关于苹果内购（IAP）的一些问题以及那些坑

标签： objective-c ios 苹果

2016-03-31 16:18 134人阅读 评论(0)

分类： ios开发（6） object-c（4）

首先，我们要去iTunes store创建几个我们需要在内购中使用到的产品，记住，产品的ID一定要唯一。苹果官方提到了，IAP购买项有几

- **Consumable products**：消耗类产品
- **Non-consumable products**：非消耗类产品
- **Auto-renewable subscriptions**：自动更新订阅产品
- **Non-renewable subscriptions.** 非自动更新订阅产品
- **Free subscriptions.** 免费订阅产

我们通常再游戏中用到的游戏币属于消耗类产品，赛车轨道等属于非消耗类产品，通常这2种会比较常见。我当时用的是消耗类产品。

当完成产品创建之后，去iTunes store申请一个测试账号，就要开始编写代码了。在编写代码之前，最重要的，是要了解整个内购实现

一个比较好的对<流程解说的帖子>，下面是流程图：

个人资料

叫我路飞

关注 发私信

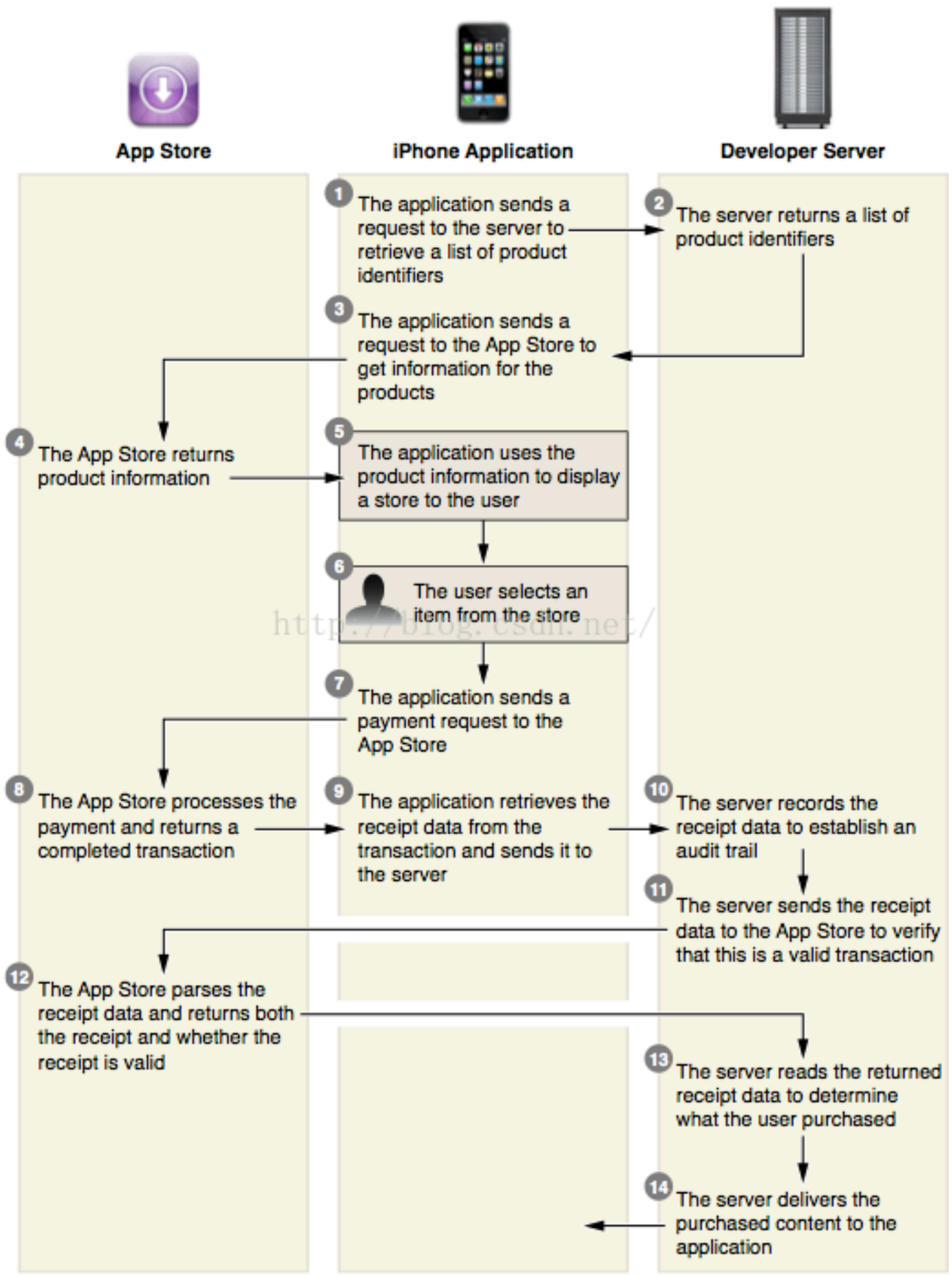
访问： 3577次
积分： 362
等级： BLOG 2
排名： 千里之外

原创： 32篇 转载： 0篇
译文： 0篇 评论： 0条

评论排行

推荐文章

Figure 1-3 Server product delivery



归根结底，其实，我们一直在和APP store在打交道，而并不是和苹果的服务器进行打交道，所以，大家要避免这个误区，而APP store

行打交道，这一层，其实我们基本是不需要考虑的。

流程：

1.首先，从图上的第一步，客户端向自己的服务器发送了一个请求，请求产品列表，然后，我们自己的服务器会返回给客户端产品的id

们在创建产品的时候，设置的产品ID，当获取之后，我们需要根据获得的identifiers向APP store请求产品的详细信息。但对于某些应用

类没有什么变动，所以，就直接将identifiers集成在了应用中，有的是直接放在了plist文件中，需要的时候，直接调用，不需要向服务器

单信息。但这样也有缺点，当产品发生变动的时候，需要发布新的版本，更新应用才行，所以，不推荐使用这种方案。

2.当获取了产品信息之后，要刷新UI，展示给用户，让用户选择需要购买那种产品，然后点击购买按钮。当用户购买某个产品的时候，

APP store发送购买请求，APP store接收到，购买请求之后，会进行订单的处理，然后，返回给我们购买的结果，同时，从上面的途中

到，返回到客户端有一个receipt data，这个东西其实是用来进行校验的证书（其实是很长的字符串，大概3000多个字符吧），防止有人

而反复获取我们的产品，尤其是类似金币这种。

3.当客户端获得购买结果之后，将支付信息（包括验证证书）发送到服务器，服务器向AppStore发起验证，这个验证必须是post请求，

发送过去，同时，receipt要进行base64编码，当苹果确认之后，会给我们返回状态码，告诉我们是否成功。

Table 2-1 Status codes

Status Code	Description
21000	The App Store could not read the JSON object you provided.
21002	The data in the `receipt-data` property was malformed or missing.
21003	The receipt could not be authenticated.
21004	The shared secret you provided does not match the shared secret on file for your account. *Only returned for iOS 6 style transaction receipts for auto-renewable subscriptions.*
21005	The receipt server is not currently available.
21006	This receipt is valid but the subscription has expired. When this status code is returned to your server, the receipt data is als returned as part of the response. *Only returned for iOS 6 style transaction receipts for auto-renewable subscriptions.*
21007	This receipt is from the test environment, but it was sent to the production environment for verification. Send it to the test e instead.
21008	This receipt is from the production environment, but it was sent to the test environment for verification. Send it to the produ environment instead.

这是苹果官方给出的集中状态值，苹果返回回来的数据也是json格式的，会有一个state字段，当为0的时候，表示成功，我们测试的接

是：`https://sandbox.itunes.apple.com/verifyReceipt`，生产环境的接口是：`https://buy.itunes.apple.com/verifyRec`

，所以大家要区分好这两个接口。21007表示将测试环境获得receipt发送到了生产环境，21008表示将生产环境的receipt发送到了测试

值，应该都表示验证失败，但是，具体是什么，我也不清楚，英语好的话，可以自己翻译一下，然后，告诉我。这里是<苹果官方验证

看这个，写出客户端验证的代码。因为我不是做服务端的，所以不知道怎么写服务端验证，但是，这两者应该是相通的，大家可以在下

论一下。

4.当服务器从APPstore获得返回状态后，判断是否有这条购买记录，如果有，就更新服务器端**数据库**，表示物品已经购买，再给客户端

里说的APPstore，我再网上找了好多资料，都是这么说的，但我觉得其实就是苹果服务器给提供的接口，只不过为了方便，所以，在面

成了向APP store发送验证，其实，这里是苹果服务器提供的一个接口。

笔者公司当时用的是RMStore这个开源库，这个用着很方便，所以，大家也可以尝试一下，但不保证完全没有问题，因为我在使用的过

了一些棘手的问题。大家也可以自己写支付这个模块，其实，正常的这个流程也不是很麻烦，先把基本流程写完，再考虑可能出现的问

我在上面引用的几个链接里面，有的链接里面有具体的代码，大家可以参考一下。

用RMStore的话，主要会调用这样一个方法：

```
[html]
01. - (void)addPayment:(NSString*)productIdentifier
02.                user:(NSString*)userIdentifier
03.             success:(void (^)(SKPaymentTransaction *transaction))successBlock
04.             failure:(void (^)(SKPaymentTransaction *transaction, NSError *error))failureBlock;
```

先来说说这些参数吧。首先，第一个参数，这个就是我们获取到的产品的identifier，就是要购买的那个产品的唯一标识；然后是这个u

这里的success block，实在支付成功后，回调的内容，只要把成功后进行的操作写在里面就可以了，但是，由于成功后，需要的操作

把操作封装一下，在里面调用，否则，逻辑会很乱，而且，下面的failure block中还要对很多异常状况进行判断和处理，其中有一个就

store”，这个问题很麻烦，后面会具体说。一般情况下，如果不考虑user这个变量，可以直接使用下面的方法：

```
[html]
01. - (void)addPayment:(NSString*)productIdentifier
02.             success:(void (^)(SKPaymentTransaction *transaction))successBlock
03.             failure:(void (^)(SKPaymentTransaction *transaction, NSError *error))failureBlock;
```

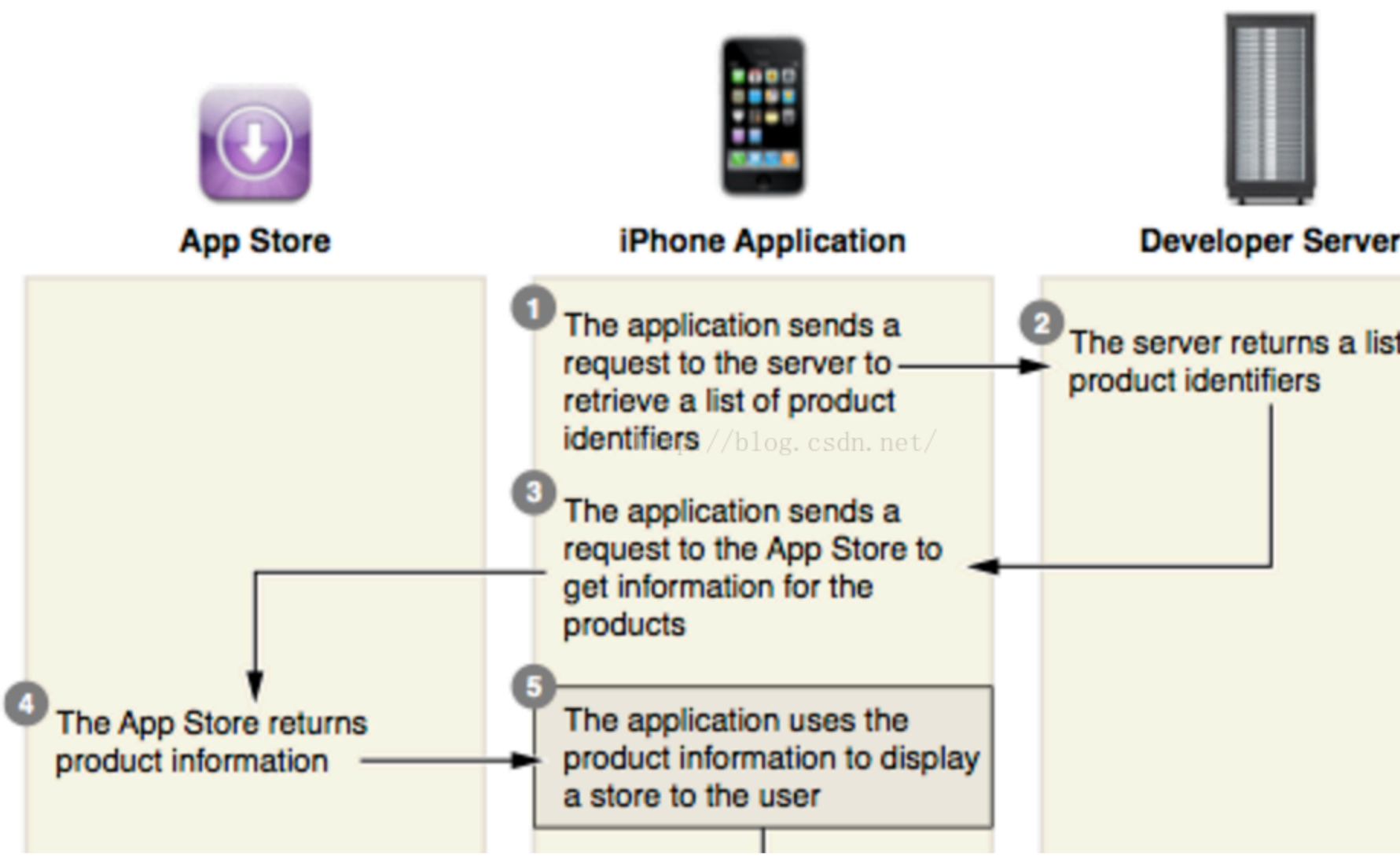
这个方法要调用上面的方法，但是user默认为nil。

支付流程看起来就是这样，感觉好像很简单，但是，这里面的问题其实很大。上面只是在一切都正常的状态下，才会走的流程，但是，

题、断网、应用闪退，有越狱插件等问题，问题就麻烦了，这个历程，各个过程中需要考虑的问题，其实，还是很多的。好的，下面我

说IAP实现过程中的各种坑。先重新把上面的图拿过来。

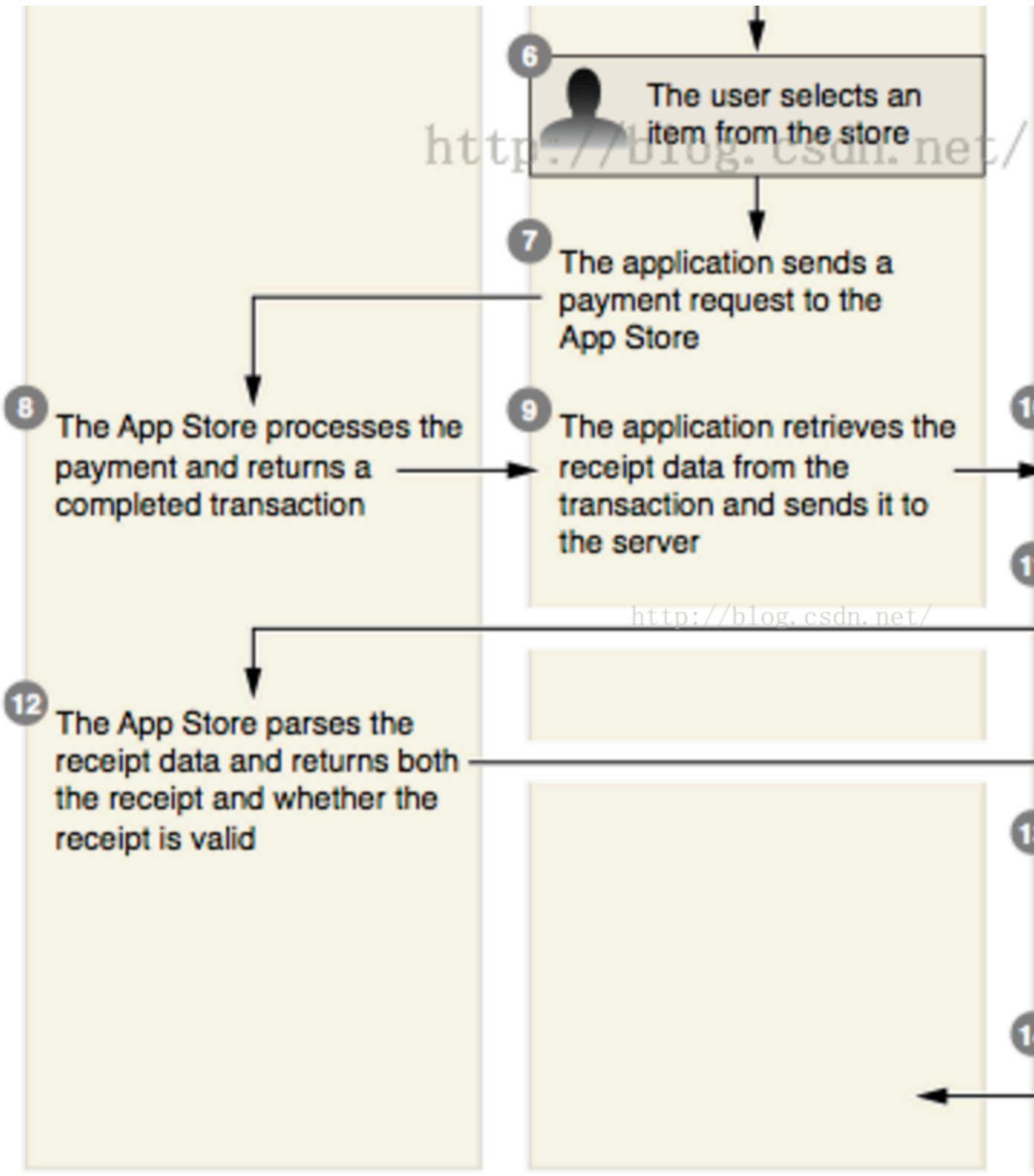
首先来说第一步：



这一步还是很轻松的，我们向服务器获取产品identifiers，由于需要进行网络请求，而且是支付，所以，一定要把断网考虑进来，这个是

这一步，我们要判断，当没有获取数据的时候，要提示用户暂时没有获取产品列表信息，这部分其实还好，不需要考虑太多。

之后的一些过程，就比较复杂了，考虑的东西也会比较多了。首先把剩下的部分拿过来：



这部分问题很多，而且，需要逻辑也很复杂。首先说第七步，这一部分，再应用中，当点击购买的时候，会弹出输入框，要求输入账号

消的时候，实际上会调用failure block.调用failure block的时候，会获得一条支付信息transaction和一个error，我们可以判断transaction

断取消状态，

```objc
if (transaction.error.code == SKErrorPaymentCancelled)
```

也就是判断这个订单信息的error的code值，这个就是取消状态。但实际上，这只是一种比较常见的状态。当用户再购买的过程中，如果

然断网了，或者请求支付的订单状态有问题，也就是上面的过程⑦出现了问题，就会触发其他的几种状态，这个时候，如果只是输出订

出现“无法连接到iTunes store”，这是一种很让人头疼的状态，因为，你根本不知道到底是什么问题，到底是怎么无法连接到iTunes sto

问题坑了，后来发现，这其实是一种请求失败，和SKErrorPaymentCancelled类似。SKErrorPaymentCancelled和其他几种状态其实是

```objc
NS_ASSUME_NONNULL_BEGIN

SK_EXTERN NSString * const SKErrorDomain NS_AVAILABLE_IOS(3_0);

// error codes for the SKErrorDomain
enum {
    SKErrorUnknown,
    SKErrorClientInvalid,               // client is not allowed to issue the request, etc.
    SKErrorPaymentCancelled,            // user cancelled the request, etc.
    SKErrorPaymentInvalid,              // purchase identifier was invalid, etc.
    SKErrorPaymentNotAllowed,           // this device is not allowed to make the payment
    SKErrorStoreProductNotAvailable,    // Product is not available in the current storefront
};

NS_ASSUME_NONNULL_END
```

属于同一类问题，也就是上面说的无法连接到iTunes store，虽然知道了这几种状态，但是，还是不知道这几种状态到底代表什么。于

档里面看了一下，

```objc
Constants
SKErrorUnknown
Indicates that an unknown or unexpected error occurred.

Available in iOS 3.0 and later.
SKErrorClientInvalid
Indicates that the client is not allowed to perform the attempted action.

Available in iOS 3.0 and later.
SKErrorPaymentCancelled
Indicates that the user cancelled a payment request.

Available in iOS 3.0 and later.
SKErrorPaymentInvalid
Indicates that one of the payment parameters was not recognized by the Apple App Store.

Available in iOS 3.0 and later.
SKErrorPaymentNotAllowed
Indicates that the user is not allowed to authorize payments.

Available in iOS 3.0 and later.
SKErrorStoreProductNotAvailable
Indicates that the requested product is not available in the store.

Available in iOS 6.0 and later.
```

这是官方的解释，可以尝试翻译一下，了解其代表的含义。后来在网上搜索了一下相关的文章，只找到一个，说了<无法连接到iTunes

几种状态，并没有全部涵盖，后来我在网上又找了一下，下面是我给出的对无法连接到iTunes store的处理：

```objc
if (transaction.error != nil) {
    switch (transaction.error.code) {

        case SKErrorUnknown:

            NSLog(@"SKErrorUnknown");
            detail = @"未知的错误，您可能正在使用越狱手机";
            break;

        case SKErrorClientInvalid:

            NSLog(@"SKErrorClientInvalid");
            detail = @"当前苹果账户无法购买商品(如有疑问，可以询问苹果客服)";
            break;

        case SKErrorPaymentCancelled:

            NSLog(@"SKErrorPaymentCancelled");
            detail = @"订单已取消";
            break;
```

```
        case SKErrorPaymentInvalid:
            NSLog(@"SKErrorPaymentInvalid");
            detail = @"订单无效(如有疑问，可以询问苹果客服)";
            break;

        case SKErrorPaymentNotAllowed:
            NSLog(@"SKErrorPaymentNotAllowed");
            detail = @"当前苹果设备无法购买商品(如有疑问，可以询问苹果客服)";
            break;

        case SKErrorStoreProductNotAvailable:
            NSLog(@"SKErrorStoreProductNotAvailable");
            detail = @"当前商品不可用";
            break;

        default:

            NSLog(@"No Match Found for error");
            detail = @"未知错误";
            break;
    }
}
```

这个SKErrorUnknown实在是很难处理，我找了好多的帖子，包括stackoverflow，也没看到太多的说法，有一些说可能是越狱手机，才

在测试的时候，我们通常也会遇到这种问题。测试的时候，我们要再iTunes connect申请测试账号，有的时候，测试账号出问题，或者

取消了，不再使用了，而支付的时候，仍然在使用这个测试账号，这个时候，也会出现unknown状态。

当然，失败有很多种，这是无法连接到iTunes store，不是网络的问题。上面提到失败的时候，会有transaction和error两个返回值，当

候，error.code是负值。这时，成功的话，没有这个error信息，这时，我们就可以判断到底是怎么回事了，当返回了error的时候，先判

是否为空，不为空的话，进行上面的switch判断，为空的话，说明交易的订单信息没有问题，这时候，就只是网络的问题了，就提示用

当我们向AppStore发送了请求之后，如果AppStore交易完成之后，也就是上面的成功的success block，我们首先要将订单信息保存到

们自己的服务器，当我们的服务器给我们返回信息的时候，我们再更新UI，同时，删除本地保存的订单信息。这个订单信息，可以保存

以保存在文件中，但是，苹果建议保存在文件中，用NSCoding进行编码保存，这样会更好一些。

向自己服务器发送消息的话，还要注意很多东西。这里面也包括AFNetworking的一些问题。但我不了解这是不是偶发的事情。当时出现

的：Error Domain=com.alamofire.error.serialization.response Code=-1016 "Request failed: unacceptable content-

type: text/html"

```
[html]
01. Error Domain=com.alamofire.error.serialization.response Code=-1016 "Request failed: unacceptable c
    type: text/html"
```

我当时还不知道这是怎么回事，后来在网上找了一些资料，才了解到，这是AFNetworking对网络请求的数据类型的一种支持问题，下面

诉大家怎么解决这个问题：

Error Domain=com.alamofire.error.serialization.response Code=-1016 "Request fai

unacceptable con

当出现这种问题的时候，订单信息会无法上传到自己的服务器，这时候，就出问题了，用户已经支付了，钱已经扣了，但是，我们的服

息，所以，无法给用户发货，类似这样损害用户利益的事情是绝对不被允许的。所以，可以按照上面帖子的说法，修改请求类型，添加

持，就可以避免这种问题了。此外，当我们自己的服务器出错的时候，当用户打算将订单信息上传到我们服务器的时候，此时，服务器

们预先设定好的状态码，对于这种状态，我们也要在客户端进行相应的判断，当遇到这样的问题的时候，提示用服务器出错，赶紧联系

问题的解决。

上面说到，我们向APP store发送支付请求的时候，当支付完成的时候，服务器会将订单返回给我们，这个时候，我们首先应该做的，

存到本地，然后，再向我们自己的服务器发送订单信息，当服务器给我们反馈信息，通知我们成功之后，再删除本地保存的订单信息。

们这里要设置一个定时器，将未完成的失败订单，定时提交到我们的服务器，从而获得要购买的商品。但是，如果一直没有网络怎么办

在每次应用打开的时候，查询是否有未完成的订单信息，然后将订单信息上传到服务器，从而获得我们要购买的商品。

这种状态处理完了之后，还有其他的一些状态，例如，网络状态不好的状态下，当我们向APP Store发起订单请求的时候，请求成功了

Store给我们返回订单的时候，断网了，或者，此时退出了应用，以及应用闪退，那该怎么办呢？其实，苹果已经替我们想好了这种问题

只要在应用启动的时候，设置一下代理，就可以了，这是<官方文档>，我们需要在应用启动的时候，设置SKPaymentQueue的代理方法

```objc
[[SKPaymentQueue defaultQueue] addTransactionObserver:self];
```

并实现代理方法

```objc
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions
{
    // Attach an observer to the payment queue
    [[SKPaymentQueue defaultQueue] addTransactionObserver:self];
    return YES;
}

// Called when the application is about to terminate
- (void)applicationWillTerminate:(UIApplication *)application
{
     // Remove the observer
    [[SKPaymentQueue defaultQueue] removeTransactionObserver:self];
}
```

当订单状态发生状态的时候，会异步调用这个方法，从而，通知我们更新订单，并上传订单信息到服务器，给用户发货。如果使用RM

需要我们手动实现，因为RMstore就是这个observe，所以，在应用启动的时候，我们就应该对RMSotre这个单例进行初始化。

下面来说下面这个方法：

```html
- (void)addPayment:(NSString*)productIdentifier
             user:(NSString*)userIdentifier
          success:(void (^)(SKPaymentTransaction *transaction))successBlock
          failure:(void (^)(SKPaymentTransaction *transaction, NSError *error))failureBlock;
```

这个方法里里面有一个user属性，是用来用户自定义的字段，当我们发送支付请求的时候，发送这个字段之后，当获取了支付成功的请

会原封不动的返回回来。当我们用同一个手机，登录了2个不同的账号的时候，这个字段就非常有用了。正常情况下，当我们获得支付

订单信息上传到自己的服务器，那么，怎么确定就是是哪个用户呢，默认情况下，我们会把保存在本地的用户账号，也一起返回给自己

我们假设这样一种状况：我们现在有一个手机，A在上面下单了，订单已经发送给APP store，这个时候，断网了，还没接到APP store

果，这个时候，A退出了账号，过了一会儿，有网络了，B登录了，这个时候，如果订单返回了，那么，我们正常状态下，需要把这个订

务器中。那么，问题来了，我们此时无法或得到下订单的A的用户信息。如果还是按照默认的状态，此时，会把B的账号信息一起发送到

样，就出错了，A买的东西，没得到，B没有买，却得到了。这是不合理的。所以，我们要在向APP store发送支付请求的时候，一起把

过去，也就是保存在上面的那个user字段值，当获得APP store的反馈的时候，再将用户信息一起取出来，然后发送到服务器，这样，就

那种问题了。

以上就是我对最近开发中遇到的一些问题的解决，有不全面的地方和说错的地方，还请大家批评指点。

- 上一篇 AFNetworking读取和设置cookie的解决方案
- 下一篇 IOS热更新 - JSPatch实现原理+Patch现场恢复

我的同类文章

ios开发（6） object-c（4）

- Masonry的简单使用 2016-05-17 阅读 15
- 远程推送 2016-01-29 阅读 87
- 安装iOS开发环境Xcode 2016-01-13 阅读 48
- IOS热更新 - JSPatch实现原理+Patch... 2016-05-03 阅
- obj-c内存管理的规则 2016-01-18 阅
- Mac中的简单快捷键 2016-01-13 阅

参考知识库

MySQL知识库

8396 关注 | 1396 收录

Swift知识库

1722 关注 | 386 收录

猜你在找

- 从零练就iOS高手实战班
- 基于Spring MVC+MyBatis+FreeMa…
- QuickTest Professional深入剖…
- 老郭全套iOS开发课程【UI技术】
- 微信平台二次开发入门
- 从零练就iOS高手实战班
- 基于Spring MVC+MyBatis+FreeMa…
- 软件测试工程师面试前突击——1…
- 老郭全套iOS开发课程【UI技术】
- 韦东山嵌入式Linux第一期视频

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VPN S
IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring Apache
HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails
Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSi
Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angular
Cloud Foundry Redis Scala Django Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持