**SSL Certificates:**

**Reference site:** https://www.codebyamir.com/blog/java-developers-guide-to-ssl-certificates

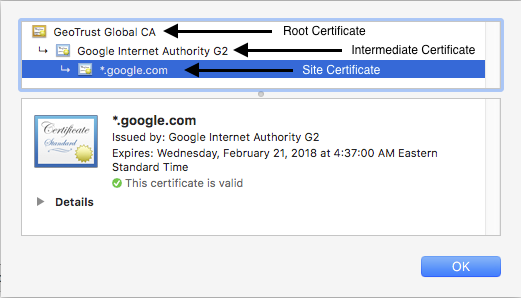
***Overview***

When developing web applications, we often need to integrate with other applications using SSL.   This could be over different protocols such as HTTPS, IMAPS, or LDAPS.

In this article, we'll cover what Java developers need to know about SSL certificates.

***Certificate Chain***

An SSL connection succeeds only if the client can trust the server.   Let's take a look at how this trust model works. In Chrome, go to google.com and bring up the Developer Tools (*F12* on Windows, *Cmd+Option+i* on Mac). Under the Security tab, click the View Certificate button to show details about the certificate.   We can see that the site certificate is part of a chain.   This particular chain consists of 3 certificates.



The site certificate has been issued by a certificate named *Google Internet Authority G2*.   This is the intermediate certificate.    In turn, the intermediate certificate is issued by the root certificate*GeoTrust Global CA*.

When we establish a connection over HTTPS, the web server will respond by providing its site and intermediate certificates.   It is then up to the client to complete the chain by having the root certificate.    This chain validation is necessary for the client to trust the site.

Since Chrome has the root certificate *GeoTrust Global CA* in its certificate store, our connection succeeds and we are not presented with any errors or warnings.

*Note: There may be more than one intermediate certificate in the chain depending on the site.*

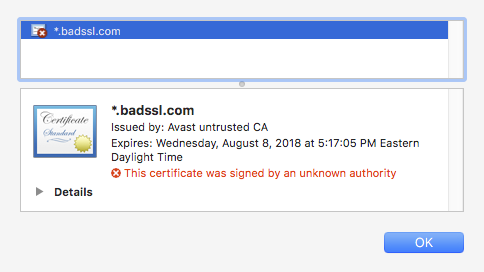
***Self-signed Certificates***

Certificates not issued by known CA but rather by the server hosting the certificate are called self-signed.

These are often used in internal development environments that are not customer facing.

The root certificates for these will be absent in the browser's certificate store.

An example of self-signed certificate is at https://self-signed.badssl.com.   We can see that this was issued by *Avast Untrusted CA* which the browser does not recognize so it displays a warning.



# *Java Truststore & KeyStore*

In this section, we'll discuss where certificates live on a system where the JDK/JRE is installed.

## **Truststore**

The **truststore** is a file that contains the root certificates for Certificate Authorities (CA) that issue certificates such as GoDaddy, Verisign, Network Solutions, and others.

The truststore comes bundled with the JDK/JRE and is located in $JAVA\_HOME/lib/security/cacerts.

The truststore is used whenever our Java code establishes a connection over SSL.

## **Keystore**

The **keystore** is a file used by an application server to store its private key and site certificate.

So if we were running a web application over SSL at tomcat.codebyamir.com, the keystore file named keystore.jks would contain two entries - one for the private key and one for the certificate.

The keystore is used by Java application servers such as Tomcat to serve the certificates.

Note: Most Java application servers only read the contents of these files during startup.   This means that any updates to the file require a restart to take effect.

# *Keytool*

Keytool is a utility bundled with the JRE for managing key pairs and certificates.    This allows us to view/modify/create certificate stores in the Java world.

## **List the certificates in the truststore**

keytool -list -keystore $JAVA\_HOME/lib/security/cacerts

We'll be prompted for a password for the truststore.   The default password is "changeit".

This truststore contains 104 entries and each entry has a unique alias and fingerprint.  We've truncated the output below for brevity.

Keystore type: JKS

Keystore provider: SUN

Your keystore contains 104 entries

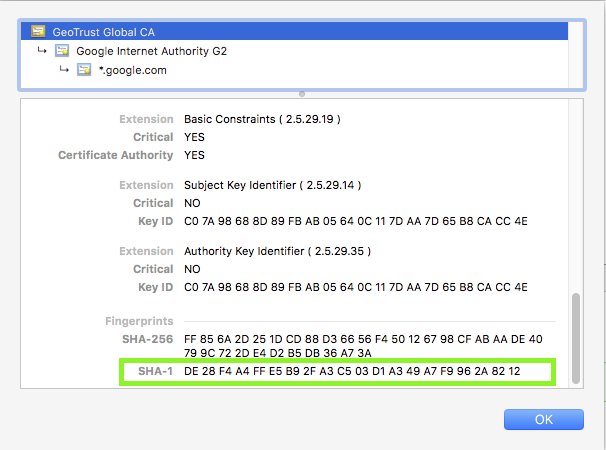
verisignclass2g2ca [jdk], Aug 25, 2016, trustedCertEntry,

Certificate fingerprint (SHA1): B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D

digicertassuredidg3 [jdk], Aug 25, 2016, trustedCertEntry,

...

Using the google.com example from before, let's take a look at the fingerprint for the GeoTrust Global CA from our browser:



Root CA certificate for google.com

The SHA-1 fingerprint is DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12

Let's look for that in our truststore:

keytool -list -keystore $JAVA\_HOME/lib/security/cacerts | grep -B1 -i DE:28

Enter keystore password: changeit

geotrustglobalca [jdk], Aug 25, 2016, trustedCertEntry,

Certificate fingerprint (SHA1): DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12

The output tells us that the certificate is in the truststore.

### **What does this mean to a Java developer?**

It means that code connecting to https://www.google.com won't throw an exception due to an SSL handshake error.

## **Add a certificate to the truststore**

Adding a certificate to the truststore is necessary if we want to trust a certificate issued from a CA not present in the bundled truststore.

keytool -import -trustcacerts -file [certificate] -alias [alias] -keystore $JAVA\_HOME/lib/security/cacerts