

iOS推送小结（证书的生成、客户端的开发、服务端的开发）



作者 寒桥 (/u/ee0520f2d909) [+ 关注](#)

2015.04.15 16:26 字数 1802 阅读 1837 评论 0 喜欢 22 阅读 1837 评论 0 喜欢 22

(/u/ee0520f2d909)

1.推送过程简介

(1) App启动过程中，使用UIApplication::registerForRemoteNotificationTypes函数与苹果的APNS服务器通信，发出注册远程推送的申请。若注册成功，回调函数

application:(UIApplication *)application

didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken 会被触发，App可以得到deviceToken，该token就是一个与设备相关的字符串。

(2) App获取到DeviceToken后，将DeviceToken发送给自己的服务端。

(3) 服务端拿到DeviceToken以后，使用证书文件，向苹果的APNS服务器发起一个SSL连接。连接成功之后，发送一段JSON串，该JSON串包含推送消息的类型及内容。

(4) 苹果的APNS服务器得到JSON串以后，向App发送通知消息，使得App的回调函数application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo被调用，App从userInfo中即可得到推送消息的内容。

2. 用到的证书文件及生成过程

(1) certSigningRequest文件，该文件在MAC系统中生成，用于在Apple网站上申请推送证书文件。

生成过程：

打开应用程序中的“钥匙串访问”软件，从菜单中选择“钥匙串访问”-》“证书助理”-》“从证书颁发机构请求证书”，邮箱和名称随便填写，然后选择保存到磁盘，就可以在本地生成一个CertificateSigningRequest.certSigningRequest文件。

(2) 注册一个支持push的app id，后面会用到。

生成过程：

进入developer.apple.com，选择member center - Certificates, Identifiers & Profiles - Identifiers- App Ids，然后选择注册app id，设置appid名称，同时，app id suffix一栏必须选择explicit app id，然后设置bundle id，最后勾选 App Services中的 Push Notifications，这样就可以注册一个支持push的appid。

(3) 推送证书cer文件，该文件在developer.apple.com中生成，用于生成服务端需要的文件。

生成过程：

进入developer.apple.com，选择member center - Certificates, Identifiers & Profiles - Certificates，然后选择创建certificate，类型分为Development和Product。这里以Development为例，选择Apple Push Notification service SSL (Sandbox)，然后下一步，选择之前生成的支持push的AppId，然后下一步，提交之前创建的CSR文件，再下一步就可以生成cer文件，然后保存到本地。

(4) 生成服务端使用的证书文件。如果是使用网上的mac 版PushMeBaby工具，在mac 机器上进行推送消息的发送，那么有上面的cer文件就够了。如果是使用PHP、java/c#开发自己的服务端，那么还需要将上面的cer文件做一个转换，生成pem文件或者p12文件。

生成php用的pem文件过程为：

首先双击前面保存的cer文件，此时会打开“钥匙串访问”软件，里面会出现一个Apple Development IOS push services证书，一个公用密钥和一个专用密钥，密钥的名称与证书助理中填写的名称一致。

选中证书，导出为 apns-dev-cert.p12 文件

选中专有密钥，导出为apns-dev-key.p12文件

通过终端命令将这些文件转换为PEM格式：

```
openssl pkcs12 -clcerts -nokeys -out apns-dev-cert.pem -in apns-dev-cert.p12
```

```
openssl pkcs12 -nocerts -out apns-dev-key.pem -in apns-dev-key.p12
```

最后， 需要将两个pem文件合并成一个apns-dev.pem文件，此文件在连接到APNS时需要使用：

```
cat apns-dev-cert.pem apns-dev-key-noenc.pem > apns-dev.pem
```

生成java/c#用的p12文件过程为：

```
openssl pkcs12 -clcerts -nokeys -out apns-dev-cert.pem -in apns-dev-cert.p12
```

```
openssl pkcs12 -nocerts -out apns-dev-key.pem -in apns-dev-key.p12
```

```
openssl pkcs12 -export -in apns-dev-cert.pem -inkey apns-dev-key.pem -certfile  
CertificateSigningRequest.certSigningRequest -name "push" -out push.p12
```

(5) 生成XCODE使用的provisioning文件,该文件用于真机调试。

生成过程：

进入developer.apple.com，选择member center - Certificates, Identifiers & Profiles - Provisioning Profiles，然后选择创建Provisioning file，接着选择iOS App Development，下一步选择AppId，选中之前建立的支持push的appid，接着下一步选择支持push的certificate，下一步勾选需要支持的device id，最后一步设置provisioning文件的文件名，这样provisioning文件就生成了。

3. 服务端的开发

(1) 如果只是希望在mac电脑上测试一下消息的推送，可以使用PushMeBaby工具，使用起来比较简单。该工具是开源的，可以从<https://github.com/stefanhafener/PushMeBaby> 下载，代码的执行过程实际上就是设置一下SSL证书，然后连接APNS，接着发送JSON数据。由于要处理SSL逻辑，因此代码稍微多点。在使用工具时，将工程资源中的cer文件替换成自己的cer文件，然后将代码中的deviceToken替换成自己设备的deviceToken即可。

(2) 使用php开发服务端

由于php已经内置了ssl模块，因此使用php连接APNS服务器来发送json的过程实际上是很简单的，代码如下：

该文件可以放到服务器中通过浏览器来访问,也可以通过命令行的方式来解释执行, 代码为: \$ php -f Pusher.php

复制代码

```
array("alert" => 'message',"badge" => 2,"sound"=>'default')); //推送方式, 包含内容和声音
$$ctx = stream_context_create();
```

//如果在Windows的服务器上, 寻找pem路径会有问题, 路径修改成这样的方法:

```
//$pem = dirname(__FILE__) . '/' . 'apns-dev.pem';
```

//linux 的服务器直接写pem的路径即可

```
stream_context_set_option($ctx,"ssl","local_cert","apns-dev.pem");
```

```
$pass = "xxxxxx";stream_context_set_option($ctx, 'ssl', 'passphrase', $pass);//
```

//此处有两个服务器需要选择, 如果是开发测试用, 选择第二名sandbox的服务器并使用Dev的pem证书, 如果是正式发布, 使用Product的pem并选用正式的服务器

```
$fp = stream_socket_client("ssl://gateway.push.apple.com:2195", $err, $errstr, 60, STREAM_CLIENT_CONNECT, $ctx);
```

```
$fp = stream_socket_client("ssl://gateway.sandbox.push.apple.com:2195", $err, $errstr, 60, STREAM_CLIENT_CONNECT, $ctx);
```

```
if (!$fp)
```

```
{echo "Failed to connect $err $errstrn";return;}
```

```
print "Connection OK\n";
```

```
$payload = json_encode($body);$msg = chr(0) . pack("n",32) . pack("H*", str_replace(' ', "", $deviceToken)) . pack("n",strlen($payload)) . $payload;
```

```
echo "sending message :". $payload ."\n";
```

```
fwrite($fp, $msg); fclose($fp);
```

```
?>
```

复制代码

4. 客户端的开发

(1) 下载前面建立的cer文件和provisioning文件, 双击, 导入到xcode中, 在build setting中code signing一栏里选择这两个文件的名称, 这样就可以将支持push的app部署到真机中。

(2) 处理推送消息

客户端对推送消息的处理分两种情况:

一. 在App没有运行的情况下, 系统收到推送消息, 用户点击推送消息, 启动App。此时, 不会执行前面提到的 didReceiveRemoteNotification函数, 而是在App的 applicationDidFinishLaunching函数中处理推送, 通过以下代码可以获取推送消息中的数据: NSDictionary *userInfo = [launchOptionsobjectForKey:UIApplicationLaunchOptionsRemoteNotificationKey];

二 . 当APP处于前台时，系统收到推送消息，此时系统不会弹出消息提示，会直接触发 application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo函数，推送数据在userInfo字典中。

当App处于后台时，如果系统收到推送消息，当用户点击推送消息时，会执行 application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo函数，

此时AppDelegate中函数执行的顺序为：

applicationWillEnterForeground

application:didReceiveRemoteNotification

applicationDidBecomeActiveI



寒桥 (/u/ee0520f2d909)

写了 67283 字，被 282 人关注，获得了 305 个喜欢 (/u/ee0520f2d909)写了 67283 字，被 282 人关注，获得了 305 个喜欢

+ 关注

iOS工程师、前端工程师

如果觉得我的文章对您有用，请随意打赏。您的支持将鼓励我继续创作！

赞赏支持


♡ 喜欢 (/sign_in)

|

22







更多分享

(http://cwb.assets.jianshu.io/notes/images/1115489/



(/sign_in)发表评论

评论

智慧如你，不想发表一点想法 (/sign_in)咩~

被以下专题收入，发现更多相似内容

