IoT平台北向应用证书使用指南

为了保障在业务中鉴权、密码、业务数据等敏感信息的传输安全。IoT平台提供HTTPS接口，与北向Application（以下简称NA）对接。

# 基础知识（必读）

## 公钥和私钥

公钥和私钥可理解为两个有关系的数字密码。经公钥加密的数据只可以被私钥解密。经私钥加密的数据也只可以被公钥解密。通常情况下公钥是对外公开的，私钥是保密的。

例：A与B通信，通信内容不想被他人知道，可以采用如下方式进行通信

1. A与B互相交换公钥（A与B各自有自己的公钥和私钥）。
2. A使用B的公钥加密通信内容后发给B（这样只有B手上的私钥才能解密）。
3. B收到A发的信后，使用自己的私钥解密，再用A的公钥对回信内容进行加密后发给A。
4. A收到B的回信，使用自己的私钥进行解密。

## 数字证书

一个数字证书包含的内容有：1、一个公钥（证书使用者的公钥） 2、证书的基本信息，主要包括使用者的名字(通常是IP或域名)、证书有效期、加密算法等公开信息 3、签发该数字证书的颁发者（称为**CA**）的信息和**签名**（使用颁发者的私钥加密的一段数据，只有使用颁发者的公钥才能解密）。

正式的数字证书的颁发者必须是**权威机构**，权威机构的证书（包含权威机构的公钥的数字证书，也就是权威机构的CA证书）是公开的，所以只要拿到权威机构的CA证书，我们就能对其签发数字证书的签名进行解密，解密成功证明该数字证书是权威机构所签发的。

于是，A与B之间的通信就变成这样：

1. A与B互相交换数字证书
2. A收到的证书为“某权威机构C签发给B的证书”
3. A拿到权威机构C的CA证书，对B证书上的签名（权威机构C的签名）进行解密，解密成功，证明这个证书是权威机构C签发给B的证书。如果A信任权威机构C，则也信任权威机构C签发给B的证书，再使用B证书中的公钥对通信内容进行加密发给B。
4. B收到A的证书也做类似处理。

**说明：**

自签名的证书是由开发者自己签发的证书。一般情况下，先制作根证书（CA证书，开发者持有对应的私钥），再用根证书的私钥签发子证书。当然，由于开发者持有自签名根证书的私钥，也可以直接使用自签名根证书作为通信过程中的数字证书。

## HTTPs认证过程

http (超文本传输协议)协议被用于在客户端和服务端之间传递信息，http协议以明文方式发送内容，**不提供任何方式的数据加密**，如果攻击者截取了客户端和服务端之间的传输报文，就可以直接读懂其中的信息，因此，HTTP协议不适合传输一些敏感信息。

为了解决http协议的这一缺陷，需要使用另一种协议： https（安全套接字层超文本传输协议），https在http的基础上加入了SSL/TLS协议，SSL/TLS依靠证书来验证双方的身份，验证通过后为客户端和服务端之间的通信进行加密。

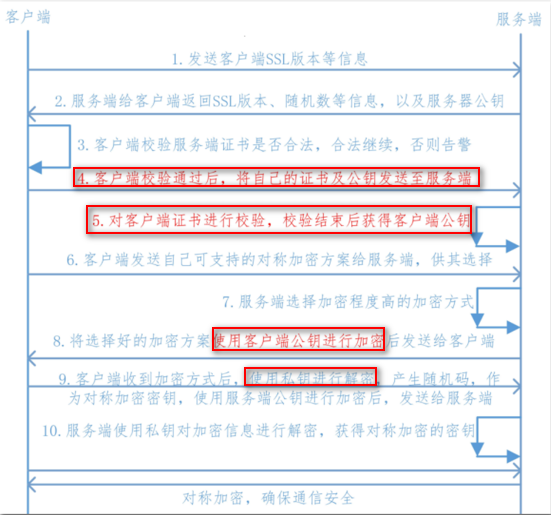
**https双向认证简单过程：**

1. 客户端与服务端交换证书
2. 先通过CA证书校验证书的真实性，再通过证书得到对端身份和公钥，**并校验对端身份**（校验对端请求中的主机地址与证书上的使用者是否一致）
3. 双方协商此次消息交互过程中使用的对称加密算法（协商过程使用对方证书中的公钥对随机加密密钥进行加密）
4. 后续消息中使用对称加密算法对消息内容进行加密，以保证双方通信内容的安全

**说明：**

1. 每次https双向认证都会进行对称加密算法的协商，使用不同的加密密钥，而不是直接使用对端公钥进行加密，这样做的好处是：对称加密算法效率高，加密解密快；每次加密密钥都是随机产生的，不会重复，不易被破解。
2. HTTPs单向认证与双向认证的主要区别在于，单向认证时，客户端对服务端身份进行校验，而服务端不对客户端身份进行校验。是否进行双向认证可在服务端进行配置。

下图就是**双向**认证的过程，去掉红色框中的步骤即为**单向**认证。



了解了以上知识，我们可以总结出：

1. 客户端用的证书叫客户端证书，服务端用的证书叫服务端证书，两个证书在本质上没有区别，检验证书的方法也是一样的。
2. 证书中有公钥信息，交换了证书相当于交换了公钥；双向认证时，每次HTTPs握手都会交换证书。但通信过程中的加密密钥却不是直接使用证书中的公钥，而是每次握手时都进行对称加密算法协商，生成随机加密密钥。
3. CA证书是证书签发者（往往是权威机构）的证书，一般不是握手过程中交换的证书（除非直接使用自签名的CA证书，因为是自签名的证书，开发者才证书对应有私钥）。CA证书是用于对证书签发者的签名进行校验的，校验通过就认为证书是CA签发的，证书可信。

# 应用对接平台

应用对接平台有两种情况：1、应用调用平台提供的鉴权、注册设备等接口，完成各种业务操作。 2、平台调用应用提供的消息推送接口，给应用推送消息。

## 应用调用平台接口

这种情况下，平台是服务端，应用是客户端。

应用要完成的工作有：

1. 把平台的CA证书加到信任列表里。这样，进行HTTPs握手时应用才能信任平台的证书
2. 使用平台签发的证书（包括公私钥）做为客户端证书。由于平台上有该证书签发者的CA证书，所以进行HTTPs握手时平台能信任应用的证书。
3. 测试环境下，没有使用域名访问，所以应用要设置不校验服务端的域名；商用环境下，需进行域名校验。

如果使用JAVA语言开发，可参考3.1应用调用平台接口（java样例）

## 平台调用应用接口

这种情况下，应用是服务端，平台是客户端。平台调用应用服务器的接口给应用推送消息（应用服务器要实现消息推送的接口，具体接口可查看平台API参考文档）。

应用要完成的工作有：

1. 把平台的CA证书加到信任列表里。这样，进行HTTPs握手时应用才能信任平台的证书。
2. 获取证书（使用自签名证书或者购买正式证书），并在服务端部署好证书，再将**CA证书**上传到平台，平台会把应用服务器的CA证书加到平台的证书信任列表里。
3. 测试环境下，没有使用域名访问，所以应用要设置不校验服务端的域名；商用环境下，需进行域名校验。

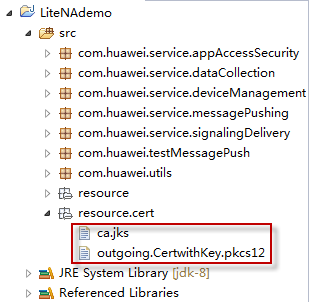
如果使用tomcat部署应用服务器，可参考3.2使用keytool制作自签名证书及tomcat证书部署，其他情况请参考3.3使用openssl制作自签名证书。

# 附录

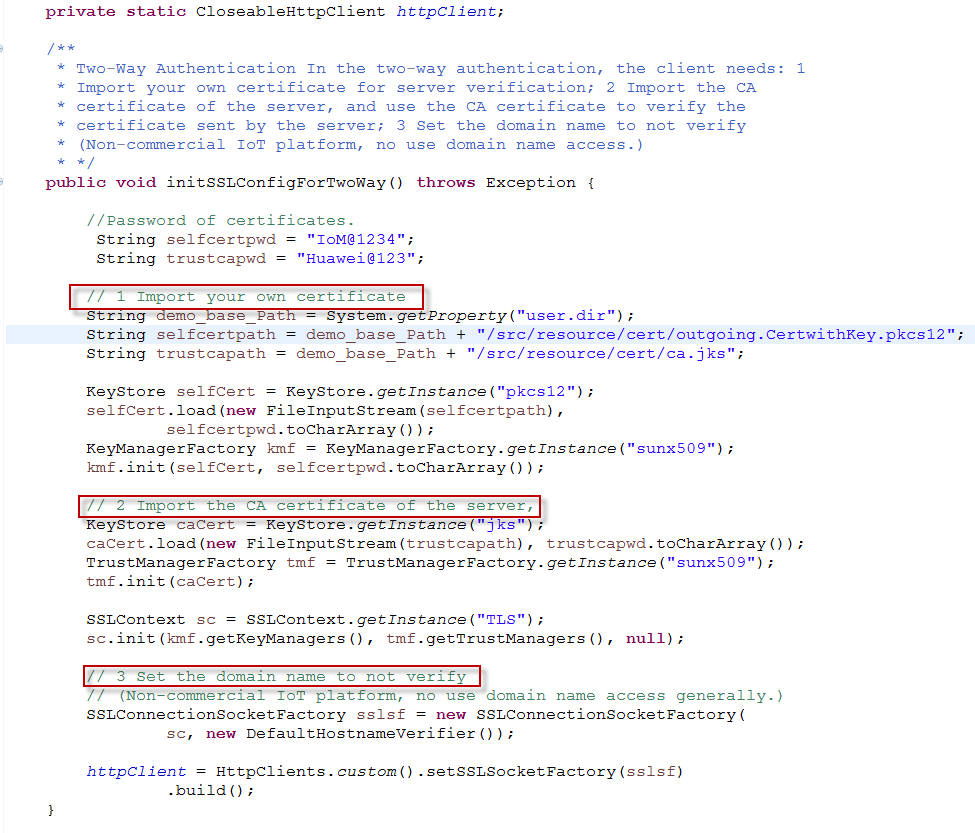
## 应用调用平台接口（java样例）

以一个简单的Java开发应用（基于rest接口）为例，NA端需根据自身开发的语言、工程等实际情况对证书作具体的加载和配置。

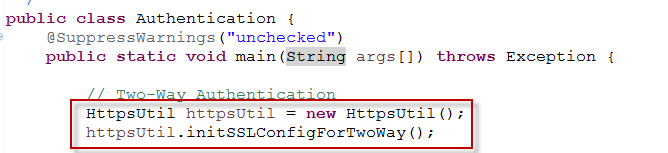
1.将IoT平台提供的ca.jks(CA证书)和outgoing.CertwithKey.pkcs12(客户端证书)拷贝到项目中



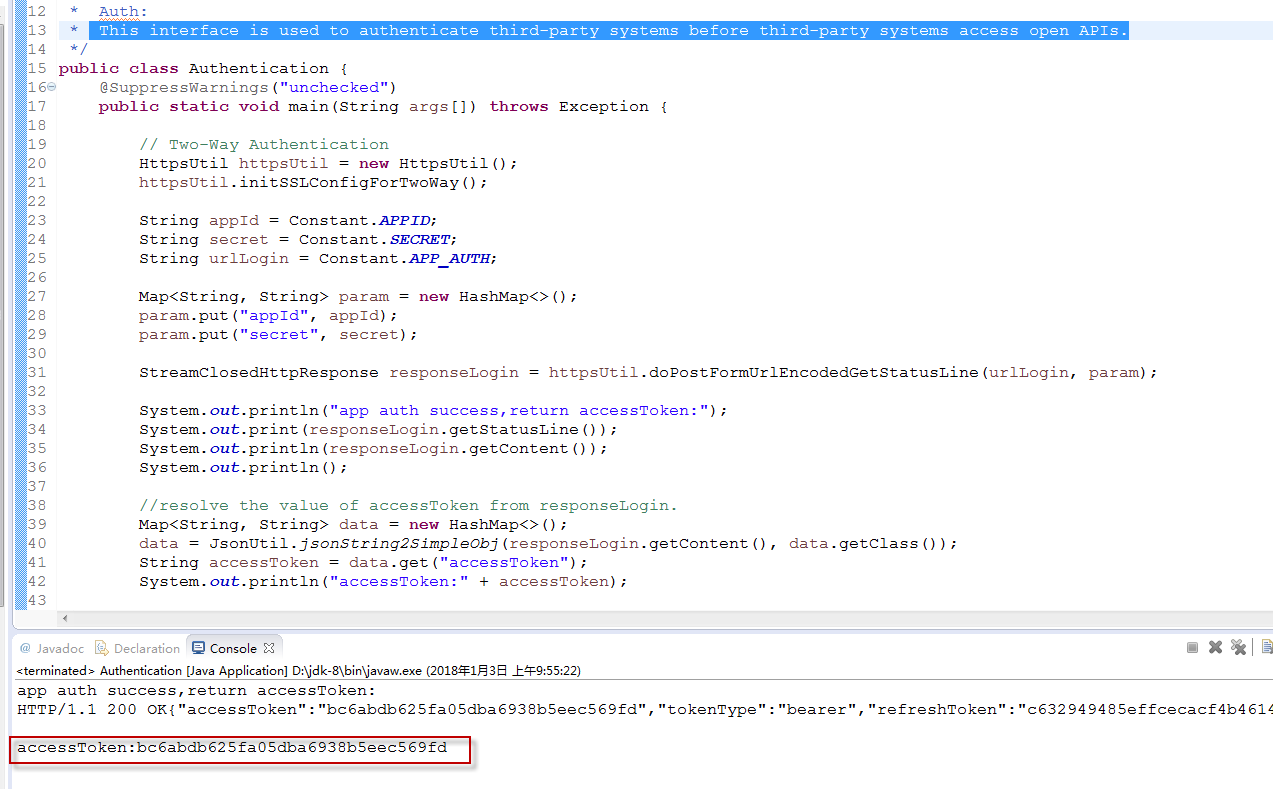
2.初始化双向验证的方法：①将客户端证书加入到NA的证书库中②将IoT的CA证书加入到NA信任列表。③测试环境下，没有使用域名访问，故设置域名不校验，商用环境下，需进行域名校验。如下图所示：

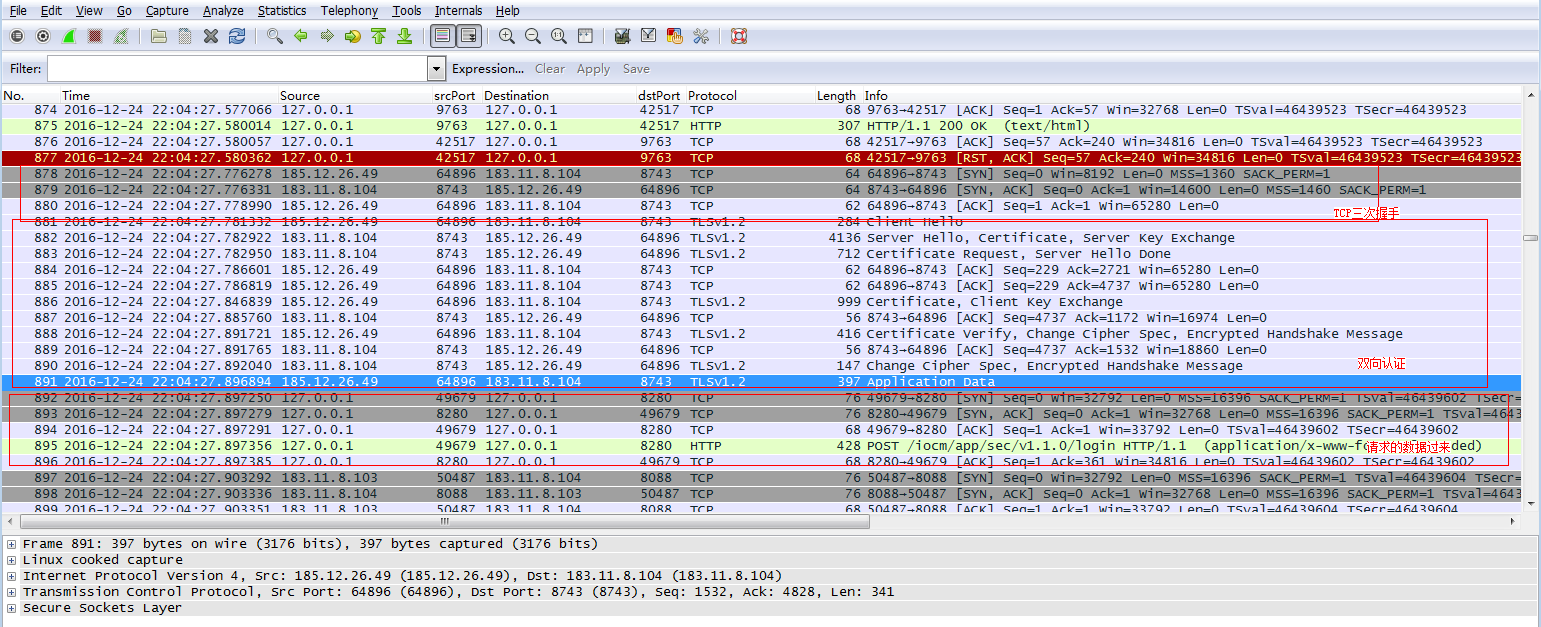


3.业务中调用双向验证初始化方法



4.调用鉴权接口测试建链是否成功，返回accessToken表示鉴权成功；利用wireshark抓包工具可以看出双方采用的是https协议；





## 使用keytool制作自签名证书及tomcat证书部署

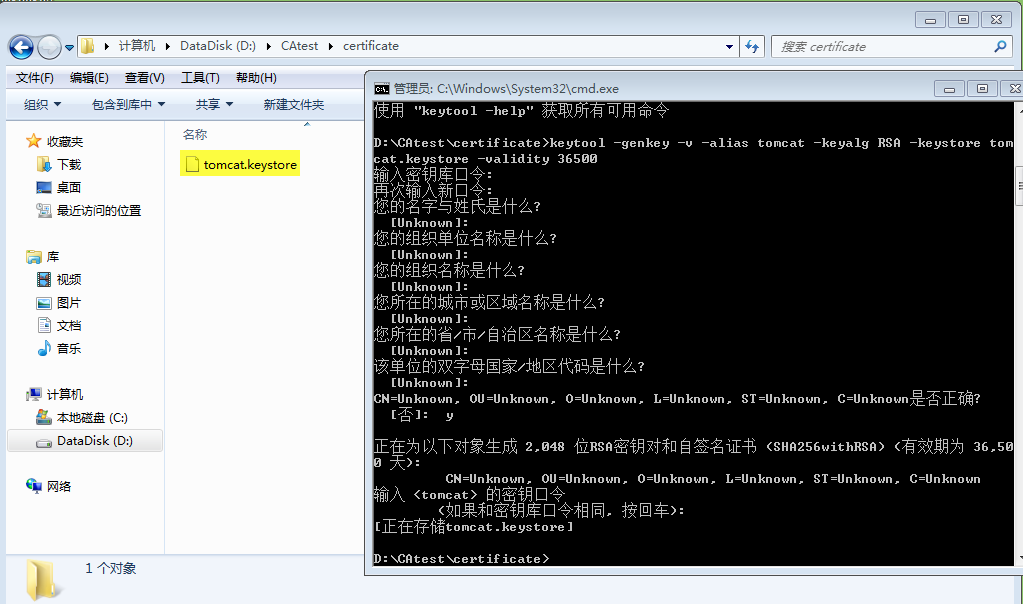
不同的服务器软件，SSL/TSL证书格式和配置方式不同。用户应根据实际的开发情况来配置SSL/TSL证书，此处以keytool制作证书在tomcat上部署为例进行简单介绍。（keytool命令可以在JDK安装路径即%JAVA\_HOME%的bin目录下执行，openssl需安装）

1. 为服务端生成证书。

使用keytool为tomcat生成证tomcat.keystore;

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

**注意：名称和姓氏要填写应用服务器的域名或IP**



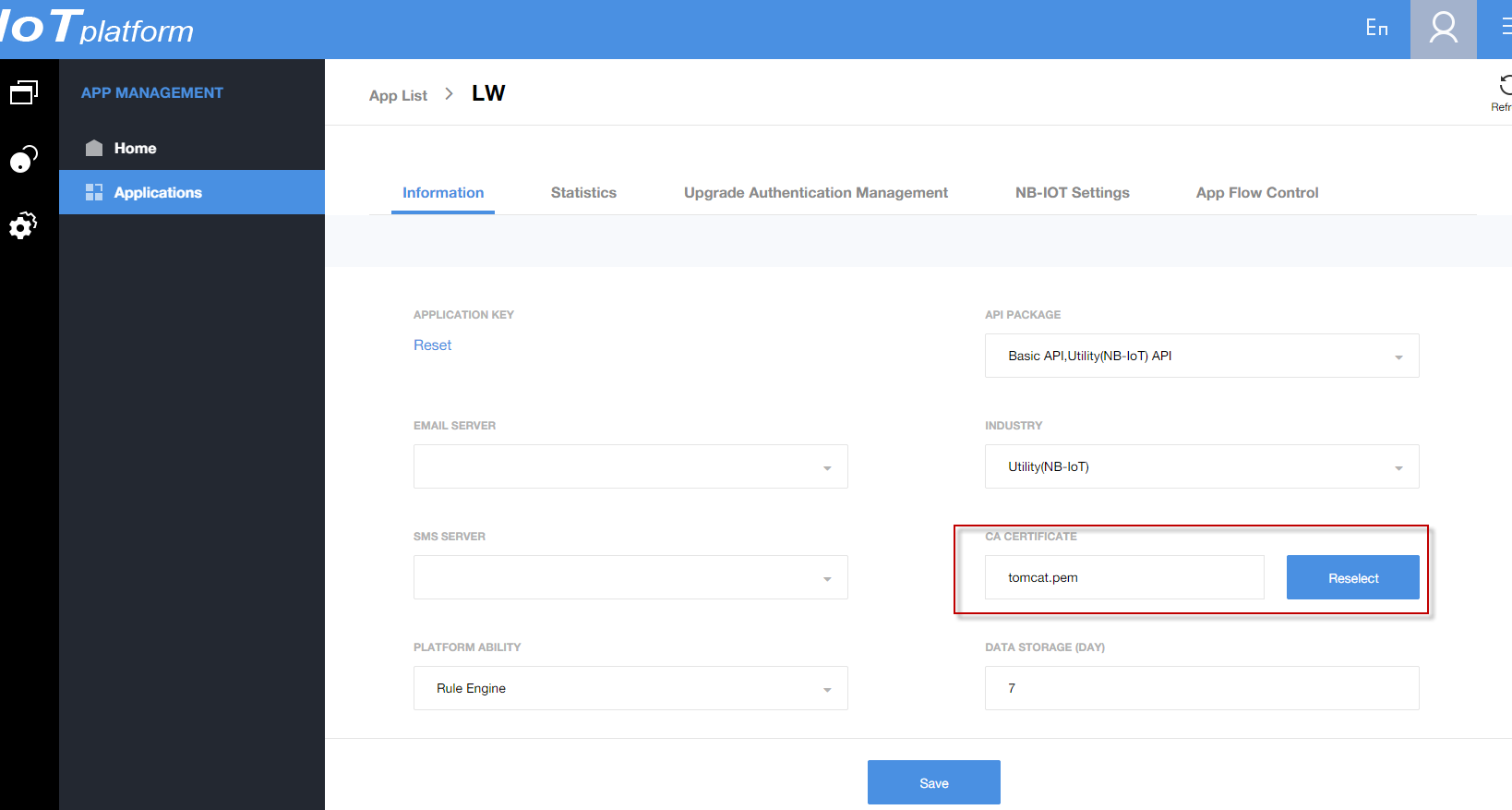
1. 让客户端（即平台）信任服务器CA证书
2. 导出根证书tomcat.cer

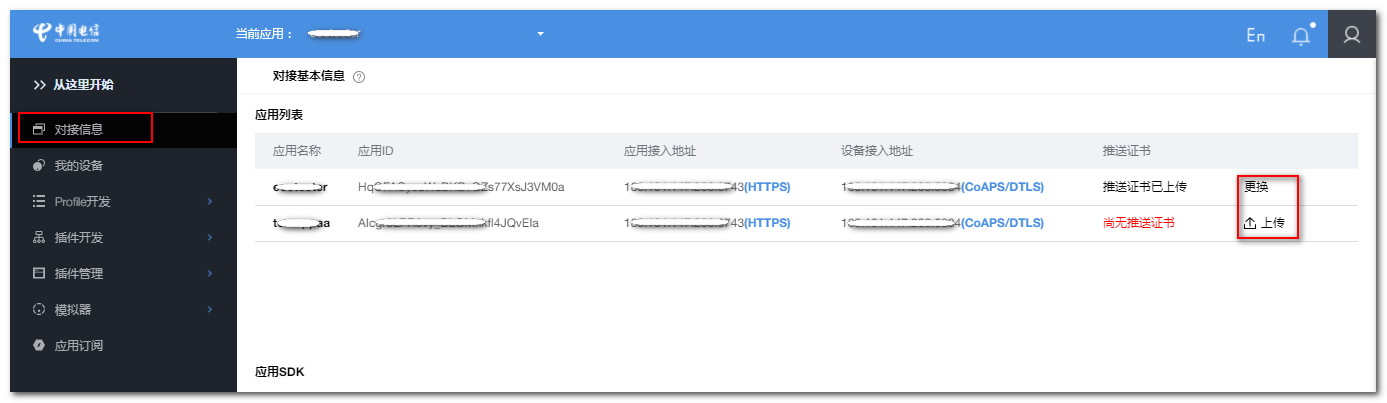
keytool -export -v -alias tomcat -file tomcat.cer -keystore tomcat.keystore

1. 使用openssl(需要安装openssl)命令把tomcat.cer文件转换成tomcat.pem文件

openssl x509 -inform der -in tomcat.cer -out tomcat.pem

1. 上传证书（使用SP Portal或开发者Portal，注意上传证书时选择对应的应用）：





1. 将IoT提供的CA证书导入tomcat的信任证书链中

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

1. 配置Tomcat服务器

属性说明：

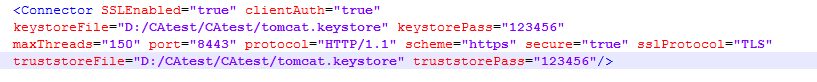
clientAuth：设置是否双向验证，默认为false，设置为true代表双向验证，测试时建议先调通单向认证，再调双向认证。双向认证才是安全的。

keystoreFile：服务器证书文件路径

keystorePass：服务器证书密码

truststoreFile：用来验证客户端证书的根证书，此例中已将平台的CA证书导入keystore文件。

truststorePass：根证书密码



## 使用openssl制作自签名证书

1. openssl genrsa -out d:\generasoft\hwcert\mycert\my.key 2048 //生成秘钥对，包含公钥和私钥。key文件的目录随意
2. openssl req -new -x509 -days 36500 -key d:\generasoft\hwcert\mycert\my.key -out d:\generasoft\hwcert\mycert\my.pem -subj "/C=CN/ST=Guangdong/L=Shenzhen/O=Huawei/OU=PersonalPC/CN=**127.0.0.1**" -config d:\generasoft\OpenSSL-Win32\bin\openssl.cfg //文件的目录随意，C=CN不要改，ST L O OU随意，**CN写你的订阅通知回调地址的域名或者IP**，用ip访问写ip，用域名访问写域名，openssl.cfg写你具体的cfg路径。在openssl安装目录可以找到
3. openssl pkcs12 -export -out d:\generasoft\hwcert\mycert\my.p12 -inkey d:\generasoft\hwcert\mycert\my.key -in d:\generasoft\hwcert\mycert\my.pem //生成p12文件格式，如果使用IIS，需生成p12格式

执行成功后，将获得私钥文件my.key，公钥证书my.pem，p12格式的公私钥证书my.p12。将my.pem作为CA证书上传到平台，使用my.p12作为IIS或者C#服务器的证书。

## 应用部署在云上如何导出CA证书（多级证书）

请参考<http://developer.huawei.com/ict/forum/forum.php?mod=viewthread&tid=47467>

## 证书使用异常处理

关于SSL/TLS证书都已加载和配置完成，但是NA端和IoT平台建链不成功，开发者可对以下几方面进行检查。

* + - 1. 检查本地与IoT平台网络链路

在本地电脑通过cmd窗口使用以下指令检查对端端口是否可达，如果出现黑色页面则表示网络链路是通的。

telnet ip port

* + - 1. 检查https接入时使用的TLS版本

对于TLS版本，IoT平台只支持TLS1.1和TLS1.2，因此第三方NA在进行对接时需要对TLS版本进行检查，避免由于TLS版本不对造成握手失败。

如果使用的是JDK1.8版本，则默认使用的是TLS1.2；如果使用的是JDK1.7版本，则默认使用的是TLS1.0。建议NA端开发者使用JDK1.8进行开发。

* + - 1. 利用keytool命令查看证书

查看单个证书：

keytool -printcert -v -file mydomain.crt

列出keystore存在的所有证书：

keytool -list -v -keystore keystore.jks

使用别名查看keystore特定条目：

keytool -list -v -keystore keystore.jks -alias mydomain

* + - 1. 其他异常情况，NA端开发人员可联系IoT平台的支撑人员进行解决。