Signing Java ME Applications

Contents

1. WHAT IS THE PURPOSE OF THIS DOCUMENT?

This document provides key information about application signing in Java ME. It starts from the basics so should be useful for anyone who is not familiar with Java ME application signing.

2. WHAT ARE THE BASICS?

2.1 What is application signing?

Application signing means that an application is signed with a private key. For each private key, there is a corresponding public key, which is delivered jointly with the application in the form of a digital certificate.[1]

When a signed application file is installed on a device, the application installer verifies that the certificate in the application was created by a one of the certificate authorities embedded in the device.

A certificate authority (CA) is an entity which issues digital certificates. CAs include device manufacturers, operators, testing and signing programs or professional companies offering certificate authority services. Java Verified is a Certification Authority and its certificate is the "Geotrust CA for UTI".

2.2 What indicates that an application has been signed?

When a MIDlet[2] is signed, two additional fields appear in the JAD file:

- MIDlet-Certificate-1-1

- MIDlet-Jar-RSA-SHA1

The MIDlet-Certificate field has the certificate used to sign the MIDlet. In some cases there are multiple MIDlet-Certificate fields, which is normal, so do not worry. Please note that all the fields which are in the JAD file are there for a good reason and should not be removed.

Note: The MIDlet-Jar-RSA-SHA1 field contains the checksum that was calculated from the JAR file and encrypted with the private key. So, if the JAR is

---

[1] In practice, this public key or certificate works like the family seal that was used to confirm the origin of letters in the last century. The receiver could verify the identity of the sender before opening the letter by examining the seal. Of course, the stamp used to introduce the seal was never shared with anyone.
[2] A MIDlet is a Java application framework for the Mobile Information Device Profile (MIDP) that is typically implemented on a Java ME -enabled device. MIDlets are applications, such as games.

changed, the calculated checksum will be incorrect and then the signature will no longer be valid.

2.3 Where does the signature take the application?

2.3.1 Domains

Certificate authorities (CAs) provide their certificates to the device manufacturers and these certificates are installed at the time of manufacture to a specific protection domain.

Note: In the case of certificates that are used with Java ME applications, it should not be possible to add these after device manufacture.

The security domain depends on the particular CA:

- Identified 3rd party protection domain – signed by or for a party which is known (formerly known as the Trusted third party domain)

- Operator domain – signed by an operator or a carrier

- Manufacturer domain – signed by a device manufacturer

Unsigned applications will be allocated to the unidentified third party protection domain (formerly known as the Untrusted third party domain)

When an application is signed, it will receive the privileges of the protection domain where the corresponding CA certificate is defined in the device. If the device does not have a corresponding certificate, the application will not install.

2.3.2 Accessing different domains

Access to a domain is in the hands of the party who acts as the certificate authority (CA) and thus application signing can carry liabilities for the CA. For example, if an operator signs an application to the operator domain, that operator can be held responsible for the application. This is why access to domains is tightly controlled.

2.3.3 Why do domains matter?

Domains matter because they determine the following parameters:

- The level of access the application has to certain device features:

- For example an unsigned MIDlet is not allowed to open datagram connections on ports 9200, 9201 or 9203 according to the MIDP 2.1 specification.

- The kinds of pop ups the user will have to deal with when using the application:

  - According to the MIDP 2.0 specification, an unsigned application with the default security settings must ask for a permission to make a network connection every time the application opens the connection.

- The options the user has to change the pop up settings:

  - An unsigned application can set "Always allowed" to local connections but not to any other feature according to the MIDP 2.0 specification.

Here are the different options granted to applications, which can be changed in the application access settings in the device:

- Not allowed - "No, the application cannot do that!"

- Ask every time / one shot - "Every time the application wants to use a feature the user is prompted."

- Ask first time / session - "When I run the application, the first time the feature is used the user is prompted."

- Always allowed / blanket - "No questions asked"

The place where the settings can be changed varies between devices.

2.3.4 Which pop up options are given to which features?

Generally it is very difficult to say which application features will be granted which options. This is mainly due to the different security implementations in devices. Nonetheless, the rules of thumb are:

- Application signed to the "Identified third party protection domain" has better options for limiting the pop ups than an application signed to the "Unidentified third party protection domain"

- Application signed to the "Operator domain" or the "Manufacturer domain" do not have any pop-ups.

3. HOW DO I USE SUN JAVA^{TM} WIRELESS TOOLKIT (WTK) FOR APPLICATION SIGNING?

To sign a Java ME application, an application such as Sun Java Wireless Toolkit for CLDC Version 2.5.2 (WTK) must be used.

Note: Usually Java ME applications signed in a person's own personal computer are signed with a Publisher ID or with a certificate obtained from a certificate authority (CA).

3.1 Importing the keys to WTK

The WTK needs to have access to the private key. The key therefore needs to be imported from a .pfx or .p12 formatted package to a keystore, which the WTK can use to load the key to sign applications.

The key tool application is used to do the import by copying the private key from the .PFX file to a specific keystore file. To do this, follow these steps:

- Open the command prompt and run the keytool.exe application.

  Note: You will need to define a password here. Please note this down, as you will need it again!

- Use the following command to copy the Publisher ID to a designated keystore file:

  Keytool –importkeystore –srckeystore "The_Name_Of_The_PFX_file.PFX" –destkeystore Name_For_The_Key_store.jks –srcstoretype PKCS12 – deststoretype JKS

3.2 Signing with WTK

The WTK has a utility called "Sign MIDlet Suite". In order to use this for signing applications, you will need to load the key into the keystore. To do this go to the "file" menu, select "Load keystore" and then "from file".

Note: When loading the keystore the keystore password is required.

3.3 What about NetBeans?

The keystore file will need to be imported to NetBeans by following these steps:

- Launch NetBeans. In the "properties" window, select the "Signing" category.

- Use the keystore manager to add a new keystore from a file (common file formats to import are ".jks, .ks, .keystore, .p12 and .pkcs12")
- After the keystore is imported it must be unlocked. When this is done check the "Sign Distribution" box, keystore and alias in the properties -> signing menu

When selecting the "Sign Distribution" the application will be signed at build time.

4. WHAT CAN GO WRONG?

To ensure that applications are signed correctly such that the application installs and works on a particular device:

- The "MIDlet-" attributes in the JAD and JAR manifest files must match.

- The correct permissions must be defined

- The correct certificate authority (CA) certificate must be present in the device

- The signature must be valid – the date and the time of the device must be correct

4.1 MIDlet attributes

- "MIDlet-" fields in JAD file

    =

- "MIDlet-" fields in JAR manifest file

If the equation is not true, then the application installation fails. There are only two exceptions: MIDlet-Jar-Size & MIDlet-Jar-URL attributes.

4. 2 Permissions

Permissions are used to protect Application Programming Interfaces (APIs) that are sensitive and require authorization. If permissions are not declared, then the application will not be able to implement functions that require permissions. Thus, permissions need to be defined to make the application work properly.

Permissions are defined in the MIDlet-Permissions attribute. For example:

- MIDlet-Permissions: javax.microedition.io.Connector.http

4.3 The device and the certificate

The device needs to have a certificate from a certificate authority (CA) to make the installation possible. It can be difficult to check the existence of that certificate as the certificate name may vary from device to device and the list of certificates is located in different places in different devices.

If the certificate is not there, but the application is signed, you can follow these steps to get the application installed:

1. Remove the following fields from the JAD:

- MIDlet-Certificate- fields

- MIDlet-Jar-RSA-SHA1 field

Note: Before removing the fields, check that the signature in the JAD file and the root certificate in the device are valid from the perspective of the device. The easiest way to do this is to confirm that the date and the time of the device are correct.

4.4. The validity period

To check the validity period of a root certificate in the device, locate the root certificate and the detailed view for the certificate should give information on the period when the certificate is valid.

To check the validity of the certificate in the JAD file, follow these steps:

- Open the JAD file with a text editor – notepad is just fine.

- Copy the contents of the "MIDlet-Certificate-1-1:" field (text starting from the colon and ending to the carriage return) and paste it to an empty text file.

- Save the text file

- Change the file extension to .cer

- Open the file, the validity is presented in the "Valid from" field.

5. GLOSSARY

- Java ME Certificate Authority (CA) certificate or Root CA

  o Public key which was self-signed and then installed to the device by the device manufacturer at the time of manufacture. The device manufacturer will have also defined the protection domain and the privileges in each domain.

- Signing an application installation file

  o A checksum is calculated from the JAR file and encrypted with the private key. That information is placed in the MIDlet-Jar-RSA-SHA1 field in the JAD file. The corresponding public key in the form of a certificate is then placed in the MIDlet-Certificate field. If the certificate included intermediate CAs, these are placed in the MIDlet-Certificate-1-2, MIDlet-Certificate-1-3, etc fields. If the application has multiple signatures these are marked as: MIDlet-Certificate-2-1, MIDlet-Certificate-3-1.

- Checksum

  o A figure which was calculated from (in this case) a file. The calculation was done using a specific function. The checksum is used to compare that (in this case) a file has not been altered. The most common function used is SHA1 and this is in fact indicated in the signature field "MIDlet-Jar-RSA-SHA1"

- JAD file

  o Java Application Descriptor file containing information related to a JAR file.

- JAR file

  o Java Archive, containing the code to be run in the Java Virtual Machine.