

部分 IV. 国密配置

第一章 国密版配置

1.1. 生成国密证书

使用 GmSSL 工具包（在 25.5 相关资料章节下载）生成相关的国密证书，假设工具安装目录位\${GmSSL}。具体步骤如下：

1.1.1. GmSSL 工具配置

1. 把 GmSSL.tar.zip 文件拷贝的/opt 目录下，当前目录切换到/opt 目录，并通过命令 `tar -xzf GmSSL.tar.zip` 进行解压，会在/opt 目录下生成 GmSSL。aarch64 版本为 gmssl.zip，需要解压到 `opt/gmssl/`；
2. 设置下面 2 个环境变量，让 gmssl 命令可以直接运行，命令类似如下（路径需要根据实际更改）：

```
export PATH=$PATH:/opt/GmSSL/bin
export LD_LIBRARY_PATH=/opt/GmSSL/lib/
```

然后在终端输入 gmssl，能够出现输入提示符则表示成功，类似如下：

```
[root@weiyongsen-LogServer opt]# gmssl
GmSSL> █
```

3. openssl.cnf 文件修改

该文件在/opt/GmSSL/ssl 目录下。修改的内容包括：

1) [req]选项中添加或者修改： `default_md = sm3`

2) 修改[v3_req]选项内容为：

```
basicConstraints = CA:FALSE
```

```
keyUsage = nonRepudiation, digitalSignature
```

3) 添加[v3enc_req]选项，其内容：

```
basicConstraints = CA:FALSE
```

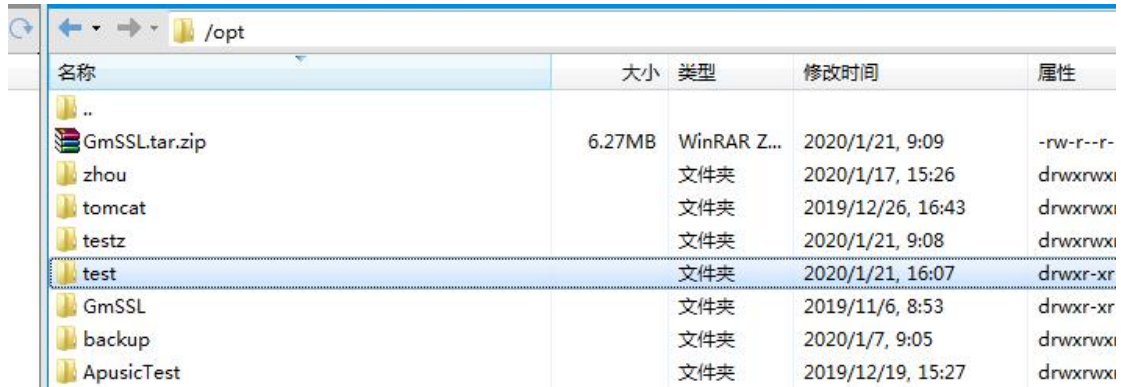
```
keyUsage = keyAgreement, keyEncipherment, dataEncipherment
```

1.1.2. 证书制作

1. 制作前准备

创建目录/opt/test，并在目录下创建一个名为"sm2Certs"的目录；

切换当前目录到/opt/test，并拷贝 openssl.cnf 文件文件到该目录下；



名称	大小	类型	修改时间	属性
..				
GmSSL.tar.zip	6.27MB	WinRAR Z...	2020/1/21, 9:09	-rw-r--r-
zhou		文件夹	2020/1/17, 15:26	drwxrwx
tomcat		文件夹	2019/12/26, 16:43	drwxrwx
testz		文件夹	2020/1/21, 9:08	drwxrwx
test		文件夹	2020/1/21, 16:07	drwxr-xr
GmSSL		文件夹	2019/11/6, 8:53	drwxr-xr
backup		文件夹	2020/1/7, 9:05	drwxrwx
ApusicTest		文件夹	2019/12/19, 15:27	drwxrwx

2. 生成 SM2 参数文件

执行命令：

```
gmssl eparam -name SM2 -out SM2.pem
```

```
[root@localhost test]# gmssl eparam -name SM2 -out SM2.pem
```

3. 生成 ca 证书

执行命令：

(1)

```
gmssl req -config ./openssl.cnf -nodes -subj "/C=CN/ST=GD/L=Shenzhen/O=Apusic LTD./OU=Development/CN=Test CA (SM2)" -keyout CA.key.pem -newkey ec:SM2.pem -new -out CA.req.pem
```

```
[root@localhost test]# gmssl req -config ./openssl.cnf -nodes -subj "/C=CN/ST=GD/L=Shenzhen/O=Apusic LTD./OU=Development/CN=Test CA (SM2)" -keyout CA.key.pem -newkey ec:SM2.pem -new -out CA.req.pem
Generating an EC private key
writing new private key to 'CA.key.pem'
-----
```

(2)

```
gmssl x509 -req -days 7300 -in CA.req.pem -extfile ./openssl.cnf -extensions v3_ca -signkey CA.key.pem -out CA.cert.pem
```

```
[root@localhost test]# gmssl x509 -req -days 7300 -in CA.req.pem -extfile ./openssl.cnf -extensions v3_ca -signkey CA.key.pem -out CA.cert.pem
Signature ok
subject=C = CN, ST = GD, L = Shenzhen, O = Apusic LTD., OU = Development, CN = Test CA (SM2)
Getting Private key
```

(3)

```
rm CA.req.pem
```

```
[root@localhost test]# rm CA.req.pem
rm: 是否删除普通文件 "CA.req.pem"? y
```

4. 生成 SSL 服务端 (或客户端) 签名证书

执行命令:

(1)

```
gmssl req -config ./openssl.cnf -nodes -subj "/C=CN/ST=GD/L=Shenzhen/O=Apusic LTD./OU=Development/CN=server sign (SM2)" -keyout sm2Certs/svrsig.key.pem -newkey ec:SM2.pem -new -out sm2Certs/svrsig.req.pem
```

```
[root@localhost test]# gmssl req -config ./openssl.cnf -nodes -subj "/C=CN/ST=GD/L=Shenzhen/O=Apusic LTD./OU=Development/CN=server sign (SM2)" -keyout sm2Certs/svrsig.key.pem -newkey ec:SM2.pem -new -out sm2Certs/svrsig.req.pem
Generating an EC private key
writing new private key to 'sm2Certs/svrsig.key.pem'
-----
```

(2)

```
gmssl x509 -req -days 365 -in sm2Certs/svrsig.req.pem -CA CA.cert.pem -CAkey CA.key.pem -extfile ./openssl.cnf -extensions v3_req -out sm2Certs/svrsig.cert.pem -CAcreateserial
```

```
[root@localhost test]# gmssl x509 -req -days 365 -in sm2Certs/svrsig.req.pem -CA CA.cert.pem -CAkey CA.key.pem -extfile ./openssl.cnf -extensions v3_req -out sm2Certs/svrsig.cert.pem -CAcreateserial
Signature ok
subject=C = CN, ST = GD, L = Shenzhen, O = Apusic LTD., OU = Development, CN = server sign (SM2)
Getting CA Private Key
```

(3)

```
rm -f sm2Certs/svrsig.req.pem
```

```
[root@localhost test]# rm -f sm2Certs/svrsig.req.pem
```

5. 生成 SSL 服务端 (或客户端) 加密证书

执行命令:

(1)

```
gmssl req -config ./openssl.cnf -nodes -subj "/C=CN/ST=GD/L=Shenzhen/O=Apusic LTD./OU=Development/CN=server enc (SM2)" -keyout sm2Certs/svrenc.key.pem -newkey ec:SM2.pem -new -out sm2Certs/svrenc.req.pem
```

```
[root@localhost test]# gmssl req -config ./openssl.cnf -nodes -subj "/C=CN/ST=GD/L=Shenzhen/O=Apusic LTD./OU=Development/CN=server enc (SM2)" -keyout sm2Certs/svrenc.key.pem -newkey ec:SM2.pem -new -out sm2Certs/svrenc.req.pem
Generating an EC private key
writing new private key to 'sm2Certs/svrenc.key.pem'
-----
```

(2)

```
gmssl x509 -req -days 365 -in sm2Certs/svrenc.req.pem -CA CA.cert.pem -CAkey CA.key.pem -extfile ./openssl.cnf -extensions v3enc_req -out sm2Certs/svrenc.cert.pem -CAcreateserial
```

```
[root@localhost test]# gmssl x509 -req -days 365 -in sm2Certs/svrenc.req.pem -CA CA.cert.pem -CAkey CA.key.pem -extfile ./openssl.cnf -extensions v3enc_req -out sm2Certs/svrenc.cert.pem -CAcreateserial
Signature ok
subject=C = CN, ST = GD, L = Shenzhen, O = Apusic LTD., OU = Development, CN = server sign (SM2)
Getting CA Private Key
```

(3)

```
rm -f sm2Certs/svrenc.req.pem
```

```
[root@localhost test]# rm -f sm2Certs/svrsig.req.pem
```

6. 生成 PKCS12 秘钥库

命令中的 JSSE_GMDIR 需要根据实际的 AAS 版本和 JDK 版本进行指定，如目录 `${APUSIC_HOME}/lib/endorsed/gm/JDK8u`。如果证书需要在其他 AAS 应用，需要注意 JDK 版本与制作时使用的 JDK 版本一致。制作时提示输入密码，正确输入即可。

注：AAS-V10.1SP4 之后的版本没有 JSSE_GMDI 对应的目录，需要使用 `kes.zip` 工具生成

执行命令：

```
gmssl pkcs12 -export -inkey sm2Certs/svrsig.key.pem -in sm2Certs/svrsig.cert.pem -CAfile CA.cert.pem -chain -out sm2Certs/svrsig.p12 -name "svrsig"
```

```
[root@localhost test]# gmssl pkcs12 -export -inkey sm2Certs/svrsig.key.pem -in sm2Certs/svrsig.cert.pem -CAfile CA.cert.pem -chain -out sm2Certs/svrsig.p12 -name "svrsig"
Enter Export Password:
Verifying - Enter Export Password:
```

```
gmssl pkcs12 -export -inkey sm2Certs/svrenc.key.pem -in sm2Certs/svrenc.cert.pem -CAfile CA.cert.pem -chain -out sm2Certs/svrenc.p12 -name "svrenc"
```

```
[root@localhost test]# gmssl pkcs12 -export -inkey sm2Certs/svrenc.key.pem -in sm2Certs/svrenc.cert.pem -CAfile CA.cert.pem -chain -out sm2Certs/svrenc.p12 -name "svrenc"
Enter Export Password:
Verifying - Enter Export Password:
```

将分别生成的秘钥库“合并”：

```
cp sm2Certs/svrsig.p12 sm2Certs/keystore.p12
```

```
java -Djava.endorsed.dirs=${JSSE_GMDIR} sun.security.tools.keytool.Main -importkeystore -srckeystore sm2Certs/svrenc.p12 -srcstoretype PKCS12 -destkeystore sm2Certs/keystore.p12 -deststoretype PKCS12
```

```
[root@localhost test]# cp sm2Certs/svrsig.p12 sm2Certs/keystore.p12
.p12 -deststoretype PKCS12[root@localhost test]# java -Djava.endorsed.dirs=/opt/testz/01031/ApusicAS/aas/lib/entry.tools.keytool.Main -importkeystore -srckeystore sm2Certs/svrenc.p12 -srcstoretype PKCS12 -destkeystore sm2Certs/keystore.p12 -deststoretype PKCS12
正在将密钥库 sm2Certs/svrenc.p12 导入到 sm2Certs/keystore.p12...
输入目标密钥库口令:
输入源密钥库口令:
已成功导入别名 svrenc 的条目。
已完成导入命令: 1 个条目成功导入, 0 个条目失败或取消
```

7. 生成信任库（可选）

生成 `jks` 格式的信任库,JSSE_GMDIR 需要根据实际的 AAS 版本和 JDK 版本进行指定，如目录 `${APUSIC_HOME}/lib/endorsed/gm/JDK8u` 或 `${APUSIC_HOME}/lib/endorsed/gm/JDK8`，制作时提示输入密码，正确输入即可。

注：AAS-V10.1SP4 之后的版本没有 JSSE_GMDI 对应的目录，需要使用 `kes.zip` 工具生成

执行命令：

```
java -Djava.endorsed.dirs=${JSSE_GMDIR} sun.security.tools.keytool.Main -importcert -trustcacerts -alias root -file CA.cert.pem -keystore truststore.jks -storetype jks
```

```

[root@localhost test]# java -Djava.endorsed.dirs=/opt/testz/01031/ApusicAS/aas/lib/endorsed/gm/JDK8u_sun.securi
ty.tools.Keytool.Main -importcert -trustcacerts -alias root -file CA.cert.pem -keystore truststore.jks -store
type jks
输入密钥库口令:
再次输入新口令:
所有者: CN=Test CA (SM2), OU=Development, O=Apusic LTD., L=Shenzhen, ST=GD, C=CN
发布者: CN=Test CA (SM2), OU=Development, O=Apusic LTD., L=Shenzhen, ST=GD, C=CN
序列号: a26e3c3e1523fcbc
有效期为 Tue Jan 21 09:17:33 CST 2020 至 Mon Jan 16 09:17:33 CST 2040
证书指纹:
    MD5: 4F:45:00:AA:2F:DA:94:E8:DE:66:FC:06:3F:DE:FC:CC
    SHA1: FB:E3:F9:63:AF:F7:3E:F9:D9:15:91:CF:8D:69:AA:2A:80:A3:C1:05
    SHA256: 51:4F:01:D9:E5:37:E3:DD:34:58:89:2F:ED:A9:F4:7B:E0:79:ED:58:19:61:84:89:D7:B2:EB:53:FD:F1:4D:5
5
签名算法名称: 1.2.156.10197.1.501
主体公共密钥算法: 256 位 EC 密钥
版本: 3

扩展:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: B5 22 1E CB F1 70 56 D0 17 A7 06 FA 04 94 15 7D ..."pV.....
0010: 09 DD E4 02 .....
]
]

#2: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
CA:true
PathLen:2147483647
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: B5 22 1E CB F1 70 56 D0 17 A7 06 FA 04 94 15 7D ..."pV.....
0010: 09 DD E4 02 .....
]
]

是否信任此证书? [否]: y
证书已添加到密钥库中

```

1.2. 证书使用

经过上面步骤后，生成了CA证书文件CA.cert.pem，信任库文件truststore.jks，服务器证书文件keystore.p12(在sm2Certs目录)。

1.开启国密支持。domain.xml中java-config的

<jvm-options>-Dcom.apusic.security.ssl.EnableGMTLS=false</jvm-options>中的false改为true。

<jvm-options>-Dcom.apusic.security.ssl.EnableGMTLS=true</jvm-options>

```

<jvm-options>
[1.8.0_251]-Xbootclasspath/a:${com.apusic.aas.installRoot}/lib/grizzly-npn-api.jar
</jvm-options>
<jvm-options>-Dcom.apusic.security.ssl.EnableGMTLS=true</jvm-options>
</java-config>
<network-config>
<protocols>
<protocol name="http-listener-1">
<http max-connections="250" default-virtual-server="server">
<file-cache/></file-cache>
</http>
</protocol>
<protocol name="http-listener-2" security-enabled="true">

```

2.配置国密证书。

1) 复制服务器文件keystore.p12复制到\${APUSIC_DOMAIN}/config/

名称	大小	类型	修改时间	属性
..				
wss-server-config-2.0.xml	6KB	XML 文件	2020/1/3, 9:42	-rw-r-----
wss-server-config-1.0.xml	5KB	XML 文件	2020/1/3, 9:42	-rw-r-----
server.policy	5KB	POLICY ...	2020/1/3, 9:42	-rw-r-----
restrict.server.policy	1021 Bytes	POLICY ...	2020/1/3, 9:42	-rw-r-----
pid.prev	5 Bytes	PREV 文件	2020/1/21, 16:58	-rw-r--r--
login.conf	606 Bytes	CONF 文件	2020/1/3, 9:42	-rwxr-x---
logging.properties	4KB	PROPERT...	2020/1/17, 17:20	-rw-r-----
lockfile	0 Bytes	文件	2020/1/17, 16:23	-rw-r--r--
local-password	41 Bytes	文件	2020/1/21, 16:58	-rw-----
keystore.p12	3KB	Personal ...	2020/1/21, 11:01	-rw-r--r--
keystore.jks	4KB	JKS 文件	2020/1/3, 9:42	-rw-----
keyfile	748 Bytes	文件	2020/1/3, 9:42	-rw-----
javaee.server.policy	2KB	POLICY ...	2020/1/3, 9:42	-rw-r-----
history-password.json	236 Bytes	JSON 文件	2020/1/21, 15:54	-rw-r-----
hidden.properties	747 Bytes	PROPERT...	2020/1/3, 9:42	-rw-r-----

2) domain.xml 文件中 network-config/protocols/protocol 子元素。在所需要的端口对应的 protocol 中, 设置 security-enabled 属性为"true", 添加 ssl 子元素, 并设置 sm11-enabled 属性为"true", 并添加对应的 keystore, truststore 相关属性。如下所示:

```

<protocol name="http-listener-2" security-enabled="true">
  <http max-connections="250" default-virtual-server="server">
    <file-cache></file-cache>
  </http>
  <ssl classname="com.sun.enterprise.security.ssl.ApusicSSLImpl" sm11-enabled="true"
    key-store-type="PKCS12"
    key-store="{com.apusic.aas.instanceRoot}/config/keystore.p12"
    key-store-password="123456"
    client-auth="want"></ssl>
</protocol>

```

属性说明:

- security-enabled: 为 true 时, 开启 HTTPS 模式
- sm11-enabled: 为 true 时, 开启国密 TLS
- key-store: 密钥库路径
- key-store-type: 密钥库类型, 一般为 (JKS, PKCS12)
- key-store-password: 密钥库密码
- trust-store: 信任库路径
- trust-store-type: 信任库类型, 一般为 (JKS)
- trust-store-password: 信任库密码
- client-auth-enabled: 为 true 时, 开启客户端认证。

```
<network-config>
<protocols>
<protocol name="http-listener-1">
<http default-virtual-server="server">
<file-cache></file-cache>
</http>
</protocol>
<protocol name="http-listener-2" security-enabled="true">
<http max-connections="250" default-virtual-server="server">
<file-cache></file-cache>
</http>
<ssl classname="com.sun.enterprise.security.ssl.ApusicSSLImpl" client-auth="want" key-store-type="PKCS12" smil-enabled="true" key-store=
"${com.apusic.aas.instanceRoot}/config/keystore.p12" key-store-password="123456"></ssl>
</protocol>
<protocol name="admin-listener">
<http max-connections="250" default-virtual-server="__asadmin">
<file-cache></file-cache>
</http>
</protocol>
<protocol name="sec-admin-listener" security-enabled="true">
<http encoded-slash-enabled="true" default-virtual-server="asadmin">
```

3.启动 AAS-V10

1.3.360 浏览器国密版安装

1.3.1. 浏览器安装

从 360 网站下载国密浏览器(<http://browser.360.cn/se/ver/gmzb.html>)进行安装，测试使用的是 V10 版本

1.3.1.1. 选择支持国密

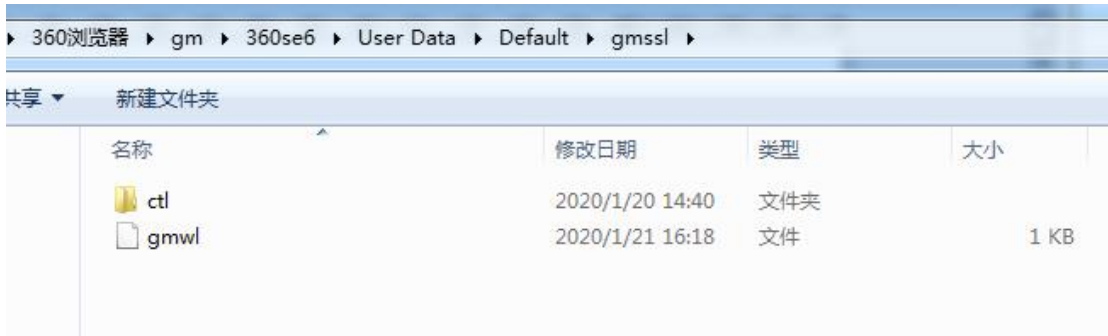
打开 360 浏览器，点击浏览器的右上角的“三”按钮，再点击“设置”选项，如下图：



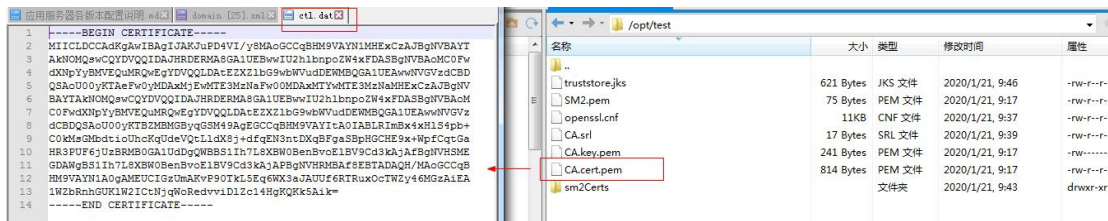
1.3.1.2. 根证书内容处理

根证书可按照以下方式设置，也可以通过浏览器导入。

- (1) 在 360 浏览器国密版的安装目录 {ROOT\360 浏览器\gm\360se6\User Data\Default\gmssl}新建文件夹 ctl



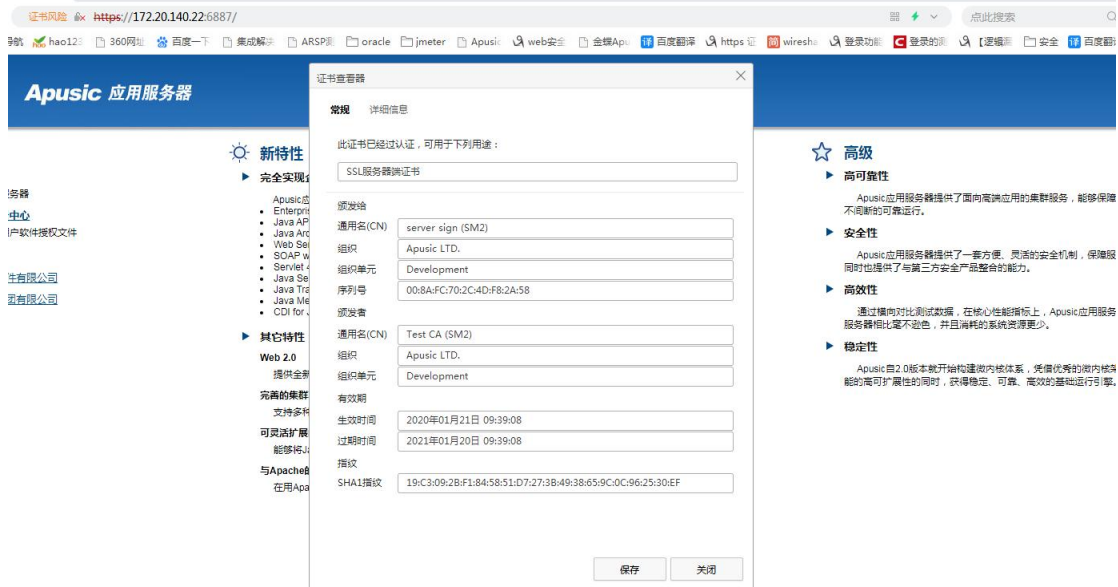
- (2) 在 ctl 目录下创建 ctl.dat 文件，并将根证书内容拷入该文件



- (3) 重启浏览器

1.4. 访问地址

在 360 浏览器国密版输入地址：https://ip:6887.



1.5. 相关资料

X86 操作系统下使用 GmSSL.tar.zip, 解压目录为/opt/GmSSL; ARM 操作系统下使用 gmssl.zip, 解压目录为/opt/gmssl



1.6. 证书导入

需要在 AAS 的证书中导入指定 SSL 证书, 可参考下列 specj.keystore.jks 示例。

导入前需要准备有 specj.keystore.jks 文件和安装有完整的 JDK。

第一步: 修改 specj.keystore.jks 密钥库的密码和密钥对的密码:

将 specj.keystore.jks 复制到 JDK 的 bin 目录下, 可以先查看证书里有多少个密钥库条目 (keytool -list -keystore specj.keystore.jks), 根据实际修改 keypasswd, 如示例有两个密钥库条目, 执行以下命令

(1)修改 storepasswd: keytool -storepasswd -keystore specj.keystore.jks(需修改口令的 keystore) -storepass specjpass(原始密码) -new changeit(新密码)

(2)修改密钥库条目 specjidentity 的 keypasswd: keytool -keypasswd -alias specjidentity(需要修改密码的别名) -keypass specjpass(原始密码) -new changeit(别名的新密码) -keystore specj.keystore.jks -storepass changeit

(3)修改密钥库条目 specjidentity2 的 keypasswd: keytool -keypasswd -alias specjidentity2(需要修改密码的别名) -keypass specjpass(原始密码) -new changeit(别名的新密码) -keystore specj.keystore.jks -storepass changeit

第二步: 将密码统一之后, 导入到中间件密钥库中(先把 AAS 的 keystore.jks 也放入 JDK 的 bin 目录下):

(1)Keytool -importkeystore -srckeystore specj.keystore.jks -destkeystore keystore.jks -srcstorepass changeit -deststorepass changeit

第三步: 将第二步执行后的 keystore.jks 拷贝至 AAS 安装目录对应位置, 如 \${DOMAIN_HOME}/mydomain/config

第四步: 启动 AAS, 登录管控平台, 将对应实例的证书昵称更改, 如更改 server 的 http-listener-2 的证书, 需要进入【配置】-【server-config】-【HTTP 服务】-【HTTP 监听程序】-【http-listener-2】中的“SSL”, 将“证书昵称”改为 specjidentity, 即第一步设置的别名[specjidentity(需要修改密码的别名)], 保存。之后重启 AAS。再次访问 https://ip:6887 时使用的证书是 specj.keystore.jks