Certificate Validation for Secured Servers

For Repertoire Server configurations that are using SSL or HTTPS setups, certification validation is required when the application makes HTTP calls using the REST API. When the application makes a call to the secured server, the certificate validation process begins where a search for a commercially signed certificate from a certification authority is carried out. Without a commercial certificate, the following error is thrown:

javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target

The following code fragment provided explains how to use a self-signed certificate for the SSL connection. Run this code fragment only once. Copy the created "jsscacerts" file created to the "/lib/security" folder of the JVM installation that is running the application. Download the full utility class code here.

More information can found here: http://blogs.sun.com/andreas/entry/no_more_unable_to_find

```
//Creates a file called "jsscacerts" which will be copied
//to the directory where the code was run
File file = new File("jssecacerts");
if (file.isFile() == false) {
    char SEP = File.separatorChar;
    File dir = new File(System.getProperty("java.home") + SEP
        + "lib" + SEP + "security");
    file = new File(dir, "jssecacerts");
    if (file.isFile() == false) {
    file = new File(dir, "cacerts");
    }
}

//Loads in the keystore with a passphrase
System.out.println("Loading KeyStore " + file + "...");
InputStream in = new FileInputStream(file);
KeyStore ks = KeyStore.getInstance(KeyStore.getDefaultType());
ks.load(in, passphrase);
in.close();
```

```
//Creates a socket instance and attempts to connect
//to the server
SSLContext context = SSLContext.getInstance("TLS");
TrustManagerFactory tmf =
    TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
tmf.init(ks);
X509TrustManager defaultTrustManager = (X509TrustManager)tmf.getTrustManagers()[0];
SavingTrustManager tm = new SavingTrustManager(defaultTrustManager);
context.init(null, new TrustManager[] {tm}, null);
SSLSocketFactory factory = context.getSocketFactory();

System.out.println("Opening connection to " + host + ":" + port + "...");
SSLSocket socket = (SSLSocket)factory.createSocket(host, port);
socket.setSoTimeout(10000);
try {
    System.out.println("Starting SSL handshake...");
    socket.startHandshake();
    socket.close();
    System.out.println();
    System.out.println("No errors, certificate is already trusted");
} catch (SSLException e) {
    System.out.println();
    e.printStackTrace(System.out);

//Prints out the certificate used by the Repertoire Server
X509Certificate[] chain = tm.chain;
if (chain == null) {
    System.out.println("Could not obtain server certificate chain");
    return;
}

BufferedReader reader =
    new BufferedReader(new InputStreamReader(System.in));

System.out.println();
System.out.println("Server sent " + chain.length + " certificate(s):");
System.out.println();
MessageDigest sha1 = MessageDigest.getInstance("SHA1");
MessageDigest md5 = MessageDigest.getInstance("MD5");
for (int i = 0; i < chain.length; i++) {
        X509Certificate cert = chain[i];
        System.out.println
            (" " + (i + 1) + " Subject " + cert.getSubjectDN());
        System.out.println("   Issuer  " + cert.getIssuerDN());
        sha1.update(cert.getEncoded());
        System.out.println("   sha1    " + toHexString(sha1.digest()));
        md5.update(cert.getEncoded());
        System.out.println("   md5     " + toHexString(md5.digest()));
        System.out.println();
    }

    //Select the appropriate certificate at the command prompt
    //to be added to "jsscacerts"
    System.out.println("Enter certificate to add to trusted keystore or 'q' to quit: [1]");
    String line = reader.readLine().trim();
    int k;
    try {
        k = (line.length() == 0) ? 0 : Integer.parseInt(line) - 1;
    } catch (NumberFormatException e) {
        System.out.println("KeyStore not changed");
        return;
    }

    X509Certificate cert = chain[k];
    String alias = host + "-" + (k + 1);
    ks.setCertificateEntry(alias, cert);

    OutputStream out = new FileOutputStream("jssecacerts");
    ks.store(out, passphrase);
    out.close();

    System.out.println();
    System.out.println(cert);
    System.out.println();
    System.out.println
        ("Added certificate to keystore 'jssecacerts' using alias '"
        + alias + "'");
    }
```

Once the code is run, retrieve the "jsscacerts" from the directory where the Java utility class was run. Copy the "jsscacerts" to the "\lib\security" folder of the JVM running the application.