91移动开发平台SDK(完全版)参考手册

版本修改记录

时间	版本	更新说明
2011-10-09	3.1.2	新建参考手册
2011-11-21	3.1.3	新增 91 豆充值和代币充值，修改测试模式相关描述
2012-1-5	3.1.5	代币充值接口调整
2012-2-8	3.1.5.1	适配 iOS4.3.x 出现的点击好友崩溃问题
2012-2-20	3.1.6	支付中心模块优化
2012-3-15	3.1.7	1.增加绑定手机号功能 2.支持 GIF 文件分享到第三方平台
2012-4-11	3.1.8	新增游客登录功能,游客账号转正，切换账号功能，捕获会话过期的通知 修改登录结果通知的代码示例 新增使用场景案例 修改版本号设定规则说明
2012-6-6	3.2.0	增加软件互推功能说明 增加获取应用图标接口
2012-6-21	3.2.0	回调接口 NdQueryAppPromotionAwardListDidFinish 修改
2012-6-29	3.2.1	NdComPlatform_SNS_Resources 资源改为 bundle 形式 整理 SDK 开发环境搭建说明
2012-8-9	3.2.1	整理登录/注销文档模块， 增加账号管理 API 说明， 增加取消本地通知 API 说明。 增加支付模式时，对多服务器充值的说明 增加对 XCode 4.x 下的开发环境配置说明
2012-9-3	3.2.2	增加软件互推界面展示的接口 更新了该文档的目录结构
2012-11-14	3.2.3	用户修改头像、个人资料时，增加通知消息 SDK 舍弃 armv6 架构（armv6 = iPhone 2G/3G, iPod 1G/2G） 支持 armv7 和 armv7s 架构，固件要求 iOS4.0+
2013-03-10	3.2.3.2	增加 MessageUI.framework 依赖，iOS6.0 优化适配
2013-4-18	3.2.5	初始化方法变更为 NdInit；增加暂停页 API；新增接入流程简要描述
2013-7-11	3.2.5.1	增加 iOS6.0 适配常见问题解决方案
2013-7-15	3.2.6	增加悬浮工具条，去除一些废弃的 API
2013-11-18	3.2.6.3	增加 AssetsLibrarys.framework，AdSupport.framework 依赖

目　录

版本修改记录.....2
目　录.....3
一、 SDK 构成.....5
二、 SDK 接入流程简要描述.....5
三、 SDK 开发环境搭建.....5
1. 工程配置.....5
2. FrameWork 的添加.....6
3. 资源文件的添加.....9
四、 91SDK 基础功能.....9
1. 导入头文件.....9
2. 初始化.....10
3. 设定调试模式.....11
4. 检查更新.....13
5. 显示工具条.....14
6. 捕获退出平台界面的通知.....15
7. 设定平台界面方向.....15
8. 登录/注销.....16
1）普通登录.....16
2）登录（支持游客登录）.....16
3）游客账户转正式账户.....17
4）判断用户登录状态.....17
5）切换账户.....17
6）登录结果通知.....18
7）注销.....21
9. 用户反馈.....21
10. 显示暂停页.....21
11. 进入平台中心.....22
12. 应用内购买.....22
1）如何使用同步购买.....22
2）如何使用异步购买.....27
3）代币充值.....31
4）指定服务器充值.....32
13. URL Scheme 设置.....33
五、 91SDK 扩展功能.....35
14. 分享到第三方平台.....35
15. 捕获会话过期的通知.....35
16. 进入平台中心各模块.....36
1）进入好友中心.....36
2）进入指定用户的空间.....36

3）进入游戏大厅.....36
4）进入指定应用的主页.....36
5）进入设置界面.....37
6）进入邀请好友界面.....37
7）进入应用论坛界面.....37
17. 虚拟商店.....37
1）简介.....37
2）进入虚拟商店.....38
3）获取虚拟商品类别.....39
4）获取应用促销信息.....40
5）获取商店里的商品信息列表.....42
6）购买虚拟商品.....43
7）使用已购买的虚拟商品.....45
8）已购买的非消费型商品查询.....47
9）已购买的订阅型商品查询.....48
10）获取已购买的指定商品信息.....49
11）获取已购买的商品信息列表.....52
12）查询游戏币余额.....54
18. 获取平台数据信息.....55
1）获取当前应用的玩家列表.....55
2）获取当前应用的我的好友列表.....55
3）获取我的好友列表.....56
4）获取我的信息.....57
5）捕获个人信息修改的通知.....57
6）获取用户的详细信息.....58
19. 好友操作.....58
1）给好友发送消息.....58
2）添加删除好友.....60
20. 获取头像/图标.....61
1）简介.....61
2）获取好友的头像.....61
3）获取好友的头像（缓存文件）.....63
4）获取默认头像、默认应用图标.....64
5）获取榜图标.....64
6）获取应用图标.....65
7）获取应用图标（缓存文件）.....66
21. 屏幕截图.....66
六、 版本号设定规则.....67
七、 FAQ.....68
八、 场景案例.....69
1. 游客登录.....69

一、 SDK 构成

91 移动开放平台库主要由以下几部分构成：

1. Framework
   从 SDK3.2.3 版本开始，只支持 armv7 和 armv7s 架构，固件要求 iOS4.0 +
2. 资源文件 NdComPlatform_SNS_Resource.bundle
   iOS 平台使用到的资源文件都在这里，需要添加到工程中。
3. 使用 91 手机平台，请在你的 app 的图标上打上 91 平台的标志，资源文件是开发包中的 91Logo 目录。

SDK 支持平台

91 移动开发平台支持 iPod Touch，iPhone，iPad，要求 armv7 或 armv7s 架构，操作系统 Mac 要求 Lion 以上，Xcode 要求 4.2 以上，iOS 要求 iOS4.0+。

二、 SDK 接入流程简要描述

1 搭建 SDK 的环境，导入 SDK 的必要文件，参见 SDK 开发环境搭建
2 导入必要的头文件，初始化 SDK，参见初始化。（初始化 API 中已包含版本更新检测）
3 如果需要的话，可以在初始化中设置 SDK 的平台方向，也可以稍后单独调用 SDK 的设置平台方向接口，原则要求设置平台方向与游戏方向一致
4 捕捉到初始化完成的通知后，根据需要调用登录，参见登录/注销
5 支付流程请参照应用内购买
6 在游戏进入暂停或者游戏从后台恢复的时候，需要调用暂停页接口，参见显示暂停页
7 需要为应用增加一个 91 的 URL Scheme，参见 URL Scheme 设置
8 主界面中要显示 91 工具条，详细 API 参见显示工具条

三、 SDK 开发环境搭建

1. 工程配置

Xcode->Project->Edit Project Settings，打开你的工程配置

A. 添加库的链接参数

在工程配置里头，找到 Linking 部分，修改 **Other Linker Flags**，添加以下内容：

```
-ObjC
```

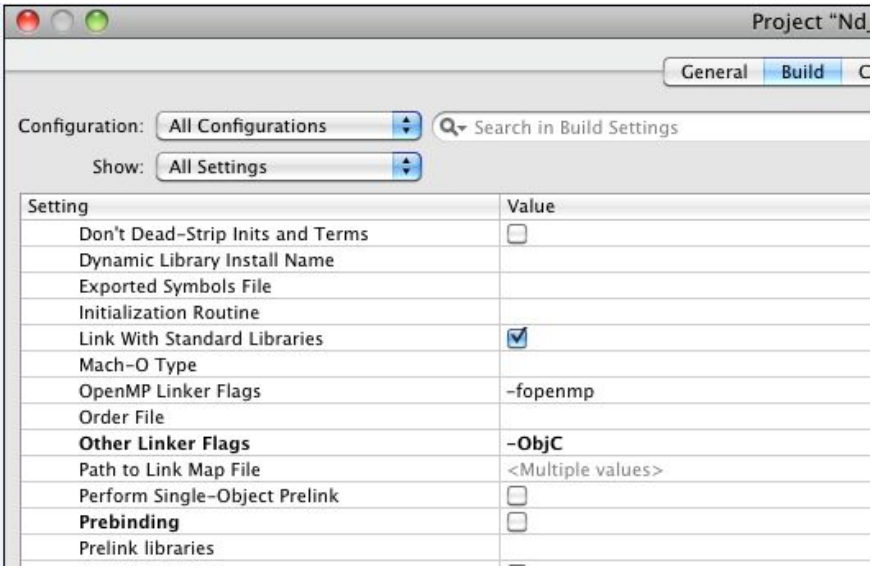


图 2-1-1　Xcode 3.x Other Linker Flags 示意图

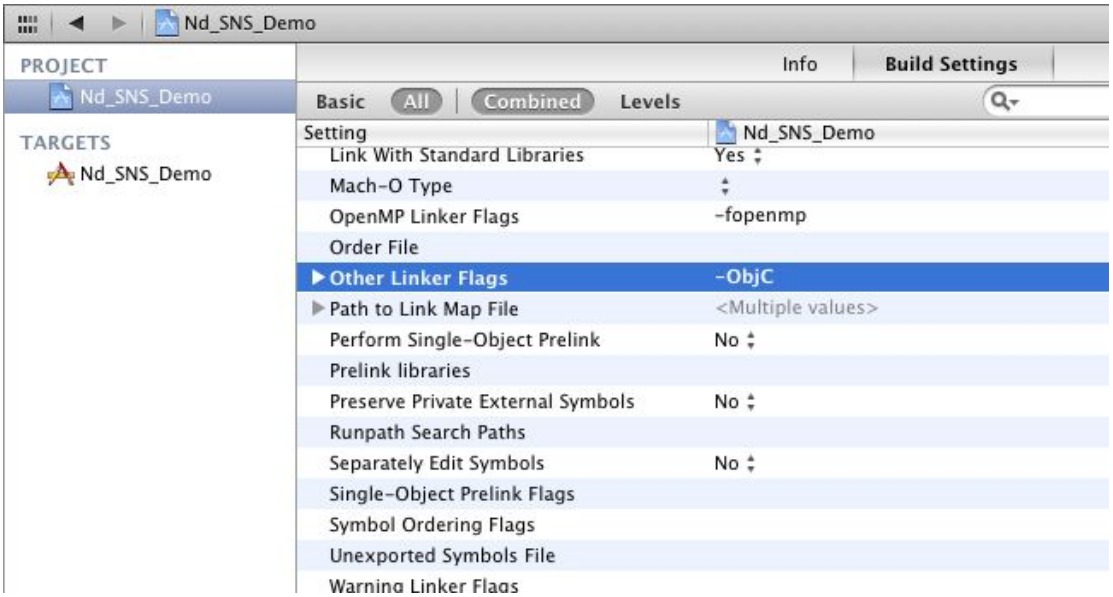


图 2-1-2　Xcode 4.x Other Linker Flags 示意图

注意：

如果你有更改过或使用的是 Target 中的配置，那么需要在 Target 的配置中作同样的修改！

2. FrameWork 的添加

1）添加系统 framework 以及库

请在你的工程中添加以下 FrameWork：

AddressBook.framework

CoreGraphics.framework

CoreTelephony.framework

QuartzCore.framework

SystemConfiguration.framework
UIKit.framework
Foundation.framework
MessageUI.framework
AssetsLibrary.framework
AdSupport.framework
还要添加以下 dylib：
libsqlite3.dylib
libz.dylib
添加方式与系统其它的 Framework 方式相同。

添加完后，请将 target 配置中的 CoreTelephony.framework 和 UIKit.framework 设置为 weak（在 Xcode4.x 上选择 optional）。

（双击 target，选中 general 即可看到所有链接库及链接类型）

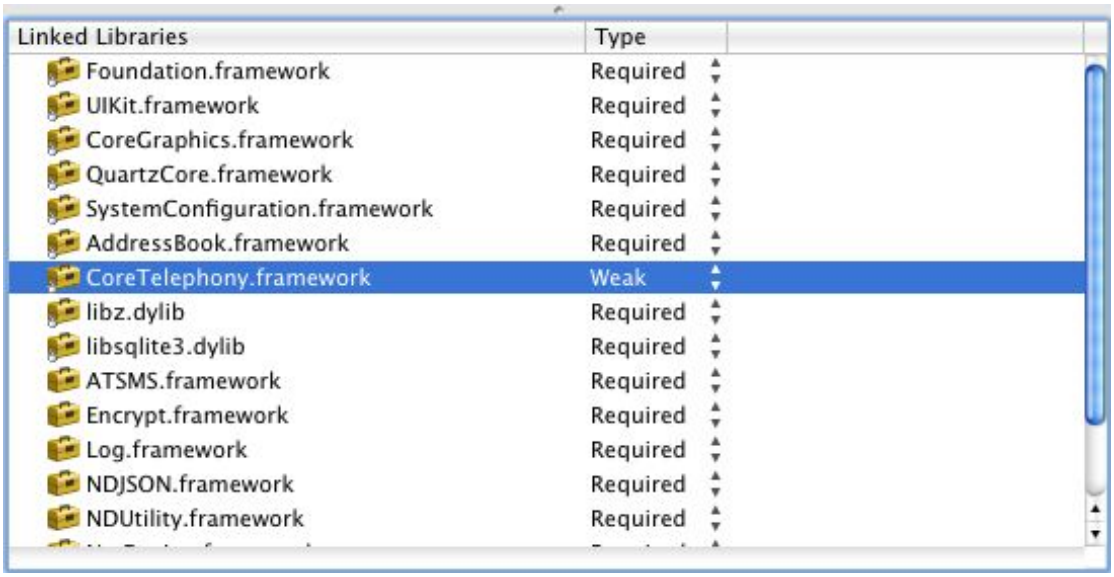


图 2-2-1　Xcode3.2.3 下的链接库及链接类型

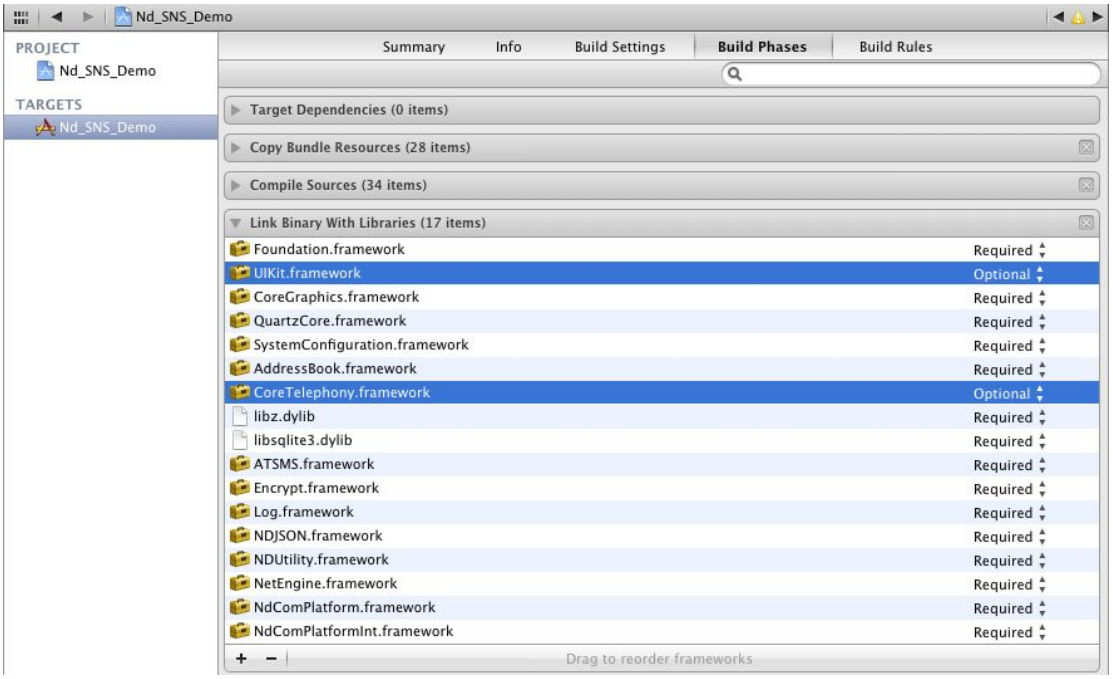


图 2-2-2　Xcode 4.2 下的链接库及链接类型

2）添加 91SDK 的 framework

将 91SDK 提供的 framework4 拖到工程 Groups & Files 面板中的 Frameworks 组中里，在弹出的选择框里，选择“create groups”的添加方式。

以下展示 Xcode3 和 Xcode4 的截图详情：

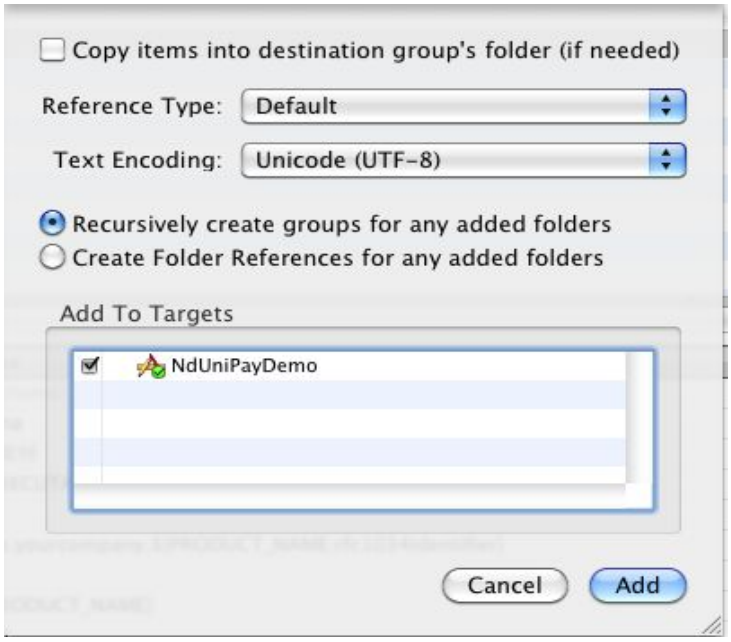


图 2-2-3　Xcode3.2.3 下的 create groups 选项

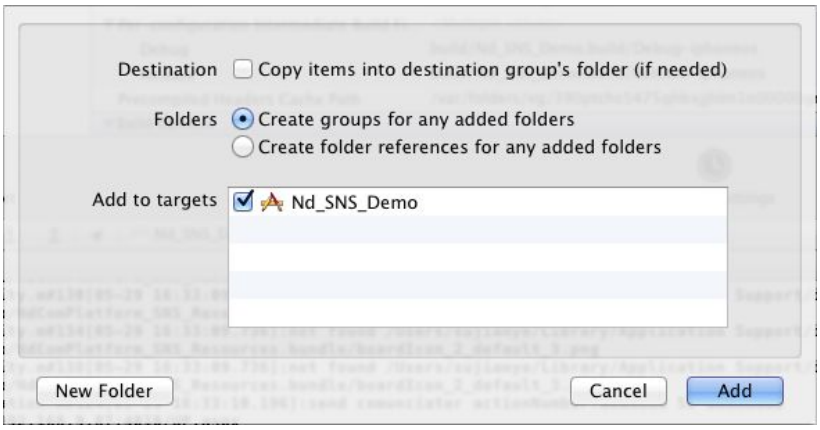


图 2-2-4　Xcdoe4.3.2 下的 create groups 选项

最终的 framework 配置如下图：

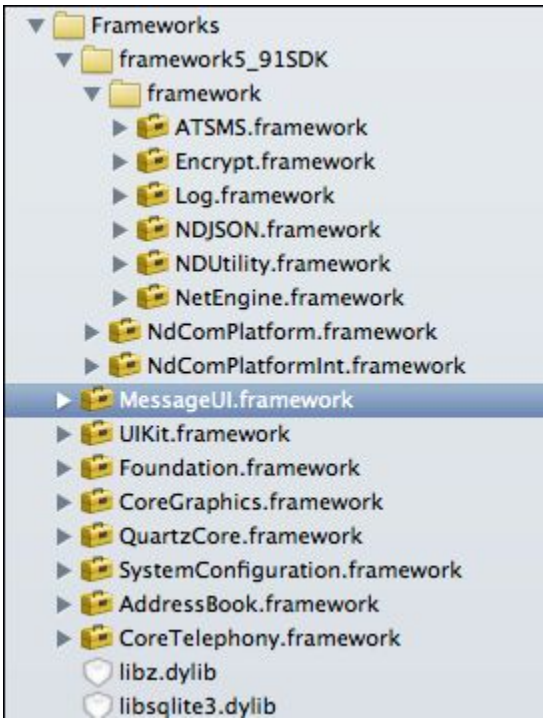


图 2-2-6　Xcode 4.2 用 create groups 添加方式添加后的 framework5

3. 资源文件的添加

把 91SDK 中的资源文件 NdComPlatform_SNS_Resources.bundle 拖到你的工程 Groups & Files 面板中的 Resources 组里，在弹出的选择框里，选择“create groups”的添加方式。

参见图 2-2-3 和图 2-2-4 中的 create groups 详情截图

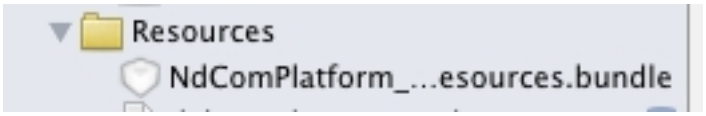


图 2-2-8　Xcode 4.2 用 create groups 添加方式添加后的 NdComPlatform_SNS_Resources.bundle

四、 91SDK 基础功能

在配置好开发环境，导入 SDK 框架和资源后，您就可以开始接入 SDK 的功能。

1. 导入头文件

首先你需要导入 SDK 框架的头文件，你还可以参照如下方式添加更多的头文件以获取 SDK 更丰富的功能支持。

```
#import <NdComPlatform/NDComPlatform.h>
#import <NdComPlatform/NdComPlatformAPIResponse.h>
#import <NdComPlatform/NdCPNotifications.h>
```

2. 初始化

首先你需要在程序开始的地方，优先进行平台的初始化。

先创建一个 NdInitConfigure 类，设置 AppId 和 AppKey；将 NdInitConfigure 传给 NdInit，使用 NdInit 接口进行平台的初始化，初始化完成后，将会发送 kNdCPInitDidFinishNotification 通知。

初始化参数中的 AppId 和 AppKey，需要您在接入 91 移动开放平台之前，向开发者后台申请可接入的 AppId 和 AppKey。所有的 SDK 的操作都需要设置这两个参数才能够正常工作。

请务必将客户端用的 AppId 和 AppKey 转告服务器端开发人员，确保服务器端用的 AppId 和 AppKey 和客户端的保持一致。

NdInitConfigure 中的 versionCheckLevel 参数，用于表示版本检测的等级，默认值为 ND_VERSION_CHECK_LEVEL_STRICT，即如果版本检测失败，则不允许进入游戏；如果你不要求如此高的版本检测等级，可以将其设置为 ND_VERSION_CHECK_LEVEL_NORMAL，单机或弱联网一般设置为此级别。

其中的 orientation 参数，用于初始化平台方向，默认为空表示不设置。

注意：从 3.2.5 开始，接入者需要为应用配置一个格式为 91-xxxxx 的 URL Scheme(xxxxx 为软件唯一标识符)。具体配置可参照 URL Scheme 配置部分。

下面为初始化的代码示例

```
- (BOOL)application:(UIApplication *)application
                  didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
    [window makeKeyAndVisible];
    //完成您必要的初始化工作，例如设置您的 rootViewController
   //初始化平台
    NdInitConfigure *cfg = [[[NdInitConfigure alloc] init] autorelease];
    cfg.appid =100010;
    cfg.appKey =@"C28454605B9312157C2F76F27A9BCA2349434E546A6E9C75";
    //单机，弱联网必须关注 versionCheckLevel 的设置说明，详见上面说明
```

```
    cfg.versionCheckLevel = ND_VERSION_CHECK_LEVEL_STRICT;
    //orientation 的设置详见上面的说明(这里以设置竖屏为例)
    cfg.orientation = UIInterfaceOrientationPortrait;

    [[NdComPlatform defaultPlatform] NdInit:cfg];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(SNSInitResult:) name:(NSString
*)kNdCPInitDidFinishNotification object:nil];
   //...write you code here
    return YES;
}
```

监听到初始化完成，开始执行自己的工作：

```
- (void)SNSInitResult:(NSNotification *)notify
{
   //...执行应用的逻辑，例如显示工具条、登录
   //[[NdComPlatform defaultPlatform] NdShowToolBar:NdToolBarAtTopLeft];
 }
```

3. 设定调试模式

设定为调试模式

```
[[NdComPlatform defaultPlatform] NdSetDebugMode:0];
```

设定为调试模式的支付功能和升级功能，参数为预留，默认为零。

其中的支付功能和升级功能具体包括：

- 游戏版本的检测升级
- 91 豆余额查询和支付
- 代币充值中的 91 豆兑换
- 虚拟商店自定义虚拟币充值中的 91 豆兑换

开发者调用该接口后，SDK 将这些功能转换为测试环境。开发者需要到 http://dev.91.com 进行配置相关的测试数据（测试账号和余额，游戏升级的版本等）。

图示：

注意：这个方法只是您在开发阶段用户测试上面描述的功能用的，当您的游戏或者应用准备正式发布时切记将该方法的调用去掉。否则用户将无法进行支付及升级。

1：配置游戏升级版本

开发者配置后调用接口时，检测升级时将会后台配置的版本号进行比较，低于后台配置的版本时，将会返回一个默认升级包的下载地址。该测试功能，主要用于你的软件中使用了平台的版本检测与升级模块功能。 我们提供这样的功能， 主要是模拟已经有一个新版本发布， 用于测试当前的版本，将来可以正确的升级到新版本。

后台配置如图： 通常测试版本号，需要比本地版本号要高， 才能模拟版本升级

2：配置支付功能的测试账户和余额

支付功能的支付余额默认为 5000 个 91 豆，开发者需要在后台配置测试帐号，即可在测试环境使用。该功能仅提供支付测试，及支付结果通知，用于测试你的软件中的购买流程。支付功能不会产生支付流水，不产生消费记录，也不产生任何收益。后台配置的测试帐号，仅在测试模式下生效，在正式环境中，是作为正常的普通用户。

开发者可以在 dev 后台中，配置测试帐号和查阅测试订单。

4. 检查更新

简介

当开发者要发布新的客户端版本时，为了区分某些版本更新的重要性，可以在后台进行更新属性配置，有“强制更新”和“普通更新”两种。默认是普通更新

强制更新：当客户端发生了重要变更（如修复了某个严重 BUG）或者开发者想强行推广新版本时，可以在开发者后台里把新版本属性设置为强制更新。用户必须更新到最新版本，才允许使用。

普通更新：非强制更新的情况。用户不需要更新到最新版本，即在当前版本也能使用。

开发者可以在后台的【应用管理】中设置是否强制更新。

SDK 仅是告知是哪种形式的更新。开发者需要根据接收到的标识符进行相应的处理。因此我们建议开发者可以按以下方式处理版本更新。

- 当版本检测失败时，按无版本更新处理
- 当收到用户取消普通更新时，允许其登录等后续相关流程
- 当收到用户取消强制更新时，开发者不需要做任何处理，由 SDK 强制用户退出。

同时，如果您是首次接入的开发者或者需要验证更新的流程，您可以通过设置调试模式来进行版本升级的功能测试。具体方法请参见设置调试模式一节。

为了保证版本比较的准确性，需要严格按照标准定义版本号．具体参见本文档的版本号设定规则章节．

NdInit 在初始化时将自动完成检查更新的工作。

5. 显示工具条

为方便游戏用户访问91移动开发平台，我们提供了悬浮工具条的功能。具体的UI是一个91移动开发平台的按钮，点击后可以展开，上面有一些常用的功能。

调用下面的代码将在屏幕最顶层显示 SDK 的工具条。

```
[[NdComPlatform defaultPlatform] NdShowToolBar:NdToolBarAtTopLeft];
```

工具条初始化的位置有 6 种选择：

NdToolBarAtTopLeft(左上角)/NdToolBarAtTopRight(右上角)

NdToolBarAtMiddleLeft(左边居中)/NdToolBarAtMiddleRight(右边居中)

NdToolBarAtBottomLeft(左下角)/NdToolBarAtBottomRight(右下角)

如果想要隐藏工具条，则可以调用下面的代码：

```
[[NdComPlatform defaultPlatform] NdHideToolBar];
```

您可以在初始化完成之后就显示工具条，也可以在登录完成后再显示工具条。

6. 捕获退出平台界面的通知

现在在退出 91 移动平台的界面时， 将发送 **kNdCPLeavePlatformNotification** （ 在 NdCPNotifications.h 文件中声明）。用户可以通过捕获该消息来得知界面的退出。

在登录和注册页面的退出原先已有 kNdCPLoginNotification,故登录注册页面不再发送
kNdCPLeavePlatformNotification 的通知。

```
[[NSNotificationCenter defaultCenter] addObserver:self
                        selector:@selector(SNSLeaveComplatformUI:)
                        name:(NSString *)kNdCPLeavePlatformNotification
                        object:nil];
```

7. 设定平台界面方向

为了适应不同类型的应用接入，我们针对应用进行横屏和竖屏的 UI 支持。SDK 的初始默认方向为 UIInterfaceOrientationPortrait。在进入平台前，可以设定平台界面的横竖屏状态。进入平台界面时，使用最近一次设置过的横竖屏状态。

```
[[NdComPlatform defaultPlatform]
      NdSetScreenOrientation:UIInterfaceOrientationPortrait];
```

iPad 默认开启自动旋转，iPhone 默认关闭自动旋转。如果不需要自动旋转功能，可以使用下面的接口 NdSetAutoRotation 关闭自动旋转功能。

```
[[NdComPlatform defaultPlatform] NdSetAutoRotation:NO];
```

设定平台是否自动旋转。需要在进入平台之前设置。

isAutoRotate 为 NO 关闭自动旋转，使用 NdSetScreenOrientation 设置的方向；

isAutoRotate 为 YES 启用自动旋转，忽略 NdSetScreenOrientation 设置的方向，iPad 支持 4 个方向切换自适应旋转；iPhone 不支持横竖屏切换自适应旋转，仅支持横屏自适应旋转或者竖屏自适应旋转。

注意事项：

如果您的游戏仅支持横屏，由于 SDK 中有拍照、查看相册的功能，但 iOS 系统提供的拍照、查看系统相册的控件（UIImagePickerController）只支持竖屏！故，iOS6 以上只支持横屏的应用接入 SDK 时，需要在应用的 UIApplicationDelegate 对象（例如常见为 AppDelegate 类）中加入如下方法：

```
- (NSUInteger)application:(UIApplication *)application
supportedInterfaceOrientationsForWindow:(UIWindow *)window
{
    return UIInterfaceOrientationMaskAll;
}
```

8. 登录/注销

检查更新完成后，您就可以开始调用登录接口了。
平台提供了两个登录接口，开发者应根据自身需求在其中选择一种进行接入，我们不建议在应用中同时使用两种类型的登录模式：

- 普通登录
  由用户提供账户和密码进行登录和注册，包括第三方类型的登录，这种类型的账户支持平台提供的各类功能，包括好友，支付，充值，成就和排行榜。
- 登录（支持游客登录）
  为没有历史登录账号的设备提供一种游客模式的快速登录，平台为玩家预分配一个 uin 进行，开发者通过这个 uin 进行玩家游戏数据的保持。同时开发者需要及时提醒玩家进行账户转正，成为正式 91 用户。
  处于游客登录状态的玩家无法进行包括好友，支付，充值，成就和排行榜等相关操作。

  使用支持游客登录的接口需要处理用户账户的判断用户登录状态和游客账户转正式账户。

1) 普通登录

如果用户是第一次登录，则系统将进到登录界面，如果用户已经登录过则系统将根据前一次的设置信息判断是否自动登录，如果是自动登录则系统将进行自动登录，如果不是自动登录则系统将进入登录界面，用户可以选择输入已有的用户名和密码进行登录，也可以进入注册界面重新注册新账号，登录及注册界面的具体功能和操作可以进入登录和注册界面查看。

```
[[NdComPlatform defaultPlatform] NdLogin:0];
```

开发者需要处理登录结果通知。

2) 登录（支持游客登录）

如果您想让游戏新手玩家，免去注册或登录时输入账号和密码操作，而快速体验游戏，您可以使用游客登录。游客登录只是系统为游客玩家预分配了一个 UIN，开发者可以通过以这个 UIN 为标识来保存玩家的游戏数据，等到玩家需要购买道具、升到一定级别或者退出游戏时，再提醒玩家进行转正注册，成为 91 用户，可查看使用场景案例。

使用游客登录功能需要配套组合使用登录（支持游客登录）， 判断用户登录状态， 游客账户转正式账户等接口，具体如下接口介绍：

```
[[NdComPlatform defaultPlatform] NdLoginEx:0];
```

本函数在普通登录的基础上增加了游客登录的支持。所谓游客登录就是不需要玩家输入账号和密码注册或登录，系统会快速为未登录过 91SDK 的设备分配一个 UIN。开发者开通过这个 UIN 保存玩家的相关游戏数据，等玩家需要注册时再通过引导游客注册设置账号和密码将分配的 UIN 转正。

如果玩家设备已经有登录过 91 账号，则本函数的调用和普通登录的功能和行为表现是一样的。

开发者需要处理登录结果通知。

3) 游客账户转正式账户

本函数是用于游客登录 Uin 转正（游客注册）。调用后会进入引导游客 Uin 转正界面。
在游客注册界面，游客输入用户名和密码后点注册 91 通行证注册成功后，这时系统的 Uin 与您调用本函数前的 Uin 一致，开发者可以视为该用户正式登录成功；但如果游客选择使用已有 91 通行证登录并登录成后，此时游客登录的 Uin 会变成已有 91 通行证的 Uin。这种情况下，由于 Uin 发生变化，开发者可以视为用户切换帐号，业务层需要使用新的 Uin 进行初始化。
本函数调用的前提是：当前处于游客登录状态，即调用完 NdLoginEx 后发现当前的登录状态是游客登录状态（可通过获取当前登录状态来判断）。

```
[[NdComPlatform defaultPlatform] NdGuestRegist:0];
```

一旦游客账户转成正式账户后，游客登录接口的功能和行为表现与普通登录一致。
调用该接口后用户可能进行转正操作和已有账户进行登录，所以开发者需要处理登录结果通知。同时需要处理登录结果消息中 NdGuestAccountStatus 的状态，以判断是否游客成功注册为普通账号

4) 判断用户登录状态

判断用户是否登录

```
[[NdComPlatform defaultPlatform] isLogined];
```

如果使用的是支持游客登录版本的 SDK，除了判断是否登录的用户，还可以调用

```
[[NdComPlatform defaultPlatform] getCurrentLoginState]
```

来判断当前用户的登录类型。适时的引导用户进行游客账户转正式账户。

5) 切换账户

A）注销当前账号并切到登录界面：

```
[[NdComPlatform defaultPlatform] NdSwitchAccount]
```

该切换账户的接口行为是先调用注销接口，清除自动登录，同时调用登录接口。

B)在已经登录的状态，进入账号管理列表，可以快速选择其它账号登录：

```
[[NdComPlatform defaultPlatform] NdEnterAccountManage];
```

该接口进入账号管理界面，用户可以进行切换账号，如果切换失败，当前账号仍然有效。
备注：该接口可以运用游戏中的小号快速切换。

注意：在“91 社区->首页”和“91 社区->更多”界面里 用户可以进行注销当前登录的账号。这时候 SDK 会抛登录结果通知，告知用户处于未登录状态。注销后，用户可能会登录其它账号，所以开发者需要进行相应的逻辑处理。

6) 登录结果通知

用户在进行普通登录，登录（支持游客登录），游客账户转正式账户和切换账户时，都会发送登录结果通知，开发者都需要在这个通知中进行相应的处理。

所以开发者需要在调用类中设置监听消息。注意只有收到消息才能够判断登录是否成功。否则不应该进行登录的相关操作。

```
//注册监听消息
      [[NSNotificationCenter defaultCenter] addObserver:self
                        selector:@selector(SNSLoginResult:)
                                name:(NSString *)kNdCPLoginNotification
                              object:nil];
```

处理函数：

```
- (void)SNSLoginResult:(NSNotification *)notify
{
      NSDictionary *dict = [notify userInfo];
      BOOL success = [[dict objectForKey:@"result"] boolValue];
      NdGuestAccountStatus* guestStatus = (NdGuestAccountStatus*)[dict
objectForKey:@"NdGuestAccountStatus"];
      //登录成功后处理
      if([[NdComPlatform defaultPlatform] isLogined] && success) {
         //也可以通过[[NdComPlatform defaultPlatform] getCurrentLoginState]判断是否游客登录
状态
         if (guestStatus) {
              if ([guestStatus isGuestLogined]) {
                  //游客账号登录成功;
              }
              else if ([guestStatus isGuestRegistered]) {
```

```
                    //游客成功注册为普通账号
                }
            }
            else  {
                    //普通账号登录成功!
            }
        }
        //登录失败处理和相应提示
        else {
            int error = [[dict objectForKey:@"error"] intValue];
            NSString* strTip = [NSString stringWithFormat:@"登录失败, error=%d", error];
            switch (error) {
                case ND_COM_PLATFORM_ERROR_USER_CANCEL://用户取消登录
                    if ((([[NdComPlatform defaultPlatform] getCurrentLoginState] ==
ND_LOGIN_STATE_GUEST_LOGIN)) {
                        strTip =  @"当前仍处于游客登录状态";
                    }
                    else {
                        strTip = @"用户未登录";
                    }
                    break;
                case ND_COM_PLATFORM_ERROR_APP_KEY_INVALID://appId未授权接入, 或appKey 无效
                    strTip = @"登录失败, 请检查appId/appKey";
                    break;
                case ND_COM_PLATFORM_ERROR_CLIENT_APP_ID_INVALID://无效的应用ID
                    strTip = @"登录失败, 无效的应用ID";
                    break;
                case ND_COM_PLATFORM_ERROR_HAS_ASSOCIATE_91:
                    strTip = @"有关联的91账号，不能以游客方式登录";
                    break;
                default:
                    //其他类型的错误提示
                    break;
            }
        }
}
```

注意：

如果用户在登录界面点击返回，取消了登录，那么将收到登录失败的消息，其中的 error 值为 ND_COM_PLATFORM_ERROR_USER_CANCEL。如果你不关心这种失败的话，请对这种错误进行忽略。

如果是游客登录和游客转正需要关注其中的 NdGuestAccountStatus 字段。如果用户账号登录成功了，可以获取当前用户的唯一标识、会话信息、用户昵称、详细信息等。如下 API：

```
[[NdComPlatform defaultPlatform] loginUin];
```

```
[[NdComPlatform defaultPlatform] sessionId];
[[NdComPlatform defaultPlatform] nickName];
[[NdComPlatform defaultPlatform] NdGetMyInfo];
[[NdComPlatform defaultPlatform] NdGetMyInfoDetail: …];
```

7) 注销

```
[[NdComPlatform defaultPlatform] NdLogout:1];
```

注销登录，传入 0 表示注销； 传入 1 表示注销并清除自动登录。

9. 用户反馈

进入用户反馈的界面，返回错误码。

用户反馈的内容，将在开发者后台呈现. 开发者可以通过开发者后台,对用户的反馈进行回复. 用户会在平台的系统通知里,收到你回复的内容. 因此建议加入该模块。可以通过类似以下方式进行使用：

```
[[NdComPlatform defaultPlatform] NdUserFeedback];
```

10. 显示暂停页

用户需要在游戏暂停或者游戏从后台恢复的时候显示暂停页。

代码示例：

```
- (void)applicationWillEnterForeground:(UIApplication *)application
{
    [[NdComPlatform defaultPlatform] NdPause];
}
```

如果需要监听暂停页关闭的消息，可以监听可以参照下面的代码示例：

```
//在合适的地方监听暂停页退出的消息
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(SNSPauseExit:) name:(NSString
*)kNdCPPauseDidExitNotification object:nil];

//在监听函数中实现你所需要的逻辑
- (void)SNSPauseExit:(NSNotification *)notify
{
    //do what you want
}
```

11. 进入平台中心

进入平台中心的首页界面。参数为标识位，目前保留，默认为 0。

可以通过类似以下方式进行使用：

```
[[NdComPlatform defaultPlatform] NdEnterPlatform:0];
```

12. 应用内购买

平台提供多种应用内购买模式：同步购买(-NdUniPay:)，异步购买(-NdUniPayAsyn:)，代币充值(-NdUniPayForCoin:)。您可以根据自己的需求选择使用其中的一种方式。相关应用内购买的信息请参见 SDK 包中的【91 移动开发平台支付功能接入规范 V2.1.pdf】

1) 如何使用同步购买

i. 要使用的头文件

要使用同步购买功能，首先请包含以下的头文件：

```
#import <NdComPlatform/NdComPlatform.h>
#import <NdComPlatform/NdCPNotifications.h>
#import <NdComPlatform/NdComPlatformError.h>
```

ii. 添加购买结果的监听

为了监听购买的结果，您需要在您的初始化函数（例如-viewDidLoad 或者-init）中添加以下代码：

```
 [[NSNotificationCenter defaultCenter] addObserver:self
                                          selector:@selector(NdUniPayResult:)
                                              name:kNdCPBuyResultNotification
                                            object:nil];
```

iii. 发起购买

发起购买，你需要构建一个 NdBuyInfo 类，然后调用-NdUniPay:购买接口。

NdBuyInfo 类的详细说明请参照<NdComPlatform/NdComPlatformAPIResponse.h>文件

```
- (void)buy
{
    NdBuyInfo *buyInfo = [[NdBuyInfo new] autorelease];
    //订单号必须唯一，推荐为GUID或UUID
    buyInfo.cooOrderSerial = @"120F6CD4018C4D9A8E852AF7D2F32666";
```

```
    buyInfo.productId = @"1";  //自定义的产品ID
    buyInfo.productName = @"苹果";  //产品名称
    buyInfo.productPrice = 0.01; //产品现价，价格大等于0.01,支付价格以此为准
    buyInfo.productOrignalPrice = 0.01; //产品原价，同现价保持一致
    buyInfo.productCount = 1;  //产品数量
    buyInfo.payDescription = @"gamezoon1";   //服务器分区，不超过20个字符，只允许英文或数字

    //发起请求并检查返回值。注意！该返回值并不是交易结果！
    int res = [[NdComPlatform defaultPlatform] NdUniPay:buyInfo];
    if (res < 0)
    {
        //输入参数有错！无法提交购买请求
    }
}
```

iv. 实现购买结果监听的函数

购买完成后，您的监听函数会被回调。在你的监听函数中，可以获取到本次购买的订单序列号，购买结果以及错误信息。示例代码如下：

```
- (void)NdUniPayResult:(NSNotification*)notify
{
    NSDictionary *dic = [notify userInfo];
    BOOL bSuccess = [[dic objectForKey:@"result"] boolValue];
    NSString* str = bSuccess ? @"购买成功" : @"购买失败";

    if (!bSuccess) {
        //TODO: 购买失败处理
        NSString* strError = nil;
        int nErrorCode = [[dic objectForKey:@"error"] intValue];
        switch (nErrorCode) {
            case ND_COM_PLATFORM_ERROR_USER_CANCEL:
                strError = @"用户取消操作";
                break;
            case ND_COM_PLATFORM_ERROR_NETWORK_FAIL:
                strError = @"网络连接错误";
                break;
            case ND_COM_PLATFORM_ERROR_SERVER_RETURN_ERROR:
                strError = @"服务端处理失败";
                break;
            default:
                strError = @"购买过程发生错误";
                break;
        }
```

```
        str = [str stringByAppendingFormat:@"\n%@", strError];
    }
    else {
        //TODO: 购买成功处理
    }

    //本次购买的请求参数
    NdBuyInfo* buyInfo = (NdBuyInfo*)[dic objectForKey:@"buyInfo"];
    str = [str stringByAppendingFormat:@"\n<productId = %@, productCount = %d, cooOrderSerial
= %@>",
            buyInfo.productId, buyInfo.productCount, buyInfo.cooOrderSerial];
    NSLog(@"NdUiPayResult: %@", str);
}
```

v. 移除监听

在退出你的界面之前，记得移除你的监听。在你的析构函数中（例如-dealloc）移除监听，示例代码如下：

```
[[NSNotificationCenter defaultCenter] removeObserver:self
                  name:kNdCPBuyResultNotification    object:nil];
```

vi. 漏单处理

在购买过程中，可能出现以下情况：购买已经完成，而应用程序却没有收到购买结果的通知，例如，购买过程中，用户退出了应用程序，或者网络出现了问题，导致购买成功的消息无法到达客户端。对于这种情况，建议采取以下措施。

在发起购买请求后，立即记录下该请求的订单号，在购买结果到达时，删除该条记录。而在每次应用程序启动时，检查是否有未收到购买结果的订单，如果有，向服务器发起验证，并根据验证结果处理该记录。

示例如下：

A）记录购买请求

```
- (void)buy
{
    //做些必要的工作
    dosomething();
    int res = [[NdComPlatform defaultPlatform] NdUniPay:buyInfo];
    if (res < 0)
    {
        //输入参数有错！无法提交购买请求
    }
```

```
    else
    {
        //记录下该笔交易
        [self addRecord:buyInfo.cooOrderSerial];
    }
}
```

B）在收到购买结果时，移除该项记录

```
- (void) NdUniPayResult:(NSNotification *)notify
{
    //这里进行上面介绍过的处理
    //然后移除本地的该笔交易记录
    [self removeRecord:cooOrderSerial];
}
```

C）程序启动时检查未核对结果的交易

在合适的地方添加类似以下的检查代码

```
NSArray *recordArray = [self getUnCheckedRecord];
for (NSString *cooOrderSerial in recordArray) {
    [[NdComPlatform defaultPlatform]  NdCheckPaySuccess : cooOrderSerial  delegate : self];
}
```

D）处理收到的核对信息

使用-NdCheckPaySuccess:delegate:来检查订单的支付状态，需要实现对应的回调函数-checkPaySuccessDidFinish:cooOrderSerial:bSuccess:。

```
- (void)checkPaySuccessDidFinish:(int)error
          cooOrderSerial:(NSString*)cooOrderSerial
          bSuccess:(BOOL)bSuccess
{
    if (bSuccess)
    {
        //说明该笔订单购买成功
        [self dealWithBuySuccess:cooOrderSerial];
    }
    else
    {
        //说明该笔订单购买失败
        [self dealWithBuyFailure:cooOrderSerial];
    }
    [self removeRecord:cooOrderSerial];
}
```

API 详细介绍请参考-NdCheckPaySuccess:delegate:

注意事项：该 API 要求用户登录，未登录情况下无法使用该 API。

2) 如何使用异步购买

使用异步购买，你所做的工作与同步购买基本一样。下面，我们来看一下异步购买的处理。

i. 要使用的头文件

要使用异步购买功能，首先请包含以下的头文件：

```
#import <NdComPlatform/NdComPlatform.h>
#import <NdComPlatform/NdCPNotifications.h>
#import <NdComPlatform/NdComPlatformError.h>
```

ii. 添加购买结果的监听

为了监听购买的结果，您需要在您的初始化函数（例如-viewDidLoad 或者-init）中添加以下代码：

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(NdUniPayAysnResult:)
                                             name:kNdCPBuyResultNotification
                                           object:nil];
```

iii. 发起购买

发起购买，你需要构建一个 NdBuyInfo 类，然后调用-NdUniPayAsyn:购买接口。
NdBuyInfo 类的详细说明请参照<NdComPlatform/NdComPlatformAPIResponse.h>文件

```
- (void)buyAysn
{
    NdBuyInfo *buyInfo = [[NdBuyInfo new] autorelease];
    //订单号必须唯一，推荐为GUID或UUID
    buyInfo.cooOrderSerial = @"12OF6CD4018C4D9A8E852AF7D2F32666";
    buyInfo.productId = @"1";  //自定义的产品ID
    buyInfo.productName = @"苹果";  //产品名称
    buyInfo.productPrice = 0.01; //产品现价，价格大等于0.01,支付价格以此为准
    buyInfo.productOrignalPrice = 0.01; //产品原价，价格同现价保持一致
    buyInfo.productCount = 1;  //产品数量
    buyInfo.payDescription = @" gamezoon1" ;   //服务器分区，不超过20个字符，只允许英文或数字

    //发起请求并检查返回值。注意！该返回值并不是交易结果！
    int res = [[NdComPlatform defaultPlatform]  NdUniPayAsyn: buyInfo];
    if (res < 0)
    {
```

```
        //输入参数有错！无法提交购买请求
    }
}
```

iv. 实现购买结果监听的函数

购买完成后，您的监听函数会被回调。在你的监听函数中，可以获取到本次购买的订单序列号，购买结果以及错误信息。

异步购买除了支持余额充足的购买以外，还支持另外一种模式：在余额不足以购买某物品时，先进行充值，并在充值金额到账后，由服务器自动为您购买该物品。

在余额充足时，异步购买的行为与同步购买一致，在购买结束后发送购买结束消息，并告知购买结果。客户端可以在收到该消息，并确认购买成功时，从服务器更新购买物品的消息。

在余额不足时，在结束异步购买行为时，会发送购买结束的消息，需要注意的是，该消息的购买状态为“购买失败”，并且失败的错误码为“请求已提交”！所以，如果收到购买失败的消息，并且错误码为“请求已提交”时，客户端可以向服务器请求更新购买物品消息。但是，必须注意的是：这时，物品很有可能还未到账，无法获取到该物品消息，甚至有可能用户在充值购买的确认页面取消了请求。

示例代码如下：

```
- (void)NdUniPayAsynResult:(NSNotification*)notify
{
    NSDictionary* dic = [notify userInfo];
    BOOL bSuccess = [[dic objectForKey:@"result"] boolValue];
    NSString* str = bSuccess ? @"购买成功" : @"购买失败";

    if (!bSuccess) {
        //TODO: 购买失败处理
        NSString* strError = nil;
        int nErrorCode = [[dic objectForKey:@"error"] intValue];
        switch (nErrorCode) {
            case ND_COM_PLATFORM_ERROR_USER_CANCEL:
                strError = @"用户取消操作";
                break;
            case ND_COM_PLATFORM_ERROR_NETWORK_FAIL:
                strError = @"网络连接错误";
                break;
            case ND_COM_PLATFORM_ERROR_SERVER_RETURN_ERROR:
                strError = @"服务端处理失败";
                break;
            case ND_COM_PLATFORM_ERROR_ORDER_SERIAL_SUBMITTED:
                //!!!: 异步支付，用户进入充值界面了
                strError = @"支付订单已提交";
                break;
```

```
            default:
                strError = @"购买过程发生错误";
                break;
        }
        str = [str stringByAppendingFormat:@"\n%@", strError];
    }
    else {
        //TODO: 购买成功处理
    }

    //本次购买的请求参数
    NdBuyInfo* buyInfo = (NdBuyInfo*)[dic objectForKey:@"buyInfo"];
    str = [str stringByAppendingFormat:@"\n<productId = %@, productCount = %d, cooOrderSerial
= %@>",
            buyInfo.productId, buyInfo.productCount, buyInfo.cooOrderSerial];

    NSLog(@"NdUiPayAsynResult: %@", str);
}
```

v. 移除监听

在退出你的界面之前，记得移除你的监听。在你的析构函数中（例如-dealloc）移除监听，示例代码如下：

```
[[NSNotificationCenter defaultCenter] removeObserver:self
                                name:kNdCPBuyResultNotification object:nil];
```

最后，使用异步购买请注意以下几点：

1）再次强调，异步购买要求应用有自己的业务服务器，同时虚拟物品必须通过业务服务器获取。

2）应用程序客户端只能从服务器获取用户所拥有的虚拟物品的信息，不能本地缓存。例如：用户通过异步购买，买了一个虚拟物品“战神斧”。购买结束后，用户打开“背包”查看自己的物品信息，背包里的物品信息必须是从服务器获取的。客户端不能在购买行为结束后，为用户加入“战神斧”物品，必须从服务端获取！只有服务端确认并发回用户拥有了战神斧物品的信息，应用程序才能认为用户确实获得了该物品。

3）如果客户端想在发起购买请求后刷新物品消息，可以在收到购买消息时，判断如下两种状态时进行刷新：

a）购买成功

b）购买失败，并且错误码为：

请求已提交（ND_COM_PLATFORM_ERROR_ORDER_SERIAL_SUBMITTED）

但需要再次强调的是，第二种状态时，由于充值到账存在延时，服务端可能不能及时得到该物品的状态信息。

3) 代币充值

代币充值是 SDK 为拥有自己业务服务器的开发者提供的一种便利快捷的充值支付渠道，可以和 SDK 内部的 91 豆充值一样直接对应用内代币进行充值操作，无须在后台配置虚拟物品再进行购买。开发者使用时需要在后台配置代币的名称，单位，以及同人民币的汇率，同时开发者还需要配置支付结果通知地址。

同时代币充值中的 91 豆充值功能支持测试模式，相关配置和注意事项具体参见设定调试模式中的相关内容。

SDK 提供的接口如下,其中

- cooOrderSerial 是合作商订单号，必须保证唯一，这是双方对账的唯一标记，应该使用 GUID 生成的 32 位的字符串；
- needPayCoins 是玩家代币余额不足的情况下，还需要充值的应用代币的数额。开发者如果不关注该参数可以传零；
- payDescription 可记录服务器分区（不要超过 20 个字符，只允许英文或数字），默认为空串，在支付结果通知时返回给开发者。

```
-(int)NdUniPayForCoin:(NSString*)cooOrderSerial  needPayCoins:(int)needPayCoins payDescription:(NSString*)payDescription;
```

使用场景示例：

《星际迷航 Demo》中使用“神马币”作为游戏内流通的应用代币，其与人民币的的兑换汇率为 1:100，后台配置完成后。在游戏中，玩家 A 的账上还有 30 个神马币的余额，玩家 A 在购买一个单价为 50 个神马币的“战斗机”时，还需支付 20 个神马币，此时玩家使用 API 进行充值时 needPayCoins 应为 20。API 调用示例代码如下所示：

```
CFUUIDRef theUUID = CFUUIDCreate(NULL);
CFStringRef guid = CFUUIDCreateString(NULL, theUUID);
CFRelease(theUUID);
NSString *uuidString = [((NSString *)guid) stringByReplacingOccurrencesOfString:@"-"
withString:@""];
CFRelease(guid);
 [[NdComPlatform defaultPlatform] NdUniPayForCoin:[uuidString lowercaseString] needPayCoins:20
payDescription: @" "];
```

调用接口后会进入到 SDK 的代币充值界面，充值结果 SDK 会按后台配置的地址通知开发者的业务服务器，开发者应该及时处理并更新数据，充值成功应该为用户发放相应的代币。客户端在离开代币充值界面会发送退出平台的消息，开发者可以通过捕获退出平台界面的通知来刷新数据，但是由于网络等原因，充值不一定会到账。

使用代币充值需要注意的：

- 开发者应该拥有自己的业务服务器，在后台准确配置消息通知地址，同时实现接收充值结果通知的逻辑。
- 由于网络等原因服务器可能无法及时通知到业务服务器充值结果，开发者可以适当延时再进行查询获取。
- 捕获到退出平台的消息只表示客户端退出充值界面，具体是否充值，充值是否成功未知，客户端需要和业务服务器进行进一步确认。

4) 指定服务器充值

开发者的游戏中，如果存在多个服务器，且在充值时，需要区分充值到哪个服务器，开发者可以按照一下方法进行处理。
每种充值模式的 API 传入参数中，都有一个 payDescription，这个字符串字段（不要超过 20 个字符）。开发者可以使用这个字段来区分充值到哪个服务器。例如 payDescription 传入 1003，表示游戏服务器 ID。

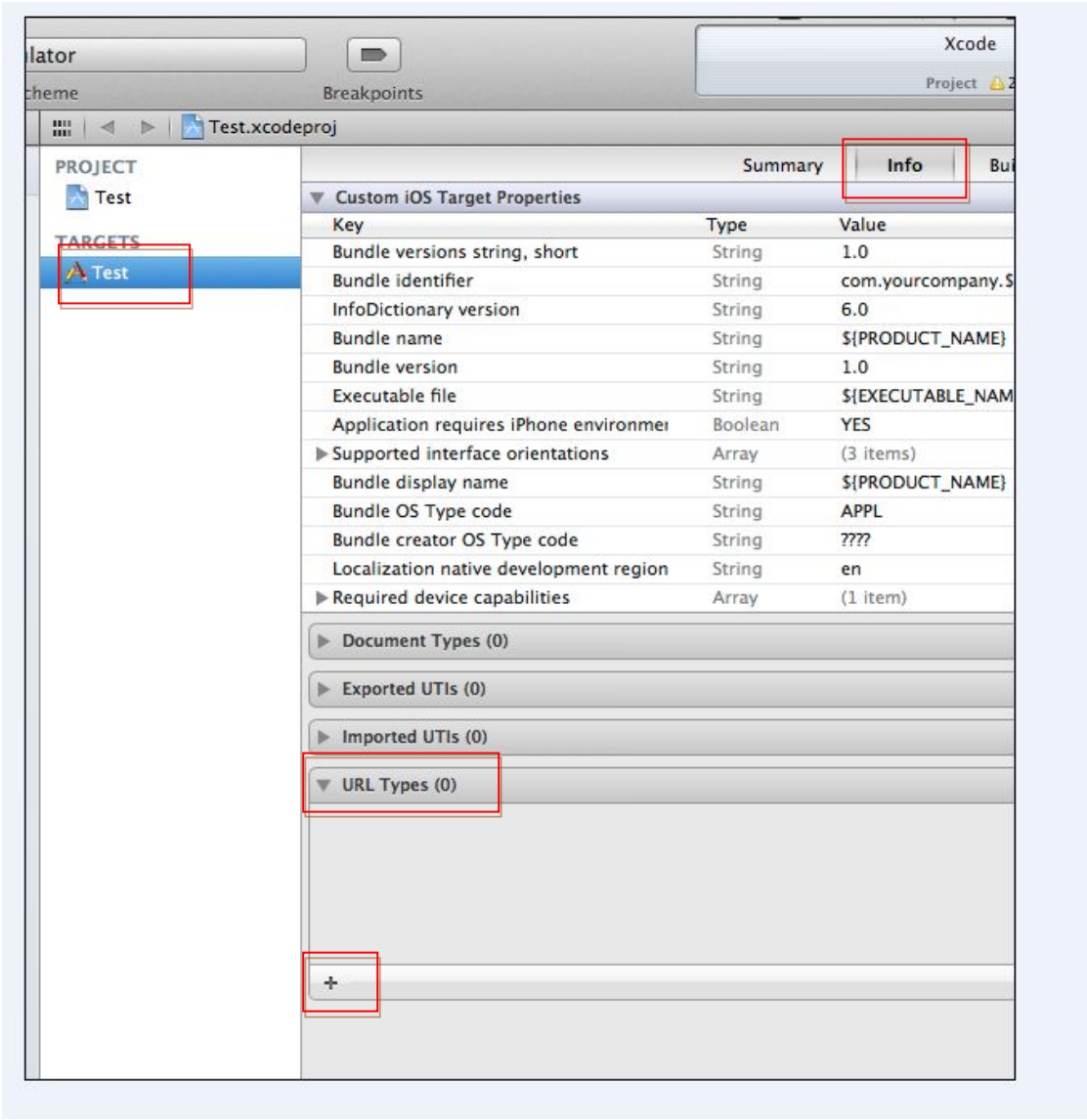
在游戏服务器与 91 服务器的通信接口中，例如:【接收支付购买结果】,【查询支付购买结果】，这两个接口中的 Note，就是客户端购买时，提交 payDescription,字段。游戏服务器通过处理 Note 字段，即可区分那个游戏服务器。详情查看【91 移动开发平台服务端与应用服务端接口规范 1.00.pdf】

13. URL Scheme 设置

从 91SDK3.2.5 开始要求接入方需要设置一个 URL Scheme，URL Scheme 格式为：91-xxxxx，其中 xxxx 为你的软件标识符。

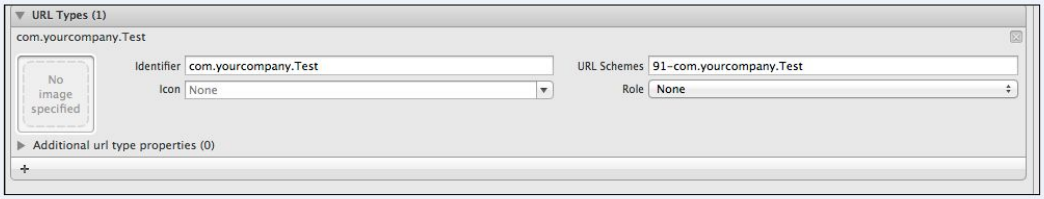
设置方法如下：

选中工程中的 Target，选中 Info 标签页，找到底下的 URL Types，展开，点击加号，创建一个新的 URL Scheme。



点击后，Identifier 字段填入你的软件标识符，URL Schemes 字段填入格式为：91-xxxxx，其中 xxxx 为你的软件标识符。Role 字段可以设置为 None，Icon 字段可以不填。

示例如下：



五、 91SDK 扩展功能

该部分为 91SDK 的扩展功能，开发者可以根据自身软件的特性，来参考使用这些扩展功能。

14. 分享到第三方平台

进入分享到第三方平台的界面，如果没有绑定指定的平台，则会跳到绑定界面。

strContent：预分享的内容（140个字符），第三方平台可能会对重复内容进行屏蔽处理（如新浪微博禁止发重复内容）

imageInfo：分享图片到新浪微博，预分享的图片，可以是当前屏幕截图，指定的UIImage 或者 图片名称（支持全路径，或者只有文件名。如果为文件名，load from main bundle）分享图片到新浪微博。具体NdImageInfo详见NdComPlatformAPIResponse.h定义。

NdImageInfo支持的功能：

```
+ (id)imageInfoWithScreenShot;                    /**< 使用当前屏幕的图像 */
+ (id)imageInfoWithImage:(UIImage*)image;         /**< 使用指定的image */
+ (id)imageInfoWithFile:(NSString*)file;              /**< 使用指定的图片文件 */
```

示例代码：（分享当前应用截屏）

```
NSDate* date = [NSDate date];
NSString* str = [NSString stringWithFormat:@"我回来了！ --%@", [date description]];
[[NdComPlatform defaultPlatform] NdShareToThirdPlatform: str
                                                imageInfo : [NdImageInfo imageInfoWithScreenShot] ] ;
```

15. 捕获会话过期的通知

当登录的用户会话标示无效时（此时用户信息需要重新登录才能再次获取），将发送 **kNdCPSessionInvalidNotification**（ 在 NdCPNotifications.h 文件中声明）。用户可以通过捕获该消息来进行相关的处理和引导用户重新登录。

```
    [[NSNotificationCenter defaultCenter] addObserver:self
```

```
                                selector:@selector(SNSSessionInvalid:)
                                 name:(NSString *)kNdCPSessionInvalidNotification
                                 object:nil];
```

16. 进入平台中心各模块

1）进入好友中心

进入好友中心界面，参数为标识位，目前保留，默认为 0。

可以通过类似以下方式进行使用：

```
[[NdComPlatform defaultPlatform] NdEnterFriendCenter:0];
```

2）进入指定用户的空间

该函数用来进入指定用户的空间，可以查看用户的详情信息，排行榜游戏信息。

如果 uin 是好友，则可以进行好友备注修改、发送消息、查看他的新鲜事、他的好友、删除好友操作。

如果 uin 不是好友，则可以进行添加好友操作。

如果 uin 为空，返回参数错误。

可以通过类似以下方式进行使用：

```
[[NdComPlatform defaultPlatform] NdEnterUserSpace:strUin];
```

3）进入游戏大厅

进入游戏大厅，nFlag 标识位，保留，默认为 0。

可以通过类似以下方式进行使用：

```
[[NdComPlatform defaultPlatform] NdEnterAppCenter:0];
```

4）进入指定应用的主页

进入指定应用的主页,nFlag 标识位,保留,默认为 0,appId 是指定应用或游戏的 appid，传入的 appid 小于等于 0 时直接进入游戏大厅。

可以通过类似以下方式进行使用：

```
int appId =  [NdComPlatform defaultPlatform] appId];
[[NdComPlatform defaultPlatform] NdEnterAppCenter:0 appId:appId];
```

5）进入设置界面

进入设置界面，nFlag 标识位，保留，默认为 0。

可以通过类似以下方式进行使用：

```
[[NdComPlatform defaultPlatform] NdEnterUserSetting:0];
```

6）进入邀请好友界面

调用该函数将进入平台的批量邀请好友的界面，strInviteContent 是邀请的内容，可为空。可以通过类似以下方式进行使用：

```
[[NdComPlatform defaultPlatform] NdInviteFriend:@"91 社区新版升级啦，赶快来啦！"];
```

7）进入应用论坛界面

进入应用论坛，返回错误码。nFlag， 保留，默认为 0。可以通过类似以下方式进行使用：

```
[[NdComPlatform defaultPlatform] NdEnterAppBBS:0];
```

17. 虚拟商店

1）简介

首先需要在开发者后台配置相关的虚拟商品信息。

虚拟商店可以配置使用 91 豆购买虚拟商品或者是自定义游戏币购买虚拟商品。

商品计费类型：非消费型商品，订阅型商品，消费型商品，总共 3 种。

商品类别：可以对商品进行归类，指定类别名称，避免商品过多引起的杂乱无章。比如农场游戏，指定“水果”、“蔬菜”等类别的虚拟商品。该属性不是必需的。

促销信息：后台还可以进行促销信息的定制。

注：需要登录后才能调用该模块接口。为了减少冗余的示例代码，下面的示例代码假设用户已登录了平台。

关于虚拟商品的展示，购买与使用流程等详细信息，可以参照【91 移动开发平台支付功能接入规范 V2.1.pdf】

2）进入虚拟商店

平台有提供虚拟商品展示界面，进去后用户可以查看商品的详细信息，可以进行购买相关操作，对所有商品一目了然。

开发者可以对商店里展示的虚拟商品进行商品类别、商品计费类型的过滤。用户如果有

发生购买行为，该次的购买流程结束后就会抛消息给开发者。开发者可能会收到多个的购买结果信息。当用户离开虚拟商店，退回应用的时候，会抛离开91平台的消息。开发者可以依据需要进行相应的处理。

附：如果要指定展示某一类的虚拟商品，需要知道类别ID，可以通过获取虚拟商品类别信息接口获得。

需要额外包含的头文件：

```
#import <NdComPlatform/NdComPlatform+VirtualGoods.h>
#import <NdComPlatform/NdComPlatformAPIResponse+VirtualGoods.h>
```

使用流程示例代码

i. 添加虚拟商店消息监听

```
//监听虚拟商店的相关消息。
//1.每一次的购买结果
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(NdBuyCommodityResult:)
                                name:(NSString *)kNdCPBuyResultNotification object:nil];
//2.退出虚拟商店
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(SNSLeaveComplatformUI:)
                                name:(NSString *)kNdCPLeavePlatformNotification object:nil];
```

ii. 进入虚拟商店

```
//进入虚拟商店，显示所有类别，所有计费类型的商品
int nFeeType = ND_VG_FEE_TYPE_POSSESS  | ND_VG_FEE_TYPE_SUBSCRIBE  | ND_VG_FEE_TYPE_CONSUME;
[[NdComPlatform defaultPlatform] NdEnterVirtualShop:nil  feeType:nFeeType];
```

iii. 实现监听函数

```
- (void)NdBuyCommodityResult:(NSNotification*)notify
{
        //TODO: 分析这次的购买结果，参照购买虚拟商品结果处理
}

- (void)SNSLeaveComplatformUI:(NSNotification*)notify
{
        // TODO: 退出虚拟商店平台
}
```

iv. 移除监听

```
[[NSNotificationCenter defaultCenter] removeObserver : self ];
```

3）获取虚拟商品类别

获取后台配置的所有虚拟商品类别信息。开发者可以用来自定义类别展示界面。

示例代码：

```
//获取虚拟商品类别信息
    [[NdComPlatform defaultPlatform] NdGetCategoryList: target];

//target 需要实现的回调方法
- (void)getCategoryListDidFinish:(int)error records:(NSArray *)records
{
    if (error < 0) {
        //TODO: 下载类别信息失败
    }
    else {
        //TODO: 类别信息处理
        for (NdVGCategory* category in records) {
            //...
        }
    }
}
```

4）获取应用促销信息

如果后台有配置促销信息，开发者想自己展示促销信息内容，可以使用该方法。
注：虚拟商店界面有展示促销信息。如果没有促销信息，文本为空。

示例代码：

```
//获取应用促销信息
       [[NdComPlatform defaultPlatform] NdGetAppPromotion: target];

//target 需要实现的回调方法
- (void)getAppPromotionDidFinish:(int)error  promotion:(NSString*)promotion
{
    if (error < 0) {
        //TODO: 下载促销消息失败
    }
```

```
    else {
        //TODO: 取得促销消息内容 promotion。如果文本为空，表示没有进行促销。
    }
}
```

5）获取商店里的商品信息列表

获取虚拟商品信息列表，开发者可以自定义展示界面，可以获取指定商品类别，商品计费类型。

示例代码：

```
//分页获取虚拟商品信息，不指定类别，计费类型
NdPagination* pagination = [[NdPagination new] autorelease];
pagination.pageSize = 10;   //5 的倍数，不要超过 50
pagination.pageIndex = …;  //获取页的索引值，从 1 开始，总共的页数可以通过返回接口里 result.totalCount 字段计算出来，第一次请求用 1
 [[NdComPlatform  defaultPlatform]  NdGetCommodityList:nil  feeType:7  pagination:pagination packageId:nil  delegate: target];

//target 需要实现的回调方法
- (void)getCommodityListDidFinish:(int)error  cateId:(NSString*)cateId
              feeType:(int)feeType              packageId:(NSString               *)packageId result:(NdBasePageList*)result
{
    if (error < 0) {
        //TODO: 下载商品信息列表失败
    }
    else {
        //TODO: 返回某一页的信息列表
        //result.pagination;  当前返回的页索引信息
        //计算出总的页数，可以用来处理下一页的请求。如果没有商品时，nPageCount 为 0，可能需要特殊处理。
        int nPageCount = (result.totalCount + pagination.pageSize - 1) / pagination.pageSize;
        for (NdVGCommodityInfo* info in result.records) {
            //商品信息，...
        }
    }
}
```

6）购买虚拟商品

购买指定的虚拟商品，购买成功或者失败，接口会弹窗提示，开发者无需再做界面提示。购买流程结束会抛消息通知开发者。

a)如果虚拟商品是 91 豆支付模式，会先请求该商品是否可购买，如果可购买则进入异

步支付的购买界面；

b)如果虚拟商品是游戏币支付模式，会直接请求支付。如果余额不足会引导用户进入虚拟币直充界面。

同时虚拟币直充界面的 91 豆充值功能支持测试模式，相关配置和注意事项具体参见设定调试模式中的相关内容。

使用流程示例代码：

i. 添加购买结果的监听

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(NdBuyCommodityResult:)
                                             name:kNdCPBuyResultNotification
                                           object:nil];
```

ii. 发起购买请求

```
//购买虚拟商品
        NdVGOrderRequest* orderRqst = [NdVGOrderRequest orderRequestWithProductId : strProductId
                                       productCount : nCount       payDescription:nil];
        //orderRqst.vgCommodityInfo = vgCommodityInfo;  //(建议)如果已经获取该商品信息，传入该值
        int res = [[NdComPlatform defaultPlatform] NdBuyCommodity:orderRqst];
        if (res < 0) {
          switch (res) {
               case ND_COM_PLATFORM_ERROR_NOT_LOGINED:
                    [DemoComFunc messageBox:@"未登录"];
                    break;
               case ND_COM_PLATFORM_ERROR_PARAM:
                    [DemoComFunc messageBox:@"参数错误"];
                    break;
               default:
               [DemoComFunc messageBox: [NSString stringWithFormat:@"购买失败 error = %d", res]];
                    break;
          }
        }
```

iii. 实现购买结果监听的函数

```
- (void)NdBuyCommodityResult:(NSNotification*)notify
{
    (NSDictionary*)dic = [notify userInfo];
    BOOL bSuccess = [[dic objectForKey:@"result"] boolValue];
    NSString* str = bSuccess ? @"购买成功" : @"购买失败";

    if (!bSuccess) {
        //TODO: 购买失败处理
        NSString* strError = nil;
        int nErrorCode = [[dic objectForKey:@"error"] intValue];
        switch (nErrorCode) {
            // {{购买虚拟商品(91 豆/游戏币)必须处理的错误码
            case ND_COM_PLATFORM_ERROR_USER_CANCEL:
                strError = @"用户取消操作";
                break;
            case ND_COM_PLATFORM_ERROR_NETWORK_FAIL:
                strError = @"网络连接错误";
                break;
            case ND_COM_PLATFORM_ERROR_SERVER_RETURN_ERROR:
                strError = @"服务端处理失败";
                break;
            case ND_COM_PLATFORM_ERROR_VG_MONEY_TYPE_FAILED:
                strError = @"查询虚拟商品币种失败";
                break;
            // }}


            // {{ 1.购买虚拟商品（91 豆）必须处理的错误码
            case ND_COM_PLATFORM_ERROR_ORDER_SERIAL_SUBMITTED:
                //!!!: 异步支付，用户进入充值界面了
                strError = @"支付订单已提交";
                break;
            case ND_COM_PLATFORM_ERROR_PARAM:
                //可能是虚拟商品价格为 0，或者总额超过最大值
                strError = @"购买的虚拟商品 91 豆价格总额不合法";
                break;
            case ND_COM_PLATFORM_ERROR_VG_ORDER_FAILED:
                strError = @"获取虚拟商品订单号失败";
                break;
            // }}
```

```
            // {{ 2 购买虚拟商品（游戏币）必须处理的错误码
            case ND_COM_PLATFORM_ERROR_VG_BACK_FROM_RECHARGE:
                strError = @"有进入虚拟币直充界面";
                break;
            case ND_COM_PLATFORM_ERROR_PAY_FAILED:
                strError = @"购买虚拟商品失败";
                break;
            // }}

            default:
                strError = @"购买过程发生错误";
                break;
        }
        str = [str stringByAppendingFormat:@"\n%@", strError];
    }
    else {
        //TODO: 购买成功处理
    }

    // 如果购买失败，可能无法得到 cooOrderSerial, productPrice 等字段值
    NdBuyInfo* buyInfo = (NdBuyInfo*)[dic objectForKey:@"buyInfo"];
    str = [str stringByAppendingFormat:@"\n<productId = %@, productCount = %d, cooOrderSerial
= %@>",
           buyInfo.productId, buyInfo.productCount, buyInfo.cooOrderSerial];


    // 如果购买虚拟商品失败，可以从 vgErrorInfo 获取具体失败详情，参照 NdVGErrorInfo 定义
    NdVGErrorInfo* vgErrInfo = (NdVGErrorInfo*)[dic objectForKey:@"vgErrorInfo"];
    if (vgErrInfo) {
        str = [str stringByAppendingFormat:@"\n\n%@", vgErrInfo.strErrDesc];
    }

    NSLog(@"NdBuyCommodityResult: %@", str);
}
```

iv. 移除监听

```
[[NSNotificationCenter defaultCenter] removeObserver:self
                                name:kNdCPBuyResultNotification object:nil];
```

7）使用已购买的虚拟商品

使用指定的虚拟商品，该商品应该是已购买并可用的。对于有使用次数/数量限制的商品，在调用该接口后，服务端会对该商品进行可用次数/数量进行相应的消减操作。比如有使用次数限制的订阅型商品，具备数量属性的消费型商品。

如果不知道商品的授权信息，可以根据下面的情况进行查询商品是否可用：

非消费型商品：可以直接调用已购买的非消费型商品查询接口看是否已购买，不需要调用该接口。

订阅型商品（按使用时间订阅）：可以直接调用已购买的订阅型商品查询接口看是否已过期，不需要调用该接口。

订阅型商品（按使用次数订阅）：首先调用获取已购买的商品信息查看授权信息。使用时，调用该接口，让服务端对该商品进行可用次数做消减操作。

消费型商品：首先调用获取已购买的商品信息查看授权信息。使用时，调用该接口，让服务端会对该商品进行可用数量做消减操作。

示例代码：

```
//使用指定的虚拟商品
    NdVGUseRequest* useRqst = [[NdVGUseRequest new] autorelease];
    useRqst.nUseCount = 1;//...
    useRqst.strProductId = …;// TODO: 所要购买的商品 id
    int nRet = [[NdComPlatform defaultPlatform] NdUseHolding:useRqst delegate: target];
    if (nRet < 0) {
        //TODO: 请求失败，检查 useRqst.strProductId 是否有效，或者网络问题
    }
//target 需要实现的回调方法
- (void)useHoldingDidFinish:(int)error  useRequest:(NdVGUseRequest*)useRequest
                useResult:(NdVGUseResult*)useResult
{
    if (error < 0) {
        //TODO: 请求使用接口失败
    }
    else {
        useResult.vgAuthInfo; //该商品当前的授权状态信息
        if (useResult.bCanUse) {
            //TODO: 使用成功
        }
        else {
            //TODO: 不可使用
            //useResult.nErrCode, useResult.strErrDesc 参见 NdVGUseResult 类的说明
        }
```

```
    }
}
```

8）已购买的非消费型商品查询

查询非消费型商品是否已购买。

示例代码：

```
//发送请求，判断非消费型商品是否已购买，productId 必须是非消费型商品 id
int nRet = [[NdComPlatform defaultPlatform] NdProductIsPayed:productId delegate: target];
if (nRet < 0) {
    //TODO: 请求失败，检查 productId 是否有效，或者网络问题，或者未登录
}

//target 需要实现的回调方法
-          (void)NdProductIsPayedDidFinish:(int)error                    isPayed:(BOOL)isPayed
canUseInThisImei:(BOOL)canUse
                        errCode:(int)errCode   errDesc:(NSString*)errDesc
{
    if (error < 0) {
        //TODO: 请求使用接口失败
    }
    else {
        if (isPayed) {
            //TODO: 商品已购买, 再判断 canUse 看是否能在当前机器上使用
        }
        else {
            //TODO: 商品未购买，或者其它错误，
            //errCode/errDesc 参见 NdComPlatformUIProtocol_VirtualGoods 中该接口的说明
        }
    }
}
```

9）已购买的订阅型商品查询

查询订阅型商品（按使用时间订阅）是否可以已过期。

示例代码：

```
//发送请求，判断订阅型商品是否已过期，productId 必须是订阅型商品 id
int nRet = [[NdComPlatform defaultPlatform] NdProductIsExpired:productId delegate: target];
if (nRet < 0) {
    //TODO: 请求失败，检查 productId 是否有效，或者网络问题，或者未登录
}

//target 需要实现的回调方法
```

```
- (void)NdProductIsExpiredDidFinish:(int)error isExpired:(BOOL)isExpired
canUseInThisImei:(BOOL)canUse
                errCode:(int)errCode  errDesc:(NSString*)errDesc
{
    if (error < 0) {
        //TODO: 请求使用接口失败
    }
    else {
        if (!isExpired) {
            //TODO: 商品未过期，再判断 canUse 看是否能在当前机器上使用
        }
        else {
            //TODO: 商品过期，或者其它错误，
            //errCode/errDesc 参见 NdComPlatformUIProtocol_VirtualGoods 中该接口的说明
        }
    }
}
```

10） 获取已购买的指定商品信息

查询指定商品（已购买）的授权信息。

示例代码：

```
//发送请求，判断某商品的授权使用信息，productId 可以是任何类型的商品
int nRet = [[NdComPlatform defaultPlatform] NdGetUserProduct:productId delegate: target];
if (nRet < 0) {
    //TODO: 请求失败，检查 productId 是否有效，或者网络问题，或者未登录
}

//target 需要实现的回调方法
- (void)NdGetUserProductDidFinish:(int)error  canUse:(BOOL)canUse  errCode:(int)errCode
                errDesc:(NSString*)errDesc  authInfo:(NdVGInfoBase*)vgAuthInfo
{
    if (error < 0) {
        //TODO: 请求使用接口失败
    }
    else {
        vgAuthInfo; //该商品当前的授权状态信息
        if (canUse) {
            //TODO: 该商品可以使用
        }
        else {
            //TODO: 该商品不可使用
            //errCode/errDesc 参见 NdComPlatformUIProtocol_VirtualGoods 中该接口的说明
```

```
        }
    }
}
```

11） 获取已购买的商品信息列表

获取用户已经购买过的商品列表。

示例代码：

```
//分页获取已购买的虚拟商品信息
NdPagination* pagination = [[NdPagination new] autorelease];
pagination.pageSize = 10;   //5 的倍数，不要超过 50
pagination.pageIndex = …;  //获取页的索引值，从 1 开始，总共的页数可以通过返回接口里 result.totalCount 字段计算出来，第一次请求用 1
int nRet = [[NdComPlatform defaultPlatform] NdGetUserProductsList:pagination delegate: target];
if (nRet < 0) {
    //TODO: 请求失败，检查 productId 是否有效，或者网络问题，或者未登录
}

//target 需要实现的回调方法
- (void)NdGetUserProductsListDidFinish:(int)error  result:(NdBasePageList*)result
{
    if (error < 0) {
        //TODO: 请求使用接口失败
    }
    else {
        //TODO: 返回某一页的信息列表
        //result.pagination;  当前返回的页索引信息

        //计算出总的页数，可以用来处理下一页的请求。如果没有商品时，nPageCount 为 0，可能需要特殊处理。
        int nPageCount = (result.totalCount + pagination.pageSize - 1) / pagination.pageSize;

        for (NdVGHoldingInfo* info in result.records) {
            //已购买的商品授权信息...
        }
    }
}
```

12） 查询游戏币余额

该接口用来查询虚拟商店中自定义的游戏币余额，如果虚拟商店配置的不是游戏币，不要调用该接口。

```
//查询自定义游戏币余额
 [[NdComPlatform defaultPlatform] NdGetVirtualBalance:target];

//target 需要实现的回调方法
- (void)NdGetVirtualBalanceDidFinish:(int)error  balance:(NSString*)balance
{
       NSString* strBalance = nil;
       if (error >= 0) {
         strBalance = (balance != nil) ? balance : @"不支持游戏币";
       }
       else {
         strBalance = @"查询失败";
       }
      //TODO: 刷新界面提示
 }
```

18.　获取平台数据信息

1）获取当前应用的玩家列表

```
      NdPagination* pagination = [[NdPagination new] autorelease];
      pagination.pageIndex = 1;
      pagination.pageSize = 10;
     int nRes = [[NdComPlatform defaultPlatform] NdGetAppUserList:pagination
                                                         delegate:self];
```

pagination:分页信息，pageIndex 从 1 开始

delegate：获取应用用户列表的回调对象，回调对象必须实现以下函数：

```
- (void)getAppUserListDidFinish:(int)error
                         resultList:(NdStrangerUserInfoList *)userInfoList
{
      if (error < 0) {
         //TODO: 下载列表信息失败
      }
      else {
         //TODO: 处理信息列表
         //计算出总的页数，可以用来处理下一页的请求，如果nPageCount为0，可能需要特殊处理
         NdPagination *pagination = userInfoList.pagination;
         int nPageCount = (userInfoList.totalCount + pagination.pageSize - 1) /
pagination.pageSize;
```

```
            for (NdStrangerUserInfoList *info in userInfoList.records) {
                //...
            }
        }
}
```

error:错误码，正确为 0，详细请查看 NdComPlatformError.h

userInfoList：返回的用户信息列表，请查看 NdComPlatformAPIResponse.h 中的定义

2）获取当前应用的我的好友列表

```
    NdPagination* pagination = [[NdPagination new] autorelease];
    pagination.pageIndex = 1;
    pagination.pageSize = 10;
int nRes = [[NdComPlatform defaultPlatform] NdGetAppMyFriendList:pagination
                                                        delegate:self];
```

pagination:分页信息，pageIndex 从 1 开始

delegate：获取当前应用我的好友列表的回调对象，回调对象必须实现以下函数：

```
- (void)getAppMyFriendListDidFinish:(int)error
                         resultList:(NdFriendUserInfoList *)userInfoList
{
    if (error < 0) {
        //TODO: 下载列表信息失败
    }
    else {
        //TODO: 处理信息列表
        //计算出总的页数，可以用来处理下一页的请求，如果nPageCount为0，可能需要特殊处理
        NdPagination *pagination = userInfoList.pagination;
        int nPageCount = (userInfoList.totalCount + pagination.pageSize - 1) /
pagination.pageSize;
        for (NdFriendUserInfo *info in userInfoList.records) {
            //...
        }
    }
}
```

error:错误码，正确为 0，详细请查看 NdComPlatformError.h

friendUserInfoList：返回的好友信息列表，请查看 NdComPlatformAPIResponse.h 中的定义

3）获取我的好友列表

```
    NdPagination* pagination = [[NdPagination new] autorelease];
    pagination.pageIndex = 1;
    pagination.pageSize = 10;
```

```
int nRes = [[NdComPlatform defaultPlatform] NdGetMyFriendList:pagination
                                                      delegate:self];
```

pagination:分页信息，pageIndex 从 1 开始

delegate：获取我的好友列表的回调对象，回调对象必须实现以下函数：

```
- (void)searchMyFriendDidFinish:(int)error
                         resultList:(NdFriendUserInfoList *)friendUserInfoList
{
      if (error < 0) {
         //TODO: 下载列表信息失败
      }
      else {
         //TODO: 处理信息列表
         //计算出总的页数，可以用来处理下一页的请求，如果nPageCount为0，可能需要特殊处理
         NdPagination *pagination = friendUserInfoList.pagination;
         int nPageCount = (friendUserInfoList.totalCount + pagination.pageSize - 1) /
pagination.pageSize;

         for (NdFriendUserInfo *info in friendUserInfoList.records) {
             //...
         }
      }
}
```

error:错误码，正确为 0，详细请查看 NdComPlatformError.h

friendUserInfoList：返回的好友信息列表，请查看 NdComPlatformAPIResponse.h 中的定义

4）获取我的信息

A 获取本地简单信息

```
NdMyUserInfo *info = [[NdComPlatform defaultPlatform] NdGetMyInfo];
NSLog(@"Uin:%@ nickName:%@",info.baseInfo.uin,info.baseInfo.nickName);
```

获取我的基本信息，NdMyUserInfo 请查看 NdComPlatformAPIResponse.h 中的定义

B 获取网络详细信息

```
[[NdComPlatform defaultPlatform] NdGetMyInfoDetail:self];
```

获取我的详细信息，该接口与上面的接口不同，属于异步接口，需要在回调中才能获取到信息。delegate 回调函数:

```
//NdGetMyInfoDetail 和 NdGetUserInfoDetail 的回调
- (void)getUserInfoDidFinish:(int)error userInfo:(NdUserInfo *)userInfo
{
      if (error < 0 ) {
         //TODO: 下载用户信息失败
      }
      else {
```

```
        //userInfo中包含昵称，生日，地区，心情等详细数据
    }
}
```

error:错误码，正确为 0，详细请查看 NdComPlatformError.h
userInfo：返回我的详细信息，请查看 NdComPlatformAPIResponse.h 中的定义

5）捕获个人信息修改的通知

当用户修改了头像时，将发送 **kNdCPUserPortraitDidChange** 通知消息；
当用户修改了个人信息时， 将发送 **kNdCPUserInfoDidChange** 通知消息；
这两个消息在 NdCPNotifications.h 文件中声明，不附带变更内容。
如果是头像变更了，可以通过获取头像接口刷新头像；如果是信息变更了，通过获取我的信息接口更新数据。

6）获取用户的详细信息

```
[[NdComPlatform defaultPlatform] NdGetUserInfoDetail:uin flag:1 delegate:self];
```

获取用户的详细信息，参数介绍：
uin 指定用户的 uin，不能为空
flag 按位与标识符：1=基本信息，2=积分，4=心情。如果都不包含这三个标识位，返回参数错误。
delegate 回调接口为：

```
//NdGetMyInfoDetail 和 NdGetUserInfoDetail 的回调
- (void)getUserInfoDidFinish:(int)error userInfo:(NdUserInfo *)userInfo
{
    if (error < 0 ) {
        //TODO: 下载用户信息失败
    }
    else {
        //userInfo中包含昵称，生日，地区，心情等详细数据
    }
}
```

19. 好友操作

1）给好友发送消息

该接口只进行网络数据传输，不进入平台界面。
示例代码：

```
    //发送消息给好友
```

```
    int nRes = [[NdComPlatform defaultPlatform] NdSendFriendMsg:uin
                                    msgContent:msgContent delegate:target];
    if (nRes < 0) {
        //发送出错，可能是文本内容不合法，或者网络出错，或者未登录
    }

//target 需要实现的回调方法
- (void)sendFriendMsgDidFinish:(int)error msgId:(NSString *)msgId
{
    //TODO: 处理消息发送结果
    NSString* str = nil;
    if (error < 0) {
        switch (error) {
            case ND_COM_PLATFORM_ERROR_SERVER_RETURN_ERROR:
                str = @"服务器处理发生错误";
                break;
            case ND_COM_PLATFORM_ERROR_CONTENT_INVALID:
                str = @"内容不合法";
                break;
            case ND_COM_PLATFORM_ERROR_PERMISSION_NOT_ENOUGH:
                str = @"对方已经不是你的好友了";
                break;
            case ND_COM_PLATFORM_ERROR_USER_NOT_EXIST:
                str = @"该用户不存在";
                break;
            case ND_COM_PLATFORM_ERROR_CONTENT_LENGTH_INVALID:
                str = @"内容长度不合法";
                break;
            case ND_COM_PLATFORM_ERROR_NOT_ALLOWED_TO_SEND_MSG:
                str = @"发送者被禁止发消息";
                break;
            case ND_COM_PLATFORM_ERROR_CAN_NOT_SEND_MSG_TO_SELF:
                str = @"不能给自己发送消息";
                break;

            default:
                str = @"网络错误";
                break;
        }
    }
    else {
        //发送成功，返回该代表该消息的 id
    }
}
```

2）添加删除好友

如果想进行好友的“添加/删除”操作，可以调用该接口，进入指定用户的空间界面，那里有提供添加/删除好友的操作。该 uin 不能为空，不能是自己。

```
- (int) NdEnterUserSpace : (NSString *) uin ;
```

使用方式请参见进入指定用户的空间接口。

20. 获取头像/图标

1）简介

1.1)、获取头像和图标都可以指定分辨率大小，

ND_PHOTO_SIZE_TYPE 枚举值对应如下：

ND_PHOTO_SIZE_TINY 16 * 16 像素

ND_PHOTO_SIZE_SMALL 48 * 48 像素

ND_PHOTO_SIZE_MIDDLE 120 * 120 像素

ND_PHOTO_SIZE_BIG 200 * 200 像素

1.2)、开发者无需自己再做图片文件缓存。

因为对于头像和图标，平台里采用 checksum 统一的缓存机制。如果缓存图片的 checksum 与新的 checksum 比对不一致，则会删除缓存，重新从服务器下载新的图片并缓存；如果新的 checksum 与缓存 checksum 一致，或者新的 checksum 为空，那将使用缓存图片不再下载（如果初始没有缓存则需要从服务器下载并缓存）。对缓存的图片如果长期不再使用，那将会定期删除这些旧的图片，以免占用空间。

1.3)、关于 checksum

该参数一般是在获取用户的信息中包含该字段，用于及时更新用户的头像。该字段在应用信息，排行榜成就榜信息，虚拟商品信息等这些结构信息中一般都会包含 checksum，都是用来校验图标是否变化以便及时更新。如果开发者不关心图标的更新问题，在相应的 API 接口该字段可以传空。

注意：不同的 PHOTO_SIZE 对应不同的 checksum，不能混淆使用。目前所有的接口中返回的 checksum 会针对设备进行适配。如果设备为 ipod4、iphone4、ipad（高分辨率），则对应的是 120*120 规格(ND_PHOTO_SIZE_MIDDLE)，否则为 48*48 规格(ND_PHOTO_SIZE_SMALL)的规格。

2）获取好友的头像

2.1)、获取默认大小的头像（48*48）

示例代码：

```
//获取 ND_PHOTO_SIZE_SMALL 分辨率的头像
NSString* uin = ...; //指定用户
NSString* checksum = ...;//checksum 是在下载用户列表信息里的，如果没有可以为 nil，平台会优先使用缓存
```

```
[[NdComPlatform defaultPlatform] NdGetPortrait:uin checkSum:checkSum delegate:target];

//target 需要实现的回调方法
- (void)getPortraitDidFinish:(int)error uin:(NSString *)uin portrait:(UIImage *)portrait
              checkSum:(NSString *)checkSum
{
    if (portrait) {
        //TODO:  更新 uin 的头像
    }
}
```

1.2)、获取指定分辨率类型的头像

示例代码：

```
    NSString* uin = ...; //指定用户
    ND_PHOTO_SIZE_TYPE  imgType = ...;// 开发者根据自己需要获取指定分辨率的图像
    NSString* checksum = ...;//checksum 是在下载用户列表信息里的，如果没有，可以为 nil，平台会优先使用缓存
    [[NdComPlatform defaultPlatform] NdGetPortraitEx:uin imageType:imgType checkSum:checkSum
                                        delegate:target];

//target 需要实现的回调方法
- (void)getPortraitDidFinish:(int)error uin:(NSString *)uin portrait:(UIImage *)portrait
               checkSum:(NSString *)checkSum
{
    if (portrait) {
        //TODO:  更新 uin 的头像
    }
}
```

3）获取好友的头像（缓存文件）

该接口是为了支持开发者不使用 UIImage 内存格式，可以使用自己的图片载入方式，从缓存文件加载。

示例代码：

```
NSString* uin = ...; //指定用户
ND_PHOTO_SIZE_TYPE  imgType = ...;// 开发者根据自己需要获取指定分辨率的图像
NSString* checksum = ...;//checksum 是在下载用户列表信息里的，如果没有，可以为 nil，平台会优先使用缓存
[[NdComPlatform defaultPlatform] NdGetPortraitPath:uin imageType:imgType
                        checkSum:checkSum delegate:target];

//target 需要实现的回调方法
```

```
- (void)getPortraitPathDidFinish:(int)error uin:(NSString *)uin
                    portraitPath:(NSString *)portraitPath checkSum:(NSString *)checkSum
{
    //TODO:  从文件 portraitPath 载入头像
    UIImage*  portrait = [UIImage imageWithContentsOfFile:portraitPath];
    if (portrait) {
        //TODO:  更新 uin 的头像
    }
}
```

4）获取默认头像、默认应用图标

该接口返回的图片分辨率为 ND_PHOTO_SIZE_SMALL 规格。

示例代码：

```
//获取默认的用户头像
UIImage* imgDefaultUserIcon = [[NdComPlatform defaultPlatform] NdGetDefaultPhoto:1];

//获取默认的应用图标
UIImage* imgDefaultAppIcon = [[NdComPlatform defaultPlatform] NdGetDefaultPhoto:2];
```

5）获取榜图标

该接口用于获取排行榜、成就榜、虚拟商品的图标。

示例代码：

```
    ND_SNS_BOARD_TYPE boardType = ...;//榜 id 类型
    NSString* strId = ...; //榜 id，如果是获取排行榜图标，对应排行榜 id；虚拟商品图标，对应虚拟商品 id；……
    ND_PHOTO_SIZE_TYPE  photoType = ...;// 开发者根据自己需要获取指定分辨率的图像
    NSString* checksum = ...;//checksum 是在下载用户列表信息里的，如果没有，可以为 nil，平台会优先使用缓存
    [[NdComPlatform defaultPlatform] NdGetBoardIcon:strId boardType:boardType
             photoType:photoType checksum:checksum delegate:target];

//target 需要实现的回调方法
- (void)getBoardIconDidFinish:(int)error strId:(NSString*)strId
                  boardType:(ND_SNS_BOARD_TYPE)boardType
                  photoType:(ND_PHOTO_SIZE_TYPE)photoType
                  checksum:(NSString*)checksum image:(UIImage*)img
{
    if (img) {
        //TODO: 更新对应 strId 的图标
```

```
    }
}
```

6）获取应用图标

```
//目前只支持获取 ND_PHOTO_SIZE_SMALL 分辨率
NSString* strAppId = ...; //应用id
NSString* checksum = ...;//checksum，如果为空，默认使用本地缓存
[[NdComPlatform defaultPlatform] NdGetAppIcon:strAppId  checkSum:checkSum delegate:
target];

//target 需要实现的回调方法
- (void)getAppIconDidFinish:(int)error appId:(NSString *)strAppId icon:(UIImage *)icon
checkSum:(NSString *)checkSum
{
        //获取到strAppId的icon
}
```

7）获取应用图标（缓存文件）

```
//目前只支持获取ND_PHOTO_SIZE_SMALL分辨率
NSString* strAppId = ...; //应用id
NSString* checksum = ...;//checksum，如果为空，默认使用本地缓存
[[NdComPlatform defaultPlatform] NdGetAppIconPath:strAppId  checkSum:checkSum delegate:
target];

//target 需要实现的回调方法
- (void)getAppIconPathDidFinish:(int)error appId:(NSString *)strAppId iconPath:(NSString
*)iconPath checkSum:(NSString *)checkSum
{
        //获取到strAppId的iconPath
}
```

21.　屏幕截图

注：截图采用的是系统底层的 API，如果是用 openGL 的视图需要自己实现截图功能。

A 获取屏幕截图

```
[NdComPlatform NdGetScreenShot];
```

获取屏幕截图并返回

B 获取某视图的局域截图

```
[NdComPlatform NdGetViewCapture:self.view captureRect:rect];
```

view 是要截图的 UIView，captureRect 是要截取的区域，返回截取的图片。

六、 版本号设定规则

为了进行版本比较和升级，需要统一版本格式。

统一格式:

A.B.C.D（其中A，B，C，D为整数）

另外允许 A.B 等同于 A.B.0.0，A.B.C 等同于 A.B.C.0

比较优先级次序，优先比较A,其次比较B,再次C，最后D.

数字大的为高版本,

例如:

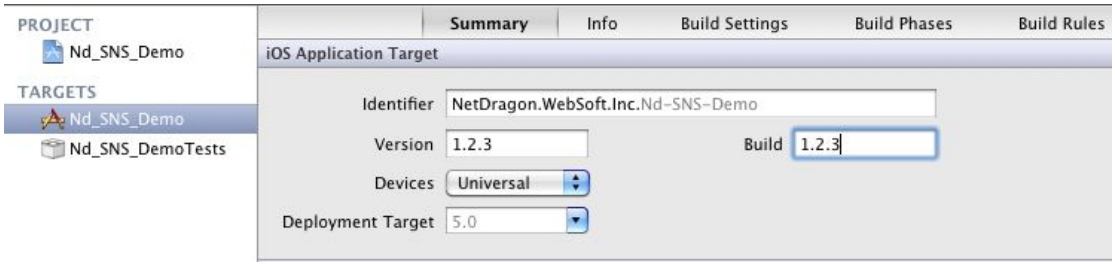
1.20 大于 1.2.1

2.0 大于 1.100

1.05 等于 1.5

设定说明:

在Info.plist中，有两个版本号相关的字段，分别是CFBundleShortVersionString和CFBundleVersion。其中的CFBundleShortVersionString对应图中的Version，而CFBundleVersion对应的是Build。



平台版本升级时：

- 使用CFBundleShortVersionString（Version）字段值进行比较
- 在没有CFBundleShortVersionString(Version)字段的情况下,才会将CFBundleVersion（Build）字段作为版本号进行比较
- 如果没有其他的特殊情况而同时设定两个字段时,建议开发者将这两个字段设定成相同的版本号

这两个字段都需要遵循上面规定的版本格式设定。发布新版本时两个字段都需要进行相应的更新，确保正确进行版本升级。

七、 FAQ

1. 为什么我不能保存自动登录的信息？

在模拟器上软件的 ipa 目录随着编译，有时候会发生改变，所以模拟器上如果不能自动登录，请用真机测试。

2. 为什么调用 91SDK 后，91 的资源没有正确载入？

Xcode 添加资源有两种方式，一种是散乱的放置在 app 目录下的方式，另一种是以文件夹存放的方式。由于第一种方式容易和应用程序的资源名字起冲突，所以从 2.5 版本开始，91SDK 不再支持第一种资源导入方式。请按照本手册中关于资源导入的部分，正确添加资源。

3. 如何查看当前程序所接入的 SDK 版本？

在 SDK3.1 版本之后，平台界面增加了查看所接的 SDK 版本等信息。查看方式：进入平台中的**“更多”**界面，进入**“关于我们”**列表项，里面展示了所接平台 SDK 的版本信息。

如果是**测试模式**，在“关于我们”界面，当前应用栏目里有**“当前为测试模式”**的字样。

4. 模拟器上测试版本更新的问题

模拟器上能测试新版本的检查，但是无法安装新版本的 ipa 包。

5. 如何制作 IPA 安装包？

把编译好并已经签名过的 xxx.app 文件拖到桌面的 iTunes 图标上，即可自动生成 IPA 安装包。在“应用程序”选项卡里找到生成后的包，点击该图标，从右键菜单中选择“在 Finder 中显示”，即可找到安装包文件。

也可以上网搜索其它详细的制作过程。

6. 登录、注销、切换账号监听的是哪个消息？

不管是登录，切换账号，还是注销，都是监听 kNdCPLoginNotification 收到这个消息时候，判断当前登录是否成功，如果不成功：

a. 如果之前有登录用户，那就是注销；

b. 如果之前没有登录用户，那就是用户还没有登录。

如果成功：

a. 如果之前有登录用户，那就是切换账号；

b. 如果之前没有登录用户，那就是第一个用户登录。

还有一个额外的要关注：如果收到 session 无效的通知，表明用户已经处于未登录状态了。

7. 在 iOS6 上进入 SDK 的修改头像界面，选择相册或者拍照后崩溃

该问题参照设定平台方向中提到的注意事项。

八、 场景案例

1. 游客登录

场景：

1. 玩家的设备未登录过 91 账号，并只是想先快速看下游戏好不好玩，再决定是否注册成为 91 用户继续玩。
2. 开发者想在玩家玩游戏的过程中通过 UIN 标识来保存玩家的游戏积分，等级等信息。
3. 开发者想在玩家处于游客登录状态且玩了 10 分钟后，或者玩家达到游戏等级 3 级时、或者玩家购买道具时、充值游戏币时再提醒玩家是否进行游客注册转正。

流程图：