void)beforeLogout;
- (void)logoutFinished;
- (void)keepLiveFaild;

@end
```

```objc
@interface HiLoginInfo : NSObject

@property(nonatomic, readonly, copy) NSString *userName;//登录用户名
@property(nonatomic, readonly, copy) NSString *userID;//登录密码
@property(nonatomic, readonly, copy) NSString *clientTicketGrantedTicket;//CTGT
@property(nonatomic, readonly, copy) NSString *portalServiceAddress;//门户服务地址
@property(nonatomic, readonly) unsigned short portalServerPort;//门户服务端口
@property(nonatomic, readonly) int globalPswStrength;//密码等级
@property(nonatomic, readonly) NSString *coreServiceAddress;//核心服务地址

@end
```

HiFrame

业务组件开发

1.登录信息管理

```objc
@interface HiLoginBusiness : NSObject

@property(readonly, strong) HiLoginInfo *loginInfo;

...

+ (instancetype)sharedInstance;

...

- (void)registerLoginDelegate:(id<HiLoginBusinessDelegate>)delegate;
- (void)unregisterLoginDelegate:(id<HiLoginBusinessDelegate>)delegate;

@end
```

HiFrame

业务组件开发

2.多媒体文件管理

```objc
@interface HiMultiMediaManager : NSObject
+ (instancetype)sharedInstance;

/**
 根据tab进行注册，某个业务组件只有注册以后，才能在门户（多媒体管理）组件中展示多媒体数据
 @param tab 和各业务组件一一对应的唯一的常量
 */
- (BOOL)registerTab:(NSString *)tab;

/**
 保存多媒体文件
 @param param 创建多媒体文件需要的参数
 */
- (HiMediaFile *)saveMediaFile:(MediaFileParam *)param;

/**
 删除某个多媒体文件
 */
- (BOOL)removeMediaFile:(HiMediaFile *)mediafile;

/**
 获取tab下的所有多媒体文件数组
 */
- (NSArray<HiMediaFile *> *)mediafilesWithTab:(NSString *)tab;

@end
```

HiFrame

业务组件开发

3.日志管理

```
#define HiVerbose(logtag, fmt, ...)
#define HiVerbose_if(exp, logtag, fmt, ...)

#define HiDebug(logtag, fmt, ...)
#define HiDebug_if(exp, logtag, fmt, ...)

#define HiInfo(logtag, fmt, ...)
#define HiInfo_if(exp, logtag, fmt, ...)

#define HiWarn(logtag, fmt, ...)
#define HiWarn_if(exp, logtag, fmt, ...)

#define HiError(logtag, fmt, ...)
#define HiError_if(exp, logtag, fmt, ...)

#define HiFatal(logtag, fmt, ...)
#define HiFatal_if(exp, logtag, fmt, ...)
```

HiFrame

业务组件开发

4.流量统计

```objc
@interface HiTrafficStatisticsManager : NSObject

+ (instancetype)sharedInstance;

/**
 增加流量值
 */
- (void)addTraffic:(unsigned long long)traffic;

@end
```

HiFrame

业务组件开发

必须实现的

- 必须定义ModuleInfo.plist文件（默认已经创建，需修改MenuInfo项，以及添加PrivacyKey）
- 如果需要初始化业务组件以及监听App生命周期事件等，则必须实现IApplicationDelegate协议
- 菜单与设置项加载如果选用注册模式，则必须在- (void)applicationDidLoad方法中添加注册代码
- 菜单与设置项加载如果选用发现模式，则必须实现IMenuDelegate与ICellDelegate（对应的子协议）

必须遵守的

- 事业部缩写必须全大写
- 组件名必须为“组件类型_事业部缩写_组件名称” 如：B_OS_Play
- 业务组件接口描述组件命名必须为“组件类型_事业部缩写_组件名称+Entry” 如：B_OS_PlayEntry
- 菜单NameKey命名必须为“事业部缩写_业务组件名_菜单名” 如：OS_Play_Preview
- 多媒体管理注册的Tab名必须为“事业部缩写_业务组件名_Tab名” 如：OS_Play_Preview
- 多语言文件必须放在模板创建的code/res/Localizable目录下

建议这样做的

- 实现IApplicationDelegate的类命名为：事业部缩写+业务组件名+ApplicationDelegate（如OSPlayApplicationDelegate）
- 菜单与设置项的加载，建议发现模式与注册模式只选择其中一种

Thanks