Enhanced Jar Signing for Oracle E-Business Suite (Doc ID 1591073.1)

The information in this document applies to Oracle E-Business Suite Release 11i and R12.x .

The most current version of this document can be obtained through My Oracle Support Knowledge Document 1591073.1.

There is a change log at the end of this document.

In This Document

References to Oracle_Homes and Paths in this document are written from a Unix perspective. Windows users should substitute such commands as appropriate, for example, `$APPL_TOP` should be replaced by `%APPL_TOP%`.

Section 1: Introduction

Historically, JAR files that are downloaded to the desktop client from Oracle E-Business Suite (EBS) have been signed using a self-signed certificate with a 1024-bit key size. Using a self-signed certificate is no longer deemed secure when using the latest Java security standards.

This document describes the implementation of enhanced security within your Oracle E-Business Suite instances through JAR file signing using a Trusted Certificate Authority (CA) utilizing larger and more secure keysizes of 2048, 3072 or 4096 bits to provide stronger encryption. Signing jar files with a Trusted CA is a requirement for running jar content using the later JRE 7 releases running on the highest security settings. This will also become a requirement when using the Java default security settings in a future release.

It is strongly recommended that Oracle E-Business Suite users sign their Java content using a Trusted CA.

Terminology

The information in this section provides more detail on the meaning of some of the terms used within this document.

Self-Signed Certificate:

This has been used historically and is the default method used within EBS as a simplistic way to sign Java content. There is no root certificate and it cannot be properly trusted. Along with unsigned code this is now regarded as an insecure way of delivering Java content. When running the latest JRE 7 releases this will now cause extra security prompts and in certain cases will not allow Java content to run.

Code Signing Certificate

A code signing certificate from a Trusted CA is required to sign your Java content securely. It allows you to deliver signed code from your server (e.g. jar files) to users desktops and verifying you as the publisher and trusted provider of that code and also verifies that the code has not been altered. A single code signing certificate allows you to verify any amount of code across multiple EBS environments. This is a different type of certificate to the

commonly used SSL certificate which is used to authorize a server on a per environment basis. You cannot use an SSL certificate for the purpose of signing jar files.

See Appendix D: Multiple Instances/Sharing a Digital Certificate for further information on sharing your keystore and other related files across multiple EBS environments.

Trusted CA:

In this document Trusted CA refers to the use of either an Official CA (such as Verisign, Thawte etc.) or your own in-house PKI/Certificate Authority CA solution (in-house CA).

Official CA

This refers to a certificate provided by an official certificate authority such as Verisign or Thawte for example. The root certificate from providers such as this are already included in the Java or Browser certificate stores and provides the easiest method of incorporating secure code signing into your EBS environment.

In-House CA

This is a certificate provided through your own in-house PKI/Certificate Authority CA solution. This is generally suitable if all of your clients are internal and you have a mechanism to have those clients trust the in-house CA. If using an in-house CA, it is your responsibility to ensure the client software trusts the CA by following the extra required steps within this document outlined under Step 4.4. Using an In-House Certificate Authority and Step 6.1. Install the Root Certificate.

Java Launch Behavior

The table below outlines the default security behavior when running Java based forms within Oracle E-Business Suite using the later JRE 7 releases. This is based on the type of certificate used to sign the jar files and the JRE release version.

Signing Certificate	JRE Security Setting	JRE 7u21 and 7u25	JRE 7u40 and 7u45	JRE 7u51 and Higher
Self-Signed Certificate [4]	Very High	Forms won't launch	Forms won't launch	Forms won't launch
	High (Default Setting)	Forms launch normally	Forms won't launch[1]	Forms won't launch
	Medium	Forms launch normally	Forms launch with warning [2]	Forms launch with warning [2]
Code signing certificate from a Trusted CA [3]	Very High	Forms launch normally	Forms launch normally	Forms launch normally
	High (Default Setting)	Forms launch normally	Forms launch normally	Forms launch normally

[1] JRE 7u40 and JRE 7u45 are now both past their expiration date, forms will no longer launch on a High security setting when using a self-signed certificate.

[2] The security Warning: **Running unsigned applications like this will be blocked in a future release because it is potentially unsafe and a security risk** will pop when jars using a self-signed certificate are detected. The user can click through this message to launch the Java content. Unlike earlier JRE releases there is no longer the option to remember this decision for future logins. When running JRE 7u40 and later this message will therefore appear every time you start a new session, it can no longer be suppressed unless your jars are signed with a Trusted CA.

[3] Unless previously installed into the Java certificate store you will still be asked to trust the publisher when launching an EBS environment for the first time from the desktop. See Step 6.3. Running an Environment for the First Time for further details.

[4] If using a Self-Signed Certificate it is still advisable to uptake the later AD patch code levels which contain a number of security enhancements. See Step 2.2. AD Patch Requirements for further details.

For further information on Java Security pop-ups see, What should I do when I see a security prompt from Java?

Section 2: Prerequisite Requirements

Step 2.1. Enhanced Jar Signing Prerequisite Requirements

- Oracle Applications 11.5.10 CU2 or higher or Oracle Applications R12.x or higher
  - For prerequisite patch requirements see Step 2.2. AD Patch Requirements
  - Desktop Client: JRE 6 or higher
  - Application Tier: JDK 6 or JDK 7

JDK Upgrade Documentation

It is recommended to upgrade to the latest JDK release on your application tier. For information on minimum JDK version requirements and update instructions for your platform, see the following document as appropriate to your Oracle E-Business Suite release:

- 11i Users:
  - Document 401561.1 titled, 'Using J2SE Version 6 with Oracle E-Business Suite 11i'
- R12.0.6, 12.1.3 Users:
  - Document 455492.1 titled, 'Using Latest Java 6.0 Update With Oracle E-Business Suite Release 12'
  - Document 1467892.1 titled, 'Using JDK 7.0 Latest Update with Oracle E-Business Suite Release 12.0 and 12.1'
- R12.2.x Users:
  - Document 1459546.1 titled, 'Using JDK 6.0 Latest Update with Oracle E-Business Suite Release 12.2'
  - Document 1530033.1 titled, 'Using the Latest JDK 7.0 Update with Oracle E-Business Suite Release 12.2'

Ascertain Current Software Versions (if required)

If required, you can check your current software versions by running the TXK Inventory Report for the application tier. For further information see Appendix F: Obtaining Support.

Step 2.2. AD Patch Requirements

The commands and steps in this document rely on the latest AD architecture in this area. If you have not done so previously, download and apply the AD prerequisite patch as applicable to your Oracle E-Business Suite release.

Step 2.2.1. Download the AD Patches

Download the appropriate patch(es) for your operating system and Oracle E-Business Suite release and place it on your server:

Patch	EBS 11i [3]	EBS 12.0.x [4]	EBS 12.1.x	EBS 12.2.x
Consolidated JRI Patch	Patch 17191279 [5]	Patch 17191279:R12.AD.A	Patch 17191279:R12.AD.B	Patch 17545215:R12.AD.C [1] or later
Change Alias Patch	Patch 19725648	Patch 18312333:R12.AD.A [2]	Patch 18312333:R12.AD.B [2]	Patch 17766337:R12.AD.C.delta.4 or later

[1] Oracle E-Business Suite 12.2.2 users must apply this patch. Oracle E-Business Suite 12.2.3 or higher users do not need to apply this patch as it is already included in the base release.

[2] With previous Patch 17066340, although the process completed successfully, there may have been a core dump when executing `adgenjky.sh` if the `$APPL_TOP/admin/adsign.txt` file did not exist. This issue is fixed in Patch 18312333 which supersedes it. If you have previously run this process against Patch 17066340 there is no requirement to re-run it against Patch 18312333.

[3] Extended support for EBS 11i ended 31-December 2015. Additional support coverage is available through Oracle Advanced Customer Support (ACS). See the following announcement for further information: 'Additional Coverage

Options for 11.5.10 E-Business Suite Sustaining Support' (Document 1596629.1).

[4] Extended support for EBS 12.0.6 ended 31-January 2015. The patches continue to be listed for historical information purposes.

[5] In addition to Patch 17191279, also apply Patch 17726123 which includes a later version of the aiojava.lc file. For further information and patching instructions, see <Doc 1604018.1> titled, '11.5.10.2 : Patch Fails With Memory Error : Unable to allocate memory in procedure aiumab()'.

Linux x86-64 Users

AD utilizes a 32-bit techstack on Linux therefore Linux x86-64 users must download and apply the Linux x86 patch.

Step 2.2.2. Run the adgrants.sql Script

Before applying the 'Consolidated JRI Patch', run the adgrants.sql script as a user that can connect as SYSDBA to grant privileges to selected SYS objects and create PL/SQL profiler objects.

1. Create the directory `$ORACLE_HOME/appsutil/admin` on the database server if it does not already exist.

2. Set the environment to point to ORACLE_HOME on the database server.

3. If you already have a version of adgrants.sql under your ORACLE_HOME that is a later version than the one supplied in the patch, go to step 4.
   If not, copy `<patch_dir>/admin/adgrants.sql` (UNIX) from this patch directory to `$ORACLE_HOME/appsutil/admin`.
   Or, copy `<patch_dir>\admin\adgrants_nt.sql` (Windows) from this patch directory to `%ORACLE_HOME%\appsutil\admin`.

4. Use SQL*Plus to run the script:

UNIX Users:

```
$ sqlplus /nolog
SQL> @$ORACLE_HOME/appsutil/admin/adgrants.sql <APPS schema name>
```

Windows Users:

```
C:\> sqlplus /nolog
SQL> @%ORACLE_HOME%\appsutil\admin\adgrants_nt.sql <APPS schema name>
```

Step 2.2.3. Apply the Patch

Apply the patch by following its readme.

Oracle E-Business Suite 12.2.x Users

Apply the patch through an AD Online Patching (ADOP) session

Oracle E-Business Suite 11i, 12.0.x & 12.1.x Users

Apply the patch through adpatch.

Step 2.2.4. Post Install Tasks

This patch also contains the fix to add permission attributes to the Jar Manifest in line with the latest Java security recommendations. This change will not take effect until you regenerate your jar files as outlined in Section 5: Regenerate the Jar Files. If you are continuing through the rest of the steps in this document you can defer the jar regeneration until later.

Section 3: Generate Keypair and Certificate Signing Request

This section outlines the required steps to generate a keypair and create a certificate signing request.

Jar Signing Files Directory

Note: In Oracle E-Business Suite 12.2.x jar signing files are located under the 'Non-Editioned File System' `$NE_BASE`. In Oracle E-Business Suite 12.1 they are located under `$APPL_TOP/admin`. For the purposes of this document the alias ***<JRI_DATA_LOC>*** is used to denote this directory.

Oracle E-Business Suite 12.2.x Users

```
<JRI_DATA_LOC> = $NE_BASE/EBSapps/appl/ad/admin
```

Oracle E-Business Suite 11i, 12.0.x and 12.1.x Users

```
<JRI_DATA_LOC> = $APPL_TOP/admin
```

Step 3.1. Source the Environment

Source your APPS env File (EBS 12.2.x Users)

On the Application tier as the file system owner source your RUN file system.

Note: The configuration steps can be applied with either the RUN or PATCH File System sourced. If you are currently running an ADOP Online Patching session you must make these changes with the PATCH File System sourced. If this is the case please ensure the cutover phase is run after regenerating the Jar Files in Section 5: Regenerate the Jar Files.

Source your APPS env File (EBS 11i, 12.0.x and 12.1.x Users)

On the Application tier as the file system owner source your APPS env file.

Step 3.2. Generate a new keypair (private key and public key)

Note: If you wish to change your current keystore passwords, do so now before generating your keypair and following the rest of the steps in this document. For information on changing your keystore passwords see Identify and Change the Keystore Passwords.

Generate Keypair

If you wish to create a new keypair (private key and public key) with the RSA algorithm and your selected bit key size and load it into the keystore `adkeystore.dat` run the following command. This will create a new keystore and also back up any existing keystore (`adkeystore.dat` to `adkeystore.bak`) as well as the `adsign.txt` file under the `<JRI_DATA_LOC> directory` which are used in the signing process:

- `adkeystore.dat`
- `adsign.txt`
- `adkeystore.bak`

The following reference file is also created under the `$APPL_TOP/admin` directory:

- `JavaVersionFile`

Definitions:
`adkeystore.dat` - the keystore file that is used to sign jar files on the server.
`adsign.txt` - Used to pass arguments to the JRI during file signing. The first value within this file is your alias.
`adkeystore.bak` - a back up copy of your previous `adkeystore.dat` keystore taken before the new one is created.
`JavaVersionFile` - A reference file showing the Java version used in compilation (The JDK version on your server)

Run the following command:

```
$ adjkey -initialize [ -keysize < 2048 | 3072 | 4096 > ] [ -alias <alias_name> ]
```

Note: The `alias` and `keysize` parameters are optional

- Valid options for the `-keysize` parameter are 2048, 3072 or 4096
  - If you do not include the `-keysize` parameter it will use the default 2048 bit key size.
- If you do not include the `alias` parameter it will be created using the environments `$CONTEXT_NAME` by default.
  - If you wish to change the alias from the current value this must be done **before** running this command to create the new adkeystore.dat.
    - To use the `alias` parameter with the `adjkey` command, apply the 'Change Alias Patch' as outlined in Step 2.2. AD Patch Requirements.
  - Do not include spaces in your `alias` name.
  - The **same** `alias` name must be used in Step 3.3. Create a Certificate Signing Request & Step 4.5.3. Import your Java Code Signing Certificate

For example, to create a **2048** bit key size with the alias name **Atoz_Widgets** enter the following command replacing the values in **bold** as applicable to your organization:

```
$ cd <JRI_DATA_LOC>

$ adjkey -initialize -keysize 2048 -alias Atoz_Widgets

  Enter the APPS username: <appsusername>
  Enter the APPS password: <appspassword>

      Enter the COMMON NAME [ ] : Atoz Widgets Ltd

      Enter the ORGANIZATION NAME [Atoz Widgets Ltd] : Atoz Widgets Ltd

      Enter the ORGANIZATION UNIT [ ] : Sales

      Enter the LOCALITY (or City) [ ] : Madison City

      Enter the STATE (or Province or County) [ ] : Missouri

      Enter the COUNTRY (two-letter ISO abbreviation) [ ] : US
```

COMMOM NAME / ORGANIZATION NAME

In earlier releases of the AD code the COMMON NAME and ORGANIZATION NAME were combined into a single entry:

```
Enter the Name of your Company (used for both CN and ORGANIZATION NAME) [CN/ORGANIZATION
NAME]
```

The ability to enter different values for the COMMON_NAME and ORGANIZATION NAME is available with R12.AD.C.delta.6 and higher for EBS 12.2 users and R12.AD.B.delta.7 and higher for EBS 12.1 users.

Additional Information

If required, you can view the contents of adkeystore.dat by running the following command:

```
$ keytool -list -v -keystore adkeystore.dat
```

Step 3.3. Create a Certificate Signing Request

Create a "Certificate Signing Request" (named `adkeystore.csr` in this example) to send to your CA provider for signing. This will be created using the same `alias` name from Step 3.2. Run adjkey to create a new keypair (Private key and public key).

Secure Hash Algorithm

The use of a SHA-1 algorithm is deprecated. It is recommended that a SHA-2 algorithm is used when creating your signing request.

Command to Create a Certificate Signing Request

Create a certificate signing request using a SHA-2 algorithm replacing the values in **bold** as applicable to your organization..

```
$ keytool -sigalg SHA256withRSA -certreq -keystore <JRI_DATA_LOC>/adkeystore.dat -file
<JRI_DATA_LOC>/adkeystore.csr -alias <alias_name>

Enter keystore password: <store_passwd>
Enter key password for <alias_name>: <key_password>
```

Where:

<JRI_DATA_LOC> is the path outlined under Jar Signing Files Directory.
<alias_name> is the alias value you used in Step 3.2. Run adjkey to create a new keypair (Private key and public key).

Additional Information

You can verify the algorithm used within your 'Certificate Signing Request' (`adkeystore.csr`) by running the following command:

```
$ openssl req -in adkeystore.csr -text -noout | grep "Signature Algorithm"
```

Step 3.4. Submit your Certificate Signing Request

Submit your certificate signing request `'adkeystore.csr'` to your official certificate authority, for example, Verisign, Thawte etc. or to your own in-house certificate authority as applicable.

> **Note:** Be sure to request a **Java Code Signing Certificate**. This certificate can be used to sign your jar content across one or multiple Oracle E-Business Suite environments.

Section 4: Import your Certificate(s)

If you are using a PKSC #12 certificate, from Comodo for example, you cannot use the instructions in this section to import your Java code signing certificate. Follow the steps in PKCS #12 Certificates (*.pfx, *.p12), Comodo and then continue to Section 5: Regenerate the Jar Files.

This section covers the process required to import your certificate(s) into the appropriate keystore.

Certificate Chain of Trust

Depending on your certificate provider your certificate chain of trust will consist of two or more certificates:

Two Certificate Chain of Trust

- Java code signing certificate
- Root certificate

Three or more Certificate Chain of Trust

- Java code signing certificate
- Intermediate certificate (one or more)
- Root certificate

In general, only customers using their own in-house certificate authority will need to import their 'root certificate' into the Java public keystore, `cacerts`, therefore most users will probably not need to touch this keystore.

All users will need to import their 'Java code signing certificate' and if the certificate chain of trust includes them, any 'intermediate certificates' into the keystore, `adkeystore.dat`.

> **Note:** This process can be done in a number of ways, for example you could also import the 'root' plus any 'intermediate certificates' into the Java Certificate Keystore, `cacerts` and then just import your 'Java code signing certificate' into the keystore, `adkeystore.dat`. Assuming most customers will be using a certificate from a recognized Trusted CA this document shows the steps to import any 'intermediate certificates' into the keystore, `adkeystore.dat` negating the need to touch the Java Certificate Keystore, `cacerts` at all. Multiple certificate chains of trust may also be concatenated and then imported to the keystore, however this can cause problems with missing hashes if not using the correct format, so again this is not covered within this document to try and keep the process as generic as possible.

Step 4.1. My Java Code Signing Certificate is Now Available

The `adkeystore.csr` file created in Step 3.3 Create a Certificate Signing Request should now have been signed, encoded and formatted to be recognized as a 'signed code certificate' by your certificate authority which will be used to verify the authenticity of downloaded content.

Once you have received the 'signed code certificate' back from your Certificate Authority, continue with the steps below to complete the process.

Step 4.2. Prepare your Oracle E-Business Suite Environment

Source and prepare your environment as applicable to your Oracle E-Business Suite release.

Oracle E-Business Suite 12.2.x Users

Source your APPS env File (EBS 12.2.x Users)

On the Application tier as the file system owner source your RUN file system.

> **Note:** The configuration steps can be applied with either the RUN or PATCH File System sourced. If you are currently running an ADOP Online Patching session you must make these changes with the PATCH File System sourced. If this is the case please ensure the cutover phase is run after regenerating the Jar Files in Section 5: Regenerate the Jar Files.

Oracle E-Business Suite 11i, 12.0.x and 12.1.x Users

Source your APPS env File (EBS 11i, 12.0.x and 12.1.x Users)

On the Application tier as the file system owner source your APPS env file.

Step 4.3. Add the Root Certificate to cacerts (if required)

The Java Keystore 'cacerts'

The Java Keystore named `caerts` is system wide keystore used by Java to store the 'Public Root Certificate' of your CA provider which already contains numerous root certificates from recognized Certificate Authorities. In the majority of cases you will therefore not need to run this step unless you are using your own in-house CA. If required you can verify if your root certificate is already included by following the commands under List Java Keystore Certificate Files before continuing from the appropriate step below.

Step 4.3.1. I need to add the 'Root Certificate' to cacerts

If you need to import the 'Root Certificate' into `cacerts` continue to Step 4.4. Add the Root Certificate to cacerts.

Step 4.3.2. My 'Root Certificate' is already included in caerts

If your 'Root Certificate' is already included in `cacerts` as will generally be the case when using a recognized certificate authority continue to Step 4.5: Import the Java Code Signing Certificate into the Keystore to import your code signing certificate and

Step 4.4. Import the Root Certificate to the Java Keystore Certificate Store 'cacerts' (if required)

The `cacerts` system-wide keystore certificates file contains many root CA certificates. By default the root certificate(s) for your in-house CA will not be present in the `cacerts` file. If you are using a certificate from an in-house CA, or the root certificate from your recognized CA is not already included, import it into `cacerts` so that the jarsigner program trusts it.

Note: Whenever you upgrade your jdk version on the server any additional certificates you have added to your cacerts file will be lost. You will need to re-import the root certificate or keep a copy of your original cacerts file which you can copy back in.

Security Properties Directory

For the purpose of this document the alias ***<SEC_PROP_LOC>*** is used to denote the "security properties directory".

Oracle E-Business Suite Release 12 "Security Properties Directory":

```
<SEC_PROP_LOC> = $OA_JRE_TOP/lib/security/
```

Oracle E-Business Suite Release 11i "Security Properties Directory":

```
<SEC_PROP_LOC> = $OA_JRE_TOP/jre/lib/security/
```

Step 4.4.1. Copy your Root Certificate to the Security Properties Directory

Note: For the purpose of the commands used in this step, we will rename the root certificate file `myca.crt` and will use the alias `myca` when inserting it into the `cacerts` keystore certificate file. If preferred you may use the name and alias of your choice and alter the commands below accordingly.

Copy your root certificate to the "security properties directory", `<sec_prop_loc>` where your `cacerts` file resides. Replace the values in **bold** as applicable to your organization:

```
$ cp <root_certificate> <sec_prop_loc>/<root_certificate>
```

Where:

<root_certificate> is the root certificate name and extension that you received from your CA provider. (e.g. root.crt, root.cer etc.)

Step 4.4.2. Import the Root Certificate into cacerts

From your **<SEC_PROP_LOC>** directory, import your Root Certificate by running the the following command, replacing the values in **bold** as applicable to your organization. You will need to alter the permissions of cacerts before you can import into it as shown in the example below:

```
$ keytool -import -alias <certificate_alias> -file <certificate_name> -trustcacerts -v -keystore
cacerts
```

Where:

<certificate_alias> is a unique alias that can be used to identify the certificate in the store. You can use any name you like for this. The example below is using 'myca'
<certificate_name> is the name of your root certificate including the file extension (e.g. root.crt, root.cer). The example below is using 'myca.crt'

Example:

You will need to alter the permissions for `cacerts` before you can import your certificate file into it as shown in the example below. Replace the values in **bold** as applicable to your organization and certificate name and extension. When running the command you will be asked to enter the keystore password **'<cacerts_password>'** and trust the certificate by answering **y** when requested. If you wish to set a new password before doing this see Changing the Keystore Certificate File Password.

```
$ cd <SEC_PROP_LOC>
$ chmod +w cacerts

$ keytool -import -alias myca -file myca.crt -trustcacerts -v -keystore cacerts

  Enter keystore password: <cacerts_password>

  Trust this certificate? [no]: y
  Certificate was added to Keystore
  [Storing cacerts]

$ chmod a-w cacerts
```

Additional Information

- Changing the Keystore Certificate File Password
- List Java Keystore Certificate Files

Changing the Keystore Certificate File Password

The keystore certificate file (`cacerts`) password can be changed using the following command, replacing the values in **bold** as applicable to your organization:

```
$ cd <SEC_PROP_LOC>
$ keytool -storepasswd -keystore cacerts

  Enter keystore password: <cacerts_password>
  New keystore password: <new_cacerts_password>
  Re-enter new keystore password: <new_cacerts_password>
```

List Java Keystore Certificate Files

If required you can list all the root certificates that are included in the keystore certificate file, `cacerts` using the following command as appropriate to your Oracle E-Business Suite release:

```
$ cd <SEC_PROP_LOC>
$ keytool -list -keystore cacerts

  Enter keystore password: <cacerts_password>
```

If you are looking for a particular certificate you can search for it by alias name, or part of the alias name. Replace the values in **bold** as applicable to your organization. The alias name is stored in lower case. To check for included verisign certificates for example, replace **myca** with **verisign**:

```
$ cd <SEC_PROP_LOC>
$ keytool -list -keystore cacerts | grep -i myca

  Enter keystore password: <cacerts_password>
```

Step 4.5. Import the Code Signing Certificate into the Keystore

This step will import your 'Code Signing Certificate' (built from your adkeystore.csr file and signed by your trusted CA) into the keystore, `adkeystore.dat` along with any intermediate certificates, if required.

Step 4.5.1. Copy and Rename the 'Code Signing Certificate'

Note: Rename the 'Code Signing Certificate' signed by your trusted CA to `adkeystore.<ext>` (where <ext> = your certificate extension, for example *.crt, *.cer etc.). This extension can vary depending on your certificate provider. In this example we are assuming a code signing certificate with a *.crt extension and therefore naming the certificate `adkeystore.crt`.

Example:

Copy the code signing certificate you received from your trusted CA to the `$<JRI_DATA_LOC>` directory and rename it `adkeystore.<ext>`, replacing the values in bold as applicable to your certificate name and extension:

```
$ cp code_signing_certificate.crt <JRI_DATA_LOC>/adkeystore.crt
```

Step 4.5.2. Import your Intermediate Certificate(s) (if required)

If you have any intermediate certificates in your chain of trust, import them into the keystore `adkeystore.dat` using an `alias` of your choice that is **different** from the one used when creating your key pair in Step 3.2. Run adjkey to create a new keypair (Private key and public key). . If you have more than one intermediate certificate you must import them in chain of trust order starting with the one that directly follows the root certificate. Use the following command replacing the values in bold as applicable to your organization.

```
$ cd <JRI_DATA_LOC>

$ keytool -import -file <intermediate_cert> -trustcacerts -alias <intermediate_alias_name> -
keystore adkeystore.dat
```

Where:

<JRI_DATA_LOC> is the path outlined under Jar Signing Files Directory.

<intermediate_cert> is the name of your intermediate certificate including the file extension (e.g. *.crt, *.cer etc.). The example below is using 'intermediate.crt'.

<intermediate_alias_name> is a **unique alias** that can be used to identify the intermediate certificate in the store. You can select any unique name for this **except** for the alias used when creating your key pair in Step 3.2. Run adjkey to create a new keypair (Private key and public key). The example below is using 'interCA'.

Example:

```
$ cd <JRI_DATA_LOC>

$ keytool -import -file intermediate.crt -trustcacerts -alias interCA -keystore adkeystore.dat
```

Step 4.5.3. Import your Java Code Signing Certificate

Import your 'Java Code Signing Certificate' into the keystore, `adkeystore.dat` replacing the values in bold as applicable to your organization. When importing this certificate you must use the same alias name that was used when creating your key pair in Step 3.2. Run adjkey to create a new keypair (Private key and public key).

```
$ cd <JRI_DATA_LOC>

$ keytool -import -file <cs_cert> -trustcacerts -alias <alias_name> -keystore adkeystore.dat
```

Where:

<JRI_DATA_LOC> is the path outlined under Jar Signing Files Directory.

<cs_cert> is the name of your code signing certificate including the file extension (e.g. *.crt, *.cer etc.). The example below is named 'adkeystore.crt'

<alias_name> must be the same alias value you used in Step 3.2. Run adjkey to create a new keypair (Private key and public key).

Example:

```
$ keytool -import -file adkeystore.crt -trustcacerts -alias <alias_name> -keystore adkeystore.dat
```

> **Note:** This command will fail if the root certificate of your trusted CA is not included in your `cacerts` keystore certificate file, see keytool error: java.lang.Exception: Failed to establish chain from reply.

Your keystore, `adkeystore.dat`, should now contain your private key and a corresponding CA-signed code-signing certificate. AD tools can now sign the JAR files served to clients using this information.

Step 4.5.4. Backup your Completed Files

It is advisable to take a copy of your keystore, `adkeystore.dat` and `adsign.txt` file for safe keeping in case you need to re-use them on any other Oracle E-Business Suite environments you may have. See Appendix D: Multiple Instances/Sharing a Digital Certificate for further information.

Section 5: Regenerate the Jar Files

Step 5.1. Shutdown the Application Tier

Step 5.1.1. Stop the Application Tier

Source your APPS env file and shutdown your application tier services.

Source your APPS env File

On the Application tier as the file system owner source your APPS env file.

Shutdown the Application Tier

Shut down the application tier services:

```
$ adstpall.sh <apps_user>/<apps_pwd>
```

Step 5.1.2. Regenerate the jar files through adadmin

Regenerate all JAR Files using the force option through **adadmin**:

Run **ADADMIN**, and select the following from the AD Administration Main Menu:

Choose **Generate Applications Files menu**
From this menu choose **Generate product JAR files**

Enter **yes** when prompted with: **Do you wish to force regeneration of all jar files? [No] ? yes**

Once your jar files have been successfully generated, restart the application tier.

Step 5.1.3. Restart the Application Tier

Restart the application tier services:

```
$ adstrtal.sh <apps_user>/<apps_pwd>
```

Proceed to Section 6: Desktop Client Requirements.

Section 6: Desktop Client Requirements

Step 6.1. Install the Root Certificate

Download your CA root certificate for example, <root>.crt to the desktop and then follow the instructions for your desktop operating system.

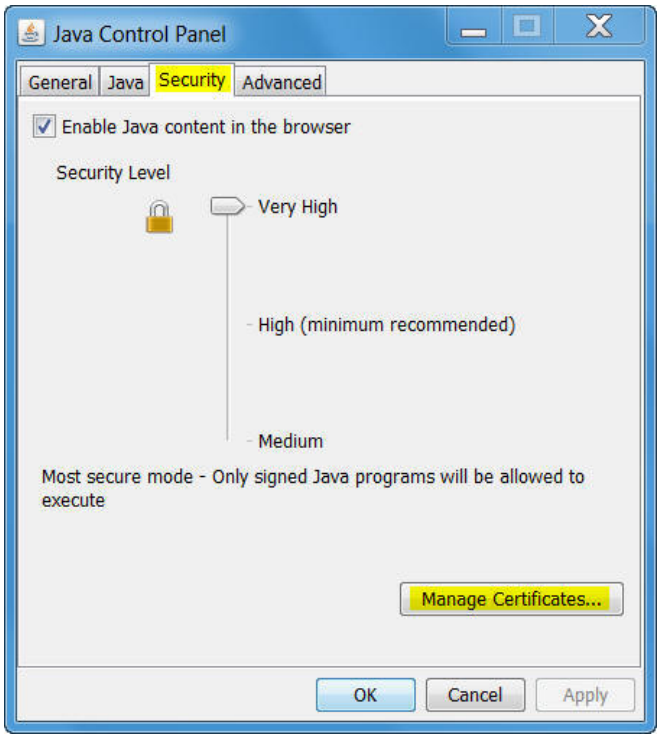
Windows: Internet Explorer and Firefox Users

Many root certificates from an official certificate authority are already installed in the Java or browser certificate store. If you are using a in-house CA the root certificate must be installed into a location accessible by Java on your desktop client and your EBS environment. The detailed example below shows how to add the certificate through the Java Control Panel which is a generic solution for both Internet Explorer and Firefox users.

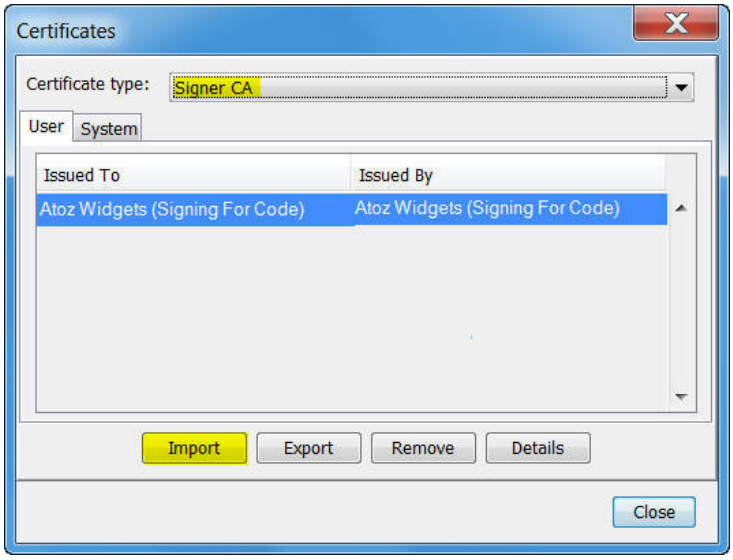
Internet Explorer users may instead install the CA root certificate into the 'Trusted Root Certificate Authorities' store in the browser if they do not wish to use the Java store through the Java Control Panel.

```
Tools -> Internet Options -> Content (tab) -> Certificate (button) -> Trusted Root
Certification Authorities (tab) -> Import (button)"
```

1. Open the 'Java Control Panel' on your desktop and select the 'Security' (tab).
2. Click `Manage Certificates` to open the 'Certificates' Window



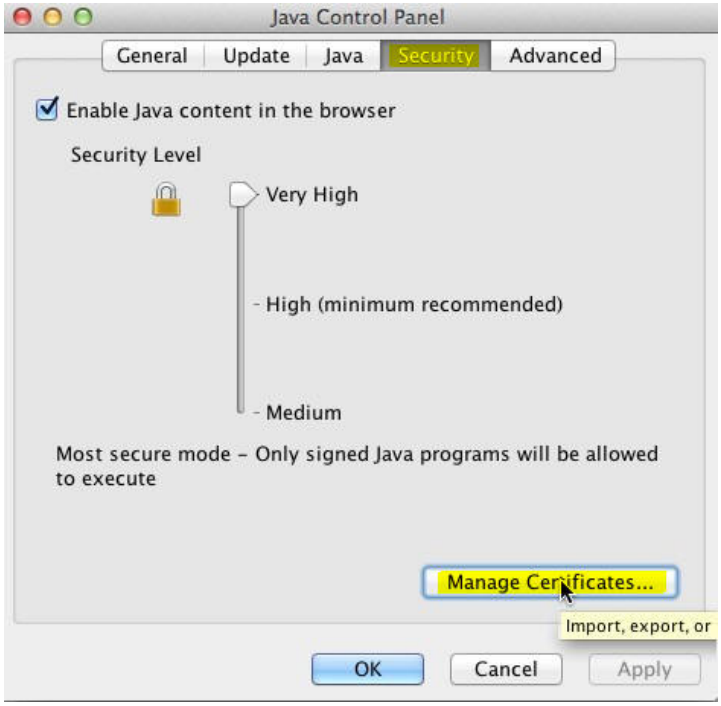
3. Choose `Certificate type: Signer CA`
4. Click `Import` and upload the *`<root>.crt`* from your desktop



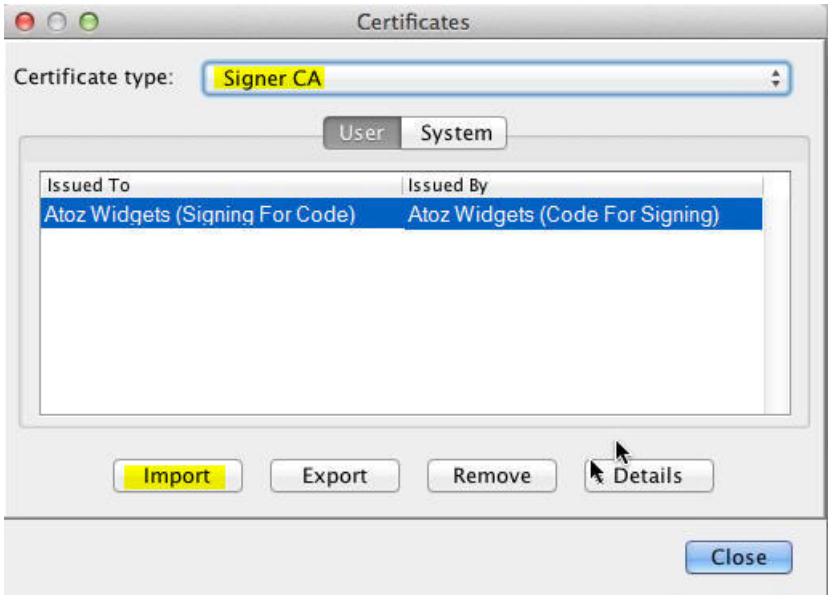
5. Close the 'Certificates' window and the 'Java Control Panel'. The certificate should now be installed in the Java 'Signer CA' store.

Mac: Safari Users

1. Open the 'Java Control Panel' on your desktop and select the 'Security' (tab).
2. Click `Manage Certificates` to open the 'Certificates' Window



3. Choose `Certificate type: Signer CA`
4. Click `Import` and upload the *`<root>.crt`* from your desktop



5. Close the 'Certificates' window and the 'Java Control Panel'. The certificate should now be installed in the Java 'Signer CA' store.

Step 6.2. Java Security Setting

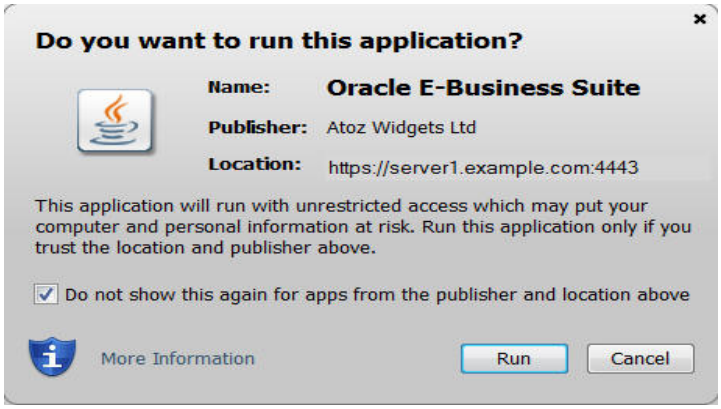
By default Java runs on a 'High' security setting which currently does **not** uptake the latest Java security enhancements. Please update the Java security level to 'Very High' within the 'Java Control Panel' if you are running JRE 1.7.0_21 (7u21) or higher and wish to prevent the use of Self-Signed certificates from the users desktop.

```
Java Control Panel -> Security -> Security Level: Very High
```

Step 6.3. Running an Environment for the First Time

After installing a new certificate authority in an Oracle E-Business Suite environment the jar signing certificate will need to be installed in the Java certificate store on the desktop client before you can successfully access any Java content. You will be asked to trust the publisher when accessing the Oracle E-Business Suite environment for the first time from a desktop client.

When attempting to launch Java content the following message window should pop:



When asked "Do you want to run this application?"

- Check "Do not show this again for apps from the publisher and location above"
  - (This will install the certificate into Java's 'Trusted Certificates' store and stop this message appearing when accessing this environment in future)
- Click the "Run" button

Step 6.4. Verify the Forms JAR Files are signed with the new certificate

If you wish to verify that your Jar files are now signed correctly with the new certificate, run the following steps from your desktop client:

- Navigate to 'Control Panel' -> 'Java' -> 'Security' (tab) -> 'Manage Certificates' (button) -> 'Certificate Type: Trusted Certificates'
- Select your new certificate from the list and click the 'Export' button, save the file as <hostname>.crt, e.g. myhost.crt
- Double click on the <hostname>.crt file to display the certificate.
- Click the 'Details' tab and verify the following values correspond to the values entered when you ran Step 3.2. Run adjkey to create a new keypair (Private key and public key):
  - '**Valid from**' displays the date and time that you created the certificate
  - '**Subject**' contains values for CN, OU, O, L, S, C
    namely [CN], [ORGANIZATION UNIT], [ORGANIZATION NAME], [LOCALITY], [STATE], [COUNTRY]
  - '**Public Key**' displays 'RSA (XXXX Bits)' e.g. 'RSA (3072 Bits)

If you wish to verify the signature on any single jar file see, Verify the Digital Signature of a Signed Jar File

Section 7: Known Issues

1. Account Hierarchy Manager & Financial Dimensions Hierarchy Manager (EBS 11i)
2. Oracle Learning Management (EBS R12)
3. adogjf() Unable to generate jar files under JAVA_TOP
4. keytool error: java.lang.Exception: Failed to establish chain from reply
5. jarsigner error: gnu.javax.crypto.keyring.MalformedKeyringException
6. Application Blocked
7. Windows Server Issues
8. ORA-01031: insufficient privileges
9. Certificate Revocation Lists (CRL) Error using LDAP URL
10. Certificate Revocation OCSP/CRL Internet Requirements
11. keytool error: java.io.IOException: Invalid keystore format
12. java.lang.Exception: Public keys in reply and keystore don't match
13. Unsigned Jars show access denied errors (sun.awt, sun.java2d)

1. Account Hierarchy Manager & Financial Dimensions Hierarchy Manager

If you are running Oracle E-Business Suite 11i, Account Hierarchy Manager (AHM) & Financial Dimensions Hierarchy Manager (FDHM) use content within $JAVA_TOP/jbodatum111.jar. This jar file is not administered through the adadmin jar signing function.

A trusted signed version of this jar file including the latest Manifest attribute updates is available through Patch 17391300

Alternatively you can manually update the Manifest and sign your existing jar file as outlined in Appendix C: Jar File Manifest Updates and Jar File Signing

2. Oracle Learning Management (EBS R12)

"The Application's Digital signature has an Error. Do you want to run the Application?"

Oracle Learning Management (OLM) uses jar content that is not administered through the adadmin jar signing function.
To fix this issue, apply Patch 16825765 by following its readme. This patch includes trusted signed versions of the required jar files which also include the latest Manifest attribute update requirements.

Note: The 12.2.x version of this patch will be released shortly. The jars can be signed manually if required in the meantime as outlined in Appendix C: Jar File Manifest Updates and Jar File Signing .

3. adogjf() Unable to generate jar files under JAVA_TOP

Particularly if you are upgrading a cloned environment you may find that regenerating you jar files fails with an error saying;

```
adogjf() Unable to generate jar files under JAVA_TOP
```

This may be happening because the `<alias>` parameter being used in the command from Step 3.2. Generate a new keypair (private key and public key) does not match the first value of your `$<JRI_DATA_LOC>/adsign.txt` file. Please change this value to the literal value of your `<alias>` and regenerate your jar files.

For example if your alias is `Atoz` but your `$<JRI_DATA_LOC>/adsign.txt` file shows `VIS_123 1 CUST` back up your adsign.txt file then update it to: `Atoz 1 CUST`

4. keytool error: java.lang.Exception: Failed to establish chain from reply

This error may occur if the root certificate for your Certificate Authority is not present in the Java keystore, `cacerts`. If you are using an In-house CA, or your recognized root CA is not included by default, please make sure you followed Step 4.4. Import the Root Certificate to the Java Keystore Certificate Store 'cacerts' (if required) to import your root certificate into `cacerts`.

A second scenario that could return this error is if you are trying to import your Java code signing certificate into the keystore, `adkeystore.dat` without first importing any required intermediate certificates from your certificate chain. See Step 4.5: Import the Java Code Signing Certificate into the Keystore.

5. jarsigner error: gnu.javax.crypto.keyring.MalformedKeyringException

"jarsigner error: gnu.javax.crypto.keyring.MalformedKeyringException: incorrect magic" can appear when running a jarsigner command where you have multiple jarsigner installs on your environment.

To fix this issue, add the full path to jarsigner for the command you are running. By default this is available under:

Oracle E-Business Suite 12.2.x Users

```
$FMW_HOME/jrockit32/bin/jarsigner
```

Oracle E-Business Suite 12.0.x and 12.1.x Users

```
$IAS_ORACLE_HOME/appsutil/jdk/bin/jarsigner
```

Oracle E-Business Suite 11i Users

```
$COMMON_TOP/util/java/1.6/jdk<version>/bin/jarsigner
```

6. Application Blocked

The "Application Blocked by Security Settings" error may pop when trying to run Java content within Oracle E-Business Suite.

After clicking the "OK" button you may also see the "Application Blocked for Security" error, "Failed to validate the certificate. The application will not be executed" error.

These errors may occur because the correct root certificate has not been correctly installed in your Java security store, see Step 6.1. Install the Root Certificate for the steps required to correct this issue.

7. Windows Server Issues

AutoConfig Error for Windows Users

Windows users may see the following error after an AutoConfig failure:

```
ORA-06576: not a valid function or procedure name Command option -storepass needs an
argument.
```

To fix this issue run the following command replacing the **<alias>** value as applicable to your organization:

```
keytool -exportcert -alias <alias> -keystore <JRI_DATA_LOC>\adkeystore.dat -file
<JRI_DATA_LOC>\admin\appltop.cer
```

8. ORA-01031: insufficient privileges

If you encounter the following error when running Step 2.2.3. Apply the Patch, make sure you have run `adgrants.sql` as outlined in Step 2.2.2. Run the adgrants.sql Script then run the patch again:

```
begin
*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SYS.DBMS_SESSION", line 101
ORA-06512: at "APPS.AD_JAR", line 44
ORA-06512: at line 2
```

9. Certificate Revocation Lists (CRL) Error using LDAP URL

The Certificate Revocation Lists (CRL) check may fail if using an LDAP URL causing forms launch to fail with the following error:

```
java.lang.ClassNotFoundException: oracle.apps.fnd.formsClient.FormsLauncher.class
```

The CRL check was introduced in JRE 7u25 so this error may also be seen when using earlier releases than JRE 7u45 in certain circumstances.

How to Fix this Issue

This Issue is fixed in JRE 7u51-b32 and higher which is only available through Patch 17981166.

(All Java patches contain both the JRE and JDK releases. Only extract the JRE version of Java from the patch and install on your desktop to fix this issue.)

Workaround

If you are unable to upgrade at this time, turn off the CRL check in the Java Control Panel as a temporary workaround:

```
Java Control Panel -> Advanced (tab) -> Perform certificate revocation checks on -> Do not check
(not recommended)
```

10. Certificate Revocation OCSP/CRL Internet Requirements

To enhance security, the certificate revocation checking feature has been enabled by default since Java 7 Update 25. Prior to launching a signed application, the associated certificate will be validated to ensure that it has not been revoked by the issuing authority. This feature has been implemented using both On-line Certificate Status Protocol (OCSP) and Certificate Revocation List (CRL) mechanisms. For further information see Certificate Revocation in the Java 7u25 release notes.

Turn Off Revocation Checking

If you are using a certificate from a recognized trusted CA you will likely require access to their website for certificate validation to work. If you have a locked down environment with no internet access this check will fail. In such cases revocation checking may be turned off through the Java Control Panel:

```
Java Control Panel -> Advanced (tab) -> Perform certificate revocation checks on -> Do not check
(not recommended)
```

For further information on certificate revocation configuration see How to configure certificate revocation checking from the Java Control Panel?

Allow Internet Access to Revocation URL

If preferred you can instead use the certificate revocation feature by allowing access to the URL through the firewall. To check the URL either copy your certificate (root or code signing) to the desktop, or export it from the Java Control Panel then open it. Under the 'Details' tab check the values for the fields titled 'CRL Distribution Points' and 'Authority Information Access'.

11. keytool error: java.io.IOException: Invalid keystore format

When trying to add the Java Code Signing Certificate to the keystore, `adkeystore.dat` you get the following error, 'keytool error: java.io.IOException: Invalid keystore format'. One reason for this error could be that you concatenated your root and any intermediate certificates in the wrong format before importing into Java certificate keystore, `cacerts`. Adding them individually or concatenating in x509 format should resolve the problem. This document does not cover this scenario, see Section 4: Import your Certificate(s).

12. java.lang.Exception: Public keys in reply and keystore don't match

Importing your code signing certificate as outlined in Step 4.5: Import the Java Code Signing Certificate into the Keystore fails with, 'keytool error: java.lang.Exception: Public keys in reply and keystore don't match'. This can occur if you are trying to import a certificate into your keystore, `adkeystore.dat` which does not contain the same key values that were used when the certificate was created. For example if you have recreated the keystore since previously creating the code signing certificate. To fix this issue copy back your saved keystore, adkeystore.dat and adsign.txt file from Step 4.5.4. Backup your Completed Files then regenerate your jar files.

13. Unsigned Jars show access denied errors (sun.awt, sun.java2d)

Running jar files using a Self-Signed Certificate with older algorithm standards (e.g. SHA1withDSA) and JRE releases (JRE 7u121 and 6u131 and lower) can display the following errors in the Java Console window:

> java.security.AccessControlException: access denied (java.lang.RuntimePermission accessClassInPackage.sun.awt)
> java.security.AccessControlException: access denied (java.lang.RuntimePermission accessClassInPackage.sun.java2d)

Note: Oracle recommends that Jar files are signed using a Trusted Certificate Authority as covered within this document and that Java content is accessed using the latest and therefore most secure JRE version. If you are unable to to sign your jar files or upgrade at present you can workaround the issue by following the steps below.

1. Your Java keystore (adkeystore.dat) should be using the current default security standards which are 2048-bit (minimum), SHA2 with an RSA key, for example:

> Signature algorithm name: SHA256withRSA
> Subject Public Key Algorithm: 2048-bit RSA key

You can check these values in your current keystore by running the following command against your adkeystore.dat file:

```
$ cd <JRI_DATA_LOC>
$ keytool -list -v -keystore adkeystore.dat | grep lgorithm
```

2. To update your keystore check you have applied the AD patches as specified in Step 2.2. AD Patch Requirements.
3. Ensure you are using JDK 6 update 18 or higher or JDK 7 on your application tier for SHA-2 support.
4. Generate a new keystore as specified in Step 3.2. Generate a new keypair (private key and public key.

5. Regenerate your jar files with the force option, see Section 5: Regenerate the Jar Files and bounce the application tier services.

PKCS #12 Certificates (*.pfx, *.p12), Comodo

Comodo deliver a PKCS #12 style certificate which is one of the Public-Key Cryptography Standards (PKCS) and use the extensions *.pfx or *.p12. As such you will not receive your certificate signing request (*.csr) back in the appropriate format to import it as outlined in Section 4: Import your Certificate(s). The instructions below will walk you through the process of creating a new Java Keystore allowing you to integrate this certificate type into your Oracle E-Business Suite environment, enabling you to securely sign your jar files.

After completing the initial steps within this document up to and including Section 3: Generate Keypair and Certificate Signing Request continue with the steps below.

Step C1. Copy your PKCS #12 certificate into your Jar Signing Files Directory

Once you have received the PKCS #12 certificate (*.pfx or *.p12 file) from your provider, source your environment and copy it to your Jar signing files directory:

1. Source your environment Step 3.1. Source the Environment
2. Change directory to your Jar Signing Files Directory. '***<JRI_DATA_LOC>***' and copy your PKCS12 certificate file into it.
3. Move your existing `<JRI_DATA_LOC>/adkeystore.dat` Java keystore to a back up, for example:

```
$ mv adkeystore.dat adkeystore.dat.backup
```

Step C2. Ascertain the Alias and Passwords

To run this process you will require the passwords and alias values that are set within your EBS environment along with the PKCS #12 certificate name (e.g. cert.pfx or cert.p12) as well as the private key password `<PKCS12_passwd>` for your PKCS #12 certificate file.

Identify and note the current Alias and Keystore passwords on your EBS environment by running the commands in Appendix A: Identifying and Changing the Passwords and Alias. You should now have a list of values for these various parameters. To identify these in the steps below we will use the following names outlined in bold:

```
PKSC #12 Certificate Information:

Certificate Name = <PKCS12_cert>
Store Password = <PKCS12_passwd>

EBS Information:

Alias = <ebs_aliasname>
Store Password = <ebs_storepasswd>
Key Password = <ebs_keypasswd>
```

Step C3. Convert your PKCS #12 Certificate to Java Keystore format (PKS)

Convert your PKCS #12 (*.pfx or *.p12) certificate file to a new keystore named, `adkeystore.jks` and assign the EBS expected Store Password `<ebs storepasswd>` at the same time by running the command below, replacing the values in bold as appropriate:

```
$ keytool -importkeystore -destkeystore adkeystore.jks -deststoretype jks -srckeystore
<PKCS12_cert> -srcstoretype pkcs12

  Enter destination keystore password: <ebs_storepasswd>
  Re-enter new password: <ebs_storepasswd>
  Enter source keystore password: <PKCS12_passwd>
```

Step C4. Change PKCS #12 Alias to the EBS Alias Value

Change the alias name in the new keystore `adkeystore.jks` to match the alias name used in your EBS environment `<ebs_aliasname>`.

You will first need to verify the alias `<PKCS12_aliasname>` in the new keystore `adkeystore.jks` by running the following command, entering your `<ebs_storepasswd>` when requested. The alias should be displayed under the entry titled `Alias name:` near the top of the output:

```
$ keytool -list -v -keystore adkeystore.jks

  Enter keystore password: <ebs_storepasswd>

Output Example:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: <PKCS12_aliasname>

etc.
```

Change the alias `<PKCS12_aliasname>` in the new keystore `adkeystore.jks` to match the alias used in your EBS environment `<ebs_aliasname>` by running the command below, replacing the values in bold as appropriate:

```
$ keytool -changealias -keystore adkeystore.jks -alias <PKCS12_aliasname> -destalias
<ebs_aliasname>

  Enter keystore password: <ebs_storepasswd>
  Enter key password for <PKCS12_aliasname>: <PKCS12_passwd>
```

Step C5. Change KeyPass to the EBS expected value

```
$ keytool -keypasswd -keystore adkeystore.jks -alias <ebs_aliasname>

  Enter keystore password: <ebs_storepasswd>
  Enter key password for <ebs_aliasname>: <PKCS12_passwd>
  New key password for <ebs_aliasname>: <ebs_keypasswd>
  Re-enter new key password for <ebs_aliasname>: <ebs_keypasswd>
```

Step C6. Copy the new Keystore

Copy the newly created Java keystore `adkeystore.jks` to the EBS Java keystore `adkeystore.dat`.

```
$ cp adkeystore.jks adkeystore.dat
```

Continue through the other steps in this document from Section 5: Regenerate the Jar Files to complete the process.

Appendix A: Identifying and Changing the Passwords and Alias

This section outlines the required steps to identify and if required change your keystore passwords and/or alias values.

Note: After changing the keystore passwords you must create a new keystore by running Step 3.2. Run adjkey to create a new keypair (Private key and public key) and then run through all the other steps in this document to create a new signed certificate.

- Identify and Change the Keystore Passwords
- Identify and Change the Alias

Identify and Change the Keystore Passwords

If required you can identify and change the keystore passwords by running the commands below.

Identify Your Keystore Passwords

To identify the existing keystore and key passwords run the following SQL script connected as the APPS user:

```
SQL> set serveroutput on
declare
spass varchar2(30);
kpass varchar2(30);
begin
ad_jar.get_jripasswords(spass, kpass);
dbms_output.put_line(spass);
dbms_output.put_line(kpass);
end;
/

This will output the passwords in the following order:

store password (spass)
key password (kpass)
```

Change Your Keystore Passwords

If required, you can change your Keystore passwords by running the two commands below and entering your new password when prompted. The new password must be at least six (6) chars long.

Change your keystore (store) password

Run the following command substituting the values in **bold** as applicable:

```
$ adjkey -storepasswd

  Enter the APPS username: <appsusername>
  Enter the APPS password: <appspassword>

  Enter the new keystore password: <new store passwd>
```

Change your key password

Run the following command substituting the values in **bold** as applicable:

```
$ adjkey -keypasswd

  Enter the APPS username: <appsusername>
  Enter the APPS password: <appspassword>

  Enter the new key password: <new key passwd>
```

Identify and Change the Alias

Identify Alias

Running the following command is one way to identify your current (default) **<alias>** which will be displayed before the date information in the output:

```
$ adjkey -list

Enter the APPS username: <appsusername>
Enter the APPS password: <appspassword>

Successfully created javaVersionFile.
alias name used is <alias>
.
.
```

The default `alias` value matches the `$CONTEXT_NAME` value of your Oracle E-Business Suite environment.

Change Alias

If you want to change your `alias` you can do so by applying the appropriate 'Change Alias Patch' as noted in Step 2.2. AD Patch Requirements. This allows you to define your new alias as a parameter when generating your new keypair as outlined in Step 3.2. Run adjkey to create a new keypair (Private key and public key).

Appendix B: Exception Site List and Deployment Rule Set Features

The 'Exception Site List' and 'Deployment Rule Set' features can to be used to create a 'whitelist' of applications that will be run without most of the security prompts. This would therefore continue to allow jar files to run through Oracle E-Business Suite that are not signed by a Trusted CA even on the highest Java security settings.

Exception Site List

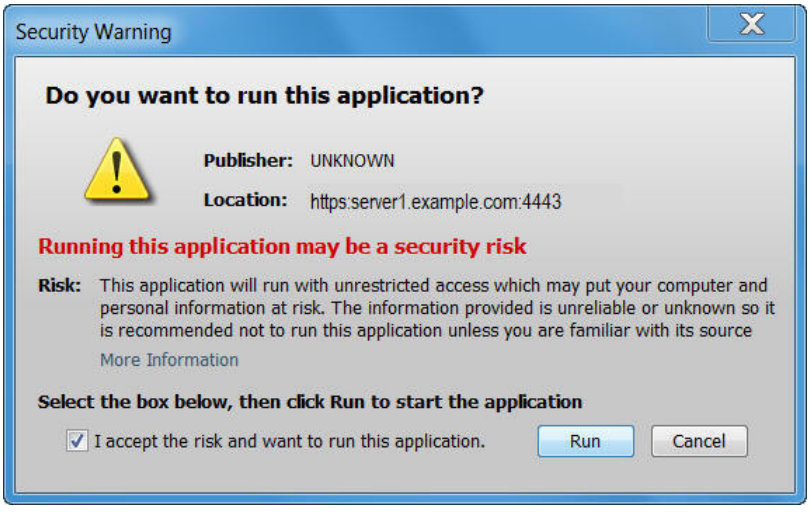
End Users running JRE 1.7.0_51 can utilize the Exception Site List feature from their desktop. This feature will allow jar files to run through Oracle E-Business Suite that are not signed by a Trusted CA even on the highest Java security settings.

To access E-Business Suite you could add your environment URL and port number for example:

```
https://myserver.example.com:port
```

For further information and examples on using this feature see the Java SE Exception Site List document.

When you access any Java content through the 'Exception Site List' you will see the following warning message appear each time you start a new session, requiring the user to agree to accept the risk before running.



Tick the checkbox and click the `'Run'` button to open the form.

Deployment Rule Set

JRE 1.7.0_ 40 or higher users can also utilize the Deployment Rule Set security feature. This feature will allow jar files to run through Oracle E-Business Suite that are not signed by a Trusted CA even on the highest Java security settings. However it's primary function is as an added security layer on users desktops used in conjunction with Oracle E-Business Suite environments running jars already signed by a Trusted CA.

The DeploymentRuleSet.jar file itself must be signed with a certificate from a Trusted CA, either an official CA (such as Verisign, Thawte etc.) or an in-house CA. Once you have a certificate it is therefore recommended that you firstly sign your Oracle E-Business Suite environments securely before further enhancing security through a Deployment Rule Set.

The following steps provide an example of how to create a DeploymentRuleSet.jar for Oracle E-Business Suite that only allows java content signed with your own unique valid certificate to run. For further information on this feature including, other ruleset examples and how to obtain your "certificate_hash" (as used in the example below) see the Deployment Rule Set Java SE Documentation.

The commands used assume you are signing your DeploymentRuleSet.jar within an Oracle E-Business Suite environment that is already signed by a Trusted CA which would then allow you to use it against other environments within your domain.

Note: The following example using the certificate hash can work for both Java Plug-in and Java Web Start set ups. Java Web Start requires that the "certificate alogorithm" value is included with the certificate hash as shown below..

Note: The following example using the "certificate hash" can work for both Java Plug-in and Java Web Start set ups. Java Web Start requires that the "certificate alogorithm" is included before the certificate hash as shown below.

1. Create Your Ruleset.xml File

Using any text editor, create your **ruleset.xml** file. :

```
<ruleset version="1.0+">
 <rule>
  <id>
   certificate algorithm="SHA-256" hash="[certificate_hash]" />
 </id>
  <action permission="run" version="SECURE" />
 </rule>
 <rule>
 </id>
  <sction permission="block">
   <message>Blocked by corporate.</message>
  </sction>
 </rule>
</ruleset>
```

Where *[certificate_hash]* is the 32 hexadecimal digit string of your code signing certificate.

2. Create the DeploymentRuleSet.jar

Incorporate your ruleset.xml file into the DeploymentRuleSet.jar, for example:

```
jar -cvf DeploymentRuleSet.jar ruleset.xml
```

3. Sign the DeploymentRuleSet.jar File

Sign your DeploymentRuleSet.jar file with the following command:

```
$ jarsigner -keystore <keystore location> <jar file> <alias>
```

For example:

```
$ jarsigner -keystore <JRI_DATA_LOC>/adkeystore.dat DeploymentRuleSet.jar <alias>

  Enter Passphrase for keystore: spass
  Enter key password for <alias>: kpass
```

4. Place the DeploymentRuleSet.jar File on the Desktop

Place the DeploymentRuleSet.jar File on the Desktop in the following directory as applicable to your desktop platform.

Windows Platforms

```
C:\Windows\Sun\Java\Deployment
```

Mac OS X Platforms

```
/etc/.java/deployment
```

Appendix C: Jar File Manifest Updates and Jar File Signing

JRE 1.7.0_25 (7u25) and higher include a security function requiring the inclusion of permissions and codebase attributes in the Jar Manifest file. Without these attributes warning messages may show in the Java Console and in future releases jar content may fail to launch when using the most secure security settings. These attributes will be included in all jar files administered through adadmin after the application of the appropriate AD prerequisite patch as listed in Section 2: Prerequisite Requirements, and the force regeneration of your jar files through 'adadmin'.

For information and patch fixes (including signed jars and permission attributes included) for jar files not administered through adadmin within Oracle E-Business Suite, see Account Hierarchy Manager & Financial Dimensions Hierarchy Manager (EBS 11i) and Oracle Learning Management (EBS R12)

Adding Additional Attributes to a Jar File Manifest

This section details how you can add additional attributes to Jar files that are administered through AD tools and therefby generated through ADADMIN.

Default Jar files that are administered through AD tools already contain the following attributes within the Manifest file:

```
Application-Library-Allowable-Codebase: *
Application-Name: Oracle E-Business Suite
Permissions: all-permissions
Codebase: *
```

Allowable attributes are explained in the JAR File Manifest Attributes for Security documentation.
If required, additional attributes can be added to a specific Jar file Manifest as outlined below.

Prerequisite Patch

Apply Patch 18283295:R12.AD.C.delta.5 or later by followig its readme.

Adding an Attribute

Jar file build instructions are held in a <product>jar.dep file. The attribute should be added within square brackets after the jar file name, before the colon (:) that separates the jar file name from the class folders. The following gives an example of adding the 'Caller-Allowable-Codebase' attribute to the 'fndcp.jar' file :

Back up your $FND_TOP/java/make/fndjar.dep file

Find fndcp.jar within the fndjar.dep file which should display as:

```
fndcp.jar :
        oracle/apps/fnd/cp
```

Add the attribute and its value:

```
fndcp.jar
        [Caller-Allowable-Codebase: host.example.com 127.0.0.1] :
        oracle/apps/fnd/cp
```

Generate your jar files through ADADMIN to recreate the jar file with the new attribute.

Manually Updating a Jar File Manifest in a 3rd Party Jar File

If you have a requirement to manually add these attributes to jar files that are not administered by ADADMIN, for example when using a custom jar file, follow the steps below by firstly updating the Jar file Manifest and secondly re-signing the Jar file. After making changes to the Jar file Manifest you **must** then sign the Jar file.

If you require further information see Modifying a Manifest File from The Java Tutorials pages.

The Jar Manifest Update Process

Step1: Take a backup copy of your jar file in case you need to revert to it at a later date.

Step 2: Create a text file named **manifest.txt** as shown below and place it in the same directory as your jar file:

```
Application-Name: Oracle E-Business Suite
Permissions: all-permissions
Codebase: *
```

Note: The manifest.text file must end with a new line or carriage return otherwise the last line will not be parsed properly.

Step 3: The new attributes from the manifest.txt file can then be added to the jar files existing Manifest by running the following command:

```
$ jar umf manifest.txt <jar file>
```

For example to update the Manifest of a jar file called MyJar.jar run the following command from the directory containing your jar file:

```
$ jar umf manifest.txt MyJar.jar
```

Step 4: After updating the Jar file Manifest, sign the Jar file by continuing through the Signing a Jar File section below.

Signing the Jar File

If you have a requirement to sign a jar file that is not administered through ADADMIN with your trusted CA, follow the steps below.

You will be required to enter your <alias>, <keystore> (spass) and <key> (kpass) passwords when running the command below. If you need to identify these values, see Appendix A: Identifying and Changing the Passwords and Alias

If you require further information see Signing Jar Files from The Java Tutorials pages.

The Jar Signing Process

Step1: Take a backup copy of your jar file in case you need to revert to it at a later date.

Step 2: A jar file can be signed using the following command, replacing the values in bold as applicable to your organization:

```
$ jarsigner -keystore <keystore location> <jar file> <alias>
```

For example to sign a jar file called MyJar.jar run the following command from the directory containing the jar file:

```
$ jarsigner -keystore <JRI_DATA_LOC>/adkeystore.dat MyJar.jar <alias>

  Enter Passphrase for keystore: spass
  Enter key password for <alias>: kpass
```

Verify the Digital Signature of a Signed Jar File

To verify the digital signature of the Jar file run the following command replacing the value in bold as applicable:

```
$ jarsigner -verify -verbose -certs <jar file>
```

For example to verify the digital signature of MyJar.jar run the following command from the directory containing the jar file:

```
$ jarsigner -verify -verbose -certs MyJar.jar
```

This should return a number of rows showing the certificate values matching those applicable to your organization, for example:

```
X.509, CN=Atoz Widgets Ltd, OU=Sales, O=Atoz Widgets Ltd, L=Madison City, ST=Missouri, C=US
[certificate is valid from 7/31/13 3:26 AM to 7/29/23 3:26 AM]
X.509, CN=Atoz Ltd. (Signing For Code), OU=CA-Atoz, O=Atoz, L=MDC, ST=MIS, C=US
[certificate is valid from 5/5/13 12:36 PM to 5/3/23 12:36 PM]
  etc.....

s = signature was verified
m = entry is listed in manifest
k = at least one certificate was found in keystore
i = at least one certificate was found in identity scope

jar verified.
```

Appendix D: Multiple Instances/Sharing a Digital Certificate

If you have multiple environments you do not need a new code signing digital certificate for each one. Only one code signing digital certificate is required to sign your jar files across multiple environments.

Once you have successfully completed signing your Oracle E-Business Suite instance with a Trusted CA ('Master' environment) you can share the same 'code signing digital certificate' across any other Oracle E-Business Suite environments ('Target' environments'). This can be achieved by copying the relevant files from your 'Master' environment to the 'Target' environment as outlined in this section.

Note: The ability to change the keystore passwords is one of the features introduced by the new AD patches as listed under Section 2: Prerequisite Requirements. (This patch must also be installed on your 'Target ' environments)

It is important to ensure that the keystore passwords on the Target environment, match those on the Master environment **before** copying in the files from the Master environment. If this step is missed see Multiple Instances Known Issues

Step 1. Check Key Passwords

Check that the the key passwords on the Target environment match the ones that are used on the Master environment. If required change the passwords to match those used on the 'Master' environment.

The key passwords can be checked and if required changed by following Identify and Change the Keystore Passwords.

Step 2. Back up and Copy Files

Step 2.1. Back up Files on 'Target' Environment

Back up the files listed below on the target instance substituting, for example:

EBS 12.2.x Users

```
cp $NE_BASE/EBSapps/appl/ad/admin/adsign.txt
$NE_BASE/EBSapps/appl/ad/admin/adsign.txt.presign
cp $NE_BASE/EBSapps/appl/ad/admin/adkeystore.dat
$NE_BASE/EBSapps/appl/ad/admin/adkeystore.presign
```

EBS 12.1.x, 12.0.x and 11i Users

```
cp $APPL_TOP/admin/adsign.txt $APPL_TOP/admin/adsign.txt.presign.
cp $APPL_TOP/admin/adkeystore.dat $APPL_TOP/admin/adkeystore.dat.presign
```

Step 2.2. Copy Files to the 'Target' Environment

Copy the files taken from your Master environment to the directory listed above on your Target environment as appropriate to your Oracle E-Business Suite release.

Step 3. In-House Certificate Authority Users

If you are using an In-House Certificate Authority your root certificate will not be included in the default Java `cacerts` file.

Step 3.1. Back up cacerts File on 'Target' Environment

Back up the existing `cacerts` file on your target environment as applicable to your Oracle E-Business Suite release.

Oracle E-Business Suite Release 12 Users

```
cp $OA_JRE_TOP/lib/security/cacerts $OA_JRE_TOP/lib/security/cacerts.presign
```

Oracle E-Business Suite Release 11i Users

```
cp $OA_JRE_TOP/jre/lib/security/cacerts $OA_JRE_TOP/jre/lib/security/cacerts.presign
```

Step 3.2. Import the Root Certificate into cacerts on the 'Target' Environment

Import the root certificate into cacerts on the target environment by following Step 4.4. Import the Root Certificate to the Java Keystore Certificate Store 'cacerts' (if required).

Step 4. Regenerate the Jar Files

Regenerate the jar files on the Target instance using the 'force' option through adadmin, see Section 5: Regenerate the Jar Files.

Step 5. Install the root certificate into the Java or Browser certificate store

The Java or browser certificate store will already contain many root certificates from recognized Certificate Authorities.

If you are using an In-House Certificate Authority your root certificate will not be included by default in the Java or Browser certificate store, install it by following Section 6: Desktop Client Requirements.

Multiple Instances Known Issues

After copying all the files, regenerating the jar files may end in the error :

```
adogjf() Unable to generate jar files under JAVA_TOP
```

This may have occured because the keystore passwords on the Master and Target environments did not match before the files were copied. Trying to change the passwords after putting the files into the Target environment will error because they are out of sync:

```
keytool error: java.io.IOException: Keystore was tampered with, or password was incorrect
```

To fix this issue run `adjkey -initialize` on the Target environment to create a new keystore, then reset the passwords to match those on the Master environment by following Identify and Change the Keystore Passwords. Once corrected continue through this process again from Step 2.2. Copy Files to the Target Environment.

(The values used when running `adjkey -initialize` are not important as the files will be overwritten again with those from the Master environment.)

Appendix E: Advanced Jar Signing

This section provides a high level overview for experienced users outlining the basic general requirements for advanced jar signing. This refers to jar signing without using the standard AD tools that are provided within Oracle E-Business Suite..

You may have requirements where you cannot sign your jar files using the standard AD tools as outlined within this document, for example:

- You have your own centrally managed in-house software certificate signing system that cannot incorporate the EBS keystore (adkeystore.dat) or does not allow the EBS admin access to the keys.
- You are using an independent keystore provided by your Certificate Authority e.g. through a USB Token.
- You have other requirements that are currently not covered by the default AD signing system (e.g. Timestamping).

Note: The setting up and incorporation of the many permutations of these requirements are outside the scope of this document and the usual support channels. If you require assistance in setting this up to your individual requirements you should contact consultancy.

AD Keystore (adkeystore.dat)

The AD keystore (`<JRI_DATA_LOC>/admin/adkeystore.dat`) is used for signing the jar files when regenerating them through ADADMIN. If signing your jars using your own keystore, adkeystore.dat is no longer required in your signing process. If jar files are regenerated through the application of a patch within EBS for example then the jar files will be generated and signed using adkeystore.dat which will not hold the correct information. Ensure you add a step to your internal processes to re-sign the jar files using your own keystore after AD tools has been used to regenerate the jar files.

How do I know what jar files need signing from my EBS environment?

When signing your jar files externally you will need to run your jarsigner command against each jar file that is being used within your EBS environment. Regenerating jar files through ADADMIN with the 'force' option outputs a list of all the jar files used on that environment to a file named jarlist.txt. This is available from the following locations:

EBS 12.2

`$<NE_BASE>/EBSapps/log/adadmin/log/jarlist.txt`

EBS 11i & 12.1

`$APPL_TOP/admin/$TWO_TASK/log/jarlist.txt`

Signing Jar Files with jarlist.txt

The jarlist.txt file can be incorporated into a script to run through the signing procedure using jarsigner, incorporating any specific requirements as needed.

The shell script below gives a basic example of how you can utilize the jarlist.txt file to sign your jar files outside of AD. This can be modified for individual requirements. The example below will:

- Read each jar file from the jarlist.txt file
- Remove the current signature files that are included when generating the jars within EBS through ADADMIN
- Sign each jar file using the certificate within the external keystore

It is strongly recommended that sensitive information such as the store passwords and alias values are either picked up from the EBS server (an example of this is in the second script below) or from some other file source so that these values are not stored directly in the script.

```
# Code signing script to sign client JARs using jarlist.txt

adjarlist="<path to jarlist.txt>"
zurl="<path to your keystore>"
zalias="<keystore alias>"
zstorepass="<store password>"
zkeypass="<key password>"

# Select the jar files from jarlist.txt

jars_to_sign=`cat $adjarlist | grep '\.jar'`
```

```
for jar in ${jars_to_sign}
do

# Remove Signature from Jar files created through ADADMIN in EBS

echo " ** Removing EBS signature from: ${jar} "

zip -d ${jar} 'META-INF/*.SF' 'META-INF/*.RSA'

# Sign the jar files

echo " ** Signing: ${jar} "

jarsigner -keystore ${zurl} -storepass ${zstorepass} -keypass ${zkeypass} ${jar} ${zalias}

done
```

A similar approach can be used for adding any extra parameter requirements. The example below is adding timestamping to the jar files using the standard EBS keystoreadkeystore.dat. This script is a basic example and can be modified as required.

In general you will need to go through a firewall to access the internet, therefore you must include the proxy host and port values in order to access the server when signing your jars with a timestamp. This example script is using http for the proxyHost and proxyPort variables: `-J-Dhttp.proxyHost & -J-Dhttp.proxyPort`. If using https then the values should be changed accordingly to: `-J-Dhttps.proxyHost & -J-Dhttps.proxyPort`

```
# Code signing script to sign client JARs using jarlist.txt with a timestamp

# Prompt for the APPS user database password

printf "Enter the password for the Oracle database APPS user [APPS] : ";
read APPS_PWD

if test "${APPS_PWD}" = "" ; then
APPS_PWD="APPS";
fi

# Get the store passwords and alias values from apps

passwds=`sqlplus -s -l <<EOF
apps/${APPS_PWD}
set serveroutput on
declare
spass varchar2(30);
kpass varchar2(30);
begin
ad_jar.get_jripasswords(spass, kpass);
dbms_output.put_line(spass);
dbms_output.put_line(kpass);
end;
/
exit
EOF`

stringarray=($passwds)
STOREPASS=${stringarray[0]}
KEYPASS=${stringarray[1]}

IFS=' ' read -a array < "<JRI_DATA_LOC>/adsign.txt"
ALIAS="${array[0]}"

# Add required parameter values

adjarlist="<path to jarlist.txt>"
zurl="<JRI_DATA_LOC>/adkeystore.dat"
ztsaurl="<URL of the timestamp server>"
zproxyhost="<the proxy host>"
zproxyport="<the proxy port>"

# Select the jar files from jarlist.txt

jars_to_sign=`cat $adjarlist | grep '\.jar'`

for jar in ${jars_to_sign}
do
echo " signing ${jar} "

# Sign the jar files with a timestamp

jarsigner -keystore ${zurl} -storepass ${STOREPASS} -keypass ${KEYPASS} -tsa ${ztsaurl} -J-
Dhttp.proxyHost=${zproxyhost} -J-Dhttp.proxyPort=${zproxyport} ${jar} ${ALIAS}
```

```
done
```

One way to see the timestamp in a jar file is to run the following command (using fndforms.jar as an example):

```
$ jarsigner -verify -verbose -certs fndforms.jar | grep signed

    [entry was signed on 7/14/16 11:50 AM]
```

Appendix F: Obtaining Support

If you encounter any issues, log a service request against Product ID 1745 and include the following information:

1. The output of the TXK Inventory Report for the application tier, which can be generated using the following command:

```
$ $ADPERLPRG $FND_TOP/patch/115/bin/TXKScript.pl \
   -script=$FND_TOP/patch/115/bin/txkInventory.pl \
   -txktop=$APPLTMP -outfile="$CONTEXT_NAME"_AT.html

$ Enter APPS password ? <appspassword>
```

2. The contents of the following files:

```
Your 'Java code signing certificate' for example, adkeystore.crt
adsign.txt
```

Appendix G: Frequently Asked Questions

Will the MD-5 Changes in April 2017 affect Oracle E-Business Suite Users?

Starting with the April Critical Patch Update releases, planned for April 18 2017, all JRE versions will treat JARs signed with MD-5 as unsigned.

In general this should have no impact on EBS users.
Recognized trusted CA's stopped providing certificates using the MD5 algorithm a number of years ago. These were replaced with certificates providing the SHA-1 algorithm which in turn have since been replaced by certificates with the SHA-2 algorithm which is the currently provisioned standard.

If you need to check what algorithm is being used to sign your jars, run the following command on your keystore:

```
keytool -list -v -keystore adkeystore.dat | grep "Signature algorithm name"
Enter keystore password: <keystore password>

In most cases this will should show that SHA-2 is being used, returning something
similar to:

Signature algorithm name: SHA256withRSA
```

If this returns MD5 it is strongly recommended that you upgrade your certificate to SHA-2 as soon as possible.
If you are still running SHA-1 (which is deprecated) it is also recommended to plan an upgrade to SHA-2

Can I sign Jar files with an SSL certificate?

No, SSL and code signing certificates use a different key paring mechanism to differentiate them.

An SSL certificate is used for authentication of your web applications server and has no code signing capabilities. A code signing certificate is specifically designed for code level authentication.

SSL Wildcard/Domain certificates are also available negating the need to have a separate certificate for each server by allowing the use of a domain. For the same reasons these also cannot be used for code signing. A code signing certificate defines a code signing authority which can be used across as many applications as you choose anyway.

Can I obtain a certificate Free of Charge?

We are not aware of any third party certificate providers that supply code signing certificates free of charge.

If all of your clients are internal, using an in-house CA is an option if have a mechanism to have those clients trust the in-house CA. This is possible with free to use OpenSSL code for example. Although this can often be done after careful planning and consideration on a case by case basis purchasing a certificate is a simpler and more robust option.

Why are E-Business Suite Jar files created and signed at the customer site instead of at Oracle?

E-Business Suite jar files are created and signed at the customer site in order to facilitate granular patching of E-Business Suite.

When Oracle releases a patch containing a fix to E-Business Suite code, we deliver the file that contains that fix, along with any prerequisite files that may be needed. These prerequisites may include E-Business Suite middle-tier and database objects. If we were to bundle java code fixes into jar files at Oracle, we would also need to include the latest versions of all other java classes belonging to the same jar file plus all the prerequisites of those other classes. It is likely that additional middle-tier and database changes would need to be included in the patch in order to satisfy prerequisites, resulting in a larger patch requiring more testing. Instead, we only deliver the relevant files for a given fix. The E-Business Suite patching utilities then bundle the fixed java code into jar files at the customer site and manage the signing of those jar files. Since the patch impacts fewer files, less testing is required after applying the patch at the customer site.

Is the signing of Jar files at the customer site a new requirement?

E-Business Suite has always used a granular patching approach for java code, with the jar files being created and signed at the customer site. Historically, customers could choose to use either a "self-signed" certificate or a certificate from a trusted certificate authority to sign their jar files. Recent versions of Java have tightened the default security settings, meaning that the Forms applet can no longer be launched if the jar files are signed with a "self-signed" certificate. Accordingly, customers are advised to sign jar files using a certificate from a trusted certificate authority.

What must I do when upgrading my EBS environment?

Always ensure you have a saved back up copy of your signed adkeystore.dat and adsign.txt files from your existing environment. While in general these should not be required, if they do happen to get overwritten when preforming an upgrade or other maintenance work you can simply add these back into the new environment and force regenerate the jar files so that they are signed with the same certificate. These files can also be shared between different EBS releases. Further information on using certificates in multiple environments is detailed under Appendix D: Multiple Instances/Sharing a Digital Certificate . Also see Section 2: Prerequisite Requirements for any additional patch requirements.

How do I renew my Certificate?

Most certificate providers will require a new Certificate Signing Request (CSR) when renewing an expired certificate. Generating a new keypair is a quick and simple process which can also be beneficial in uptaking the latest recommended keysize and algorithm values if required. When renewing, follow the entire signing process again:

- Section 3: Generate Keypair and Certificate Signing Request
- Section 4: Import your Certificate(s)
- Section 5: Regenerate the Jar Files
- Section 6: Desktop Client Requirements

Note: Generating your keypair will create new adkeystore.dat and adsign.txt files. Therefore you may want to ensure you still have a back up of the current version of these files before regenerating your keypair.

Related My Oracle Support Knowledge Documents

- Oracle E-Business Suite with Java Web Start (Document 2188898.1)
- Deploying JRE (Native Plug-in) for Windows Clients in Oracle E-Business Suite 11i (Document 290807.1)
- Deploying JRE (Native Plug-in) for Windows Clients in Oracle E-Business Suite Release 12 (Document 393931.1)
- Using J2SE Version 6 with Oracle E-Business Suite 11i (Document 401561.1)
- Using Latest Java 6.0 Update With Oracle E-Business Suite Release 12 (Document 455492.1)

- Using JDK 7.0 Latest Update with Oracle E-Business Suite Release 12.0 and 12.1 (Document 1467892.1)

Further Information

- The Java Tutorials
- keytool - Key and Certificate Management Tool
- jarsigner - JAR Signing and Verification Tool

Change Log

Date	Description
Aug 31, 2018	Added Unsigned Jars show access denied errors (sun.awt, sun.java2d) to Known Issues.
Mar 17, 2017	Added Patch 17726123 to Step 2.2. AD Patch Requirements for EBS 11i users.
Feb 15, 2017	Added Will the MD-5 Changes in April 2017 affect Oracle E-Business Suite Users? to Appendix G: Frequently Asked Questions
Oct 18 , 2013	Initial Document.