Procédure de signature de .jar :

Demande d'une clé de signature

La demande s'effectue auprès de Renater et c'est l'établissement hébergeur qui en fera la demande. Quand il la recevra, elle vous sera transmise. Lors de la demande, il faut fournir une personne référente dont les coordonnées seront dans la clé. Cela n'a pas d'incidence sur la procédure ensuite

A réception de la clé :

La clé est livrée en fichier .tar qui comprend 3 fichiers :
2 fichiers ***.pem*** : *cert-xxx**.pem*** et *chain-xxxx**.pem***
et un fichier ***.key*** : ***.**code signing-xxx.key**.***

Génération de la clé de signature :

Les opérations suivantes ne sont à faire qu'une seule fois, à la réception de la clé, avant utilisation.
Les fichiers générés sont valables 3 ans.

Vérifier si l'accès aux commandes de java est correct :

Vérifier l'accès aux commandes : *keytool, jarsigner*
Et modifier éventuellement la variable PATH pour un accès plus aisé. Ces utilitaires sont dans les binaires de Java.

Vérifier si la clé est bonne :

$>openssl x509 -noout -modulus -in `cert-xxx.pem` ***| openssl md5 ;openssl rsa -noout -modulus -in*** code-signing-xxx.key ***| openssl md5***
=> affiche les 2 cles. Verifier si elles sont identiques => ok.
ou

$>openssl x509 -noout -modulus -in `cert-xxx.pem` ***| openssl md5 ;openssl rsa -noout -modulus -in*** code-signing-xxx.key ***| openssl md5)| uniq***
=> affiche une seule cle si ok sinon affiche les 2 (si differentes)

Procédure pour générer la clé :

==A faire dans le répertoire où sont les fichiers clés : ==

$> openssl pkcs12 -export -in `cert-xxx.pem` ***-out*** `cert-xxx.p12` ***-name "Certificat" -inkey*** code-signing-xxx.key ***-certfile*** chain-xxx.pem

Cela va générer un fichier **.p12** à l'aide des fichiers **.chain** et **.key**
Un mot de passe est demandé au moment de cette génération.
A noter puisqu'il sera demandé plus tard, et lors de la signature des jar.
C'est ici qu'est défini aussi le nom de la clé qui sera utilisée plus tard pour la signature : **« Certificat »**

Le fichier Keystore (<fichier>.jks):

Générer un keystore :

Le fichier**.jks** ne sera pas employé tel quel – cette procédure permet juste de générer un contenant.

$> keytool -genkey -alias Test -keystore <fichier>.jks

Il est demandé de donner un mot de passe pour le keystore.
Les informations demandées à la suite ne sont pas essentielles puisque le fichier généré ne sera pas employé en tant que tel.
Taper ***<Enter>*** jusqu'à la demande de confirmation

Est-ce CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown ?
[non]:

Taper <oui> (ou <Yes> suivant ce qui est demandé) pour confirmer.

A chaque instruction suivante concernant le keystore, il sera demandé le pwd du fichier keystore (.jks) généré.

Vider le keystore :

$> keytool -delete -alias Test -keystore <fichier>.jks
La commande sera execute après avoir entré le pwd du keystore.

Vérifier si le keystore est vide :

$> keytool -list -v -keystore <fichier>.jks
Pwd.

Remplir le keystore avec les bonnes informations :

A ce moment-là, re-remplir le fichier keystore avec les informations officielles reçues ainsi que la clé.

$> keytool -v -importkeystore -srckeystore cert-<xxx>.p12 ***-srcstoretype PKCS12 -destkeystore*** <fichier>.jks ***-deststoretype JKS***
Pwd keystore.
Pwd clé (celle générée plus haut).

Vérifier le contenu du keystore :

$> keytool -list -v -keystore <fichier>.jks
Pwd keystore.

<u>*Rappel*</u> : Tout ceci vient de décrire les étapes préliminaires avant la procédure de signature proprement dite. La clé a une durée de vie de 3 ans.
Cela n'est à faire qu'une seule fois, à la réception des informations de signature.

NE PAS OUBLIER LES PASSWORDs, keystore et clé. Sinon, il est nécessaire de **régénérer**.

Signature du code

$> jarsigner -keystore <fichier>.jks <fichier>.jar ***Certificat***

Demande le mot de passe du keystore, puis celui de la clé et ensuite, génération du jar signé.

Pour qu’une application soit identifiée comme sécurisée, il faut signer tous les .jar utilisés, même ceux des librairies utilisées.