# Java Keystores

# Video Overview:



The following procedure applies to Tomcat and Jboss:

Please NOTE:

* for Jboss Domain (legacy) you need to use a 'keytool' command on one of the old jboss domain nodes, otherwise the cert format is not recognized on these old hosts.
* for Tomcat keystore same applies.
* for Jboss Standalone and Springboot you can compile the keystore on your workstation or rather on a Jboss Standalone node.
* I'm  not exactly sure why that is, seems to me it's a version of keytool matter, but not sure, couldn't spare time to find out why.

Request the Certificate and Key from an InfoSec or Universal resource. You will need to request they download the file in a PEM OpenSSL format, with separate key and cert files:

NOTE:

If you receive a key/cert from Venify, you will need to decrypt it before use:

Use the following command and the supplied password (given to you when the certificate was given) to decrypt the files:

###

openssl rsa -in <original encrypted file name> -out <name of new file for decrypted file>

###

e.g.:

openssl rsa -in qa1\_wildcard\_cert\_2022\_Key.key\_encr -out qa1\_wildcard\_cert\_2022\_Key.key

**You will then be prompted for the encryption key. Input this, and your decrypted file should be generated and can be used in the following steps**

To properly create a keystore from scratch

First make a copy of the cert using the setalias command which will force the friendlyName attribute

openssl x509 -in \_.allegiantair.com.crt -setalias "g4\_wildcardcert" -out \_.allegiantair.com.alias.crt

Next, convert the PEM format to pkcs12, which bundles everything together (provide password: changeit).

openssl pkcs12 -export -out \_.allegiantair.com.pk12 -name g4\_wildcardcert -caname g4\_wildcardcert -inkey \_.allegiantair.com.key -in \_.allegiantair.com.alias.crt -certfile gd\_bundle-g2-g1.crt

Then, you must convert the pkcs bundle to a Java Keystore

keytool -importkeystore -deststorepass changeit -destkeypass changeit -destkeystore jboss.keystore.wcz -srckeystore \_.allegiantair.com.pk12 -srcstoretype pkcs12 -srcstorepass XXXXXXXXXX -alias g4\_wildcardcert -noprompt

You can verify the single entry with the alias of g4\_wildcardcert

keytool -list -keystore [keystore name]

You can then verify the entirety of the new keystore

keytool -list -keystore [keystore name] -v

It should show the private key, host cert, and CA cert  
  
Passwords mentioned in the commands can be derived from the java\_keystore automation in SS&M. Unless overridden, the default is changeit.  This process can probably be done without them but I maintained the existing passwords in case JBoss and Tomcat have the passwords configured in their XML files.  
  
Once complete and verified, the keystore can be added to automation with the prior mentioned steps:  
  
    Base64 encode the file:

openssl enc -base64 <[filename of the keystore]>[new filename]

    Add the variable you want the contents assigned to the top of the newly encoded file, usually:

---

variable\_name: |

encoded lines are here....

    You MUST add 2 spaces before each of the encoded lines or ansible will not read the file into the variable  
  
You can then vault the file with ansible.

## ****JBoss Keystores****

Jboss domain keystores are located at **/etc/jbossas/domain/jboss.keystore.**

This keystore is required for the HCs to be able to connect to the DC.

Once you have created the above keystore, you can drop this file into the above location **(name must remain jboss.keystore).**

## ****Springboot Keystores****

Springboot Keystores use the same keystore as jboss. On the springboot hosts, this file is located at **/etc/pki/java/springboot.keystore.**

This keystore is used for the springboot microservices to contact one another directly.

The jboss.keystore file for a given environment can be dropped into this location on a given springboot server **(name must remain springboot.keystore)**.Afterwards, any springboot services would need to be restarted for the updated keystore to take effect.

## ****Flydesk Keystores****

**IMFWS** Flydesk keystores are locatedat **/usr/share/java/tomcat**

The keystore file for a given environment can be dropped into this location on a given server **(name must remain keystore)**.Afterwards, Flydesk (service name: flydesk.target) service would need to be restarted for the updated keystore to take effect.

## ****INFlight Keystores****

**IMFWS (InFlight Flydesk)** keystores are located at **/usr/share/java/tomcat7**

The keystore file for a given environment can be dropped into this location on a given server **(name must remain keystore)**. Afterwards, it is easier to just **restart the host** so Tomcat and the 4 Flydesk services (admin, discovery, edge, and gatekeeper) will start up in the correct order.

**IMOWS (InFlight mobile)** does not appear to have a keystore.

## ****Jenkins Keystores****

Jenkins keystores are stored in /var/lib/jenkins/.keystore and have slightly different requirements.  There are two entries with two aliases in this keystore: "1" and "internal\_ca".  The "1" alias is reserved for the certificate and the "internal\_ca" for the CA bundle cert.

Assign the "1" alias to the cert:

openssl x509 -in \_.trn.allegiantair.com -setalias "1" -out \_.trn.allegiantair.com.alias.crt

After gathering the certificate, CA bundle, and private key, bundle them all together:

openssl pkcs12 -export -out \_.trn.allegiantair.com.pk12 -name 1 -caname internal\_ca -inkey \_.trn.allegiantair.com.key -in \_.trn.allegiantair.com.alias.crt -certfile gd\_bundle-g2-g1.crt

Now, create the keystore from the pkcs12 bundle (note: the password used will be the same for all fields and can be found in the "ps -ef | grep jenkins" output on the host):

keytool -importkeystore -srckeystore <path-to-bundle-file.pk12> -srcstoretype pkcs12 -destkeystore jenkins.example.com.jks -deststoretype JKS

Finally, add the second entry for the "internal\_ca" alias:

keytool -import -alias internal\_ca -file gd\_bundle-g2-g1.crt -keystore jboss.keystore.trn.jenkins