**一、证书制作**

在北向推送的时候，如果是https方式，平台需要加载证书到ca.pem文件里面，但是厂商提供的证书格式有各种各样的，平台要求使用的是.pem格式的，这里提供转换证书的指导，来帮助加载证书。  
1. 如果提供的是.keystore格式：（jks也可以用这种方式转换）

首先把证书转换成pkcs12格式keytool -importkeystore -srckeystore debug.keystore -destkeystore tmp.p12 -srcstoretype JKS -deststoretype PKCS12

这一步在CMD下或者在环境上都可以进行，然后把PKCS12 dump成pem格式：openssl pkcs12 -in tmp.p12 -nodes -out sample.pem 这一步需要在环境上进行

2. 如果提供的是.cer格式：直接执行openssl命令即可，openssl x509 -inform der -in certificate.cer -out certificate.pem，然后生成的pem文件就可以直接用txt打开，看到BEGIN CERTIFICATE和END CERTIFICATE这两行之间即为一个证书，将这个从BEGIN复制到END，然后贴在/opt/oss/IoM/apps/loadbalance-1.1.2.100/etc/crt/platform/naserver/ 该路径下的ca.pem文件里，注意不能和别的BEGIN-END交叉重合。然后重启loadbalance进程即可推送数据到https回调地址。可以使用以下命令来验证证书的有效性：openssl s\_client -connect IP:PORT -tls1 -CAfile ca.pem，其中IP为回调地址IP，PORT为回调地址端口。如果返回0（OK）表示验证成功

**二、双向认证的部署和验证方法（用于应用订阅平台数据通知场景）**

介绍利用tomcat配置https双向认证。此方法仅供参考。

1.找到jdk安装目录，运行控制台，切换到该目录；

2.使用keytool为tomcat生成证书;

keytool -genkey -v -alias tomcat -keyalg RSA -keystore tomcat.keystore -validity 36500

3.为客户端生成证书;

keytool -genkey -v -alias huawei -keyalg RSA -storetype PKCS12 -keystore huaweitest.p12 -validity 36500

4.将huaweitest.p12导入到tomcat的信任证书链中

keytool -export -alias huawei -keystore huaweitest.p12 -storetype PKCS12 -rfc -file huaweitest.cer

keytool -import -alias huawei -v -file huaweitest.cer -keystore tomcat.keystore

5.从tomcat的证书链里导出跟证书

keytool -export -v -alias tomcat  -file CA.cer -keystore tomcat.keystore

6.将华为的outgoing.Cert.pem导入tomcat的信任证书链

keytool -import -v -file outgoing.Cert.pem -alias huawei\_out -keystore tomcat.keystore

7.将华为的ca.pem导入tomcat的信任证书链

keytool -import -v -file ca.pem -alias huawei\_ca -keystore tomcat.keystore

8.配置tomcat

双向认证：

单向认证：

9.配置完后，可以在本地验证配置是否成功。

在服务器上进行格式转换成pem格式

openssl x509 -inform der -in CA.cer -out ca.pem

通过以下命令模拟与应用服务器建链

单向认证模拟建链：

openssl s\_client -connect ip:port -tls1 -CAfile ca.pem

  
双向认证模拟建链：

openssl s\_client -connect ip:port -cert huaweitest.pem -CAfile CA.pem -tls1

10.将生成的pem格式证书通过sp portal(开发者portal)导入。证书导入后需要应用服务器再调用一下订阅接口才能生效。

三、**用IKVMC将jar转成dll供c#工程调用**

IKVM.NET是一个针对Mono和微软.net框架的java实现，其设计目的是在.NET平台上运行java程序。

有些在java程序上可以实现的功能，但在C#中无法实现或者目前不知道怎么实现时（如带有特殊Padding的RSA2048加密，https双向认证等），可以考虑使用IKVM工具把已生成的jar转成dll供C#程序去调用。

IKVM官网地址：https://www.nuget.org/packages/IKVM/

由于IKVM7最高只支持jdk1.6编译出来的jar转换，建议使用最新版本IKVM8。

**使用说明**

前提条件：已通过java实现所需功能，且编译成一个jar，如MyHttpClient.jar。

1. 在官网下载ikvm压缩包，这里用的是ikvmbin-8.1.5717.0.zip，解压到一个文件夹中，如C:\tools\ikvm-8.1.5717.0。

2. 配置环境变量：在系统环境变量中增加IKVM\_HOME：C:\tools\ikvm-8.1.5717.0，同时Path中增加%IKVM\_HOME%/bin;

3. 把MyHttpClient.jar放到一个工作目录下，如C:\tools\ikvm，在命令行中进入该目录，执行 ikvmc -target:library MyHttpClient.jar，即在当前目录下生产MyHttpClient.dll，这里只需要生成一个dll，如需生产可执行程序，可执行命令 ikvmc -target:exe xxx.jar。

4. 在C#工程中引用MyHttpClient.dll以及IKVM相关的dll（C:\tools\ikvm-8.1.5717.0\bin目录下的以IKVM.开头的dll，这里为了方便全部引用了），C#中类的命名空间和java中的类的包名是一致的，如java中import com.huawei.myhttpclient，C#中为using com.huawei.myhttpclient

**特别注意：**

由于实际的java实现工程很多情况下都依赖一些第三方jar，比如举例的MyHttpClient工程使用了Apache httpClient相关组件实现的设置客户端，服务端证书，发送消息功能，依赖的jar有：commons-codec-1.9.jar，commons-logging-1.2.jar，httpclient-4.5.2.jar，httpcore-4.4.4.jar，httpmime-4.5.2.jar，在网上查询很多做法是把这些第三方jar依次编译成dll，执行ikvm命令时需要指定指定依赖的dll，比较麻烦，这里的做法是把这些jar用winrar等解压软件打开，然后里面的根目录文件夹org（里面包含了Apache httpClient编译生成的class）,拷到MyHttpClient.jar中。