Progeny 10

Generating SSL Certificates for Progeny Web

Generating SSL Certificates for Progeny Web

Customer Support

Support is available for Support Plan members who purchase Progeny Clinical, Progeny Lab, or Progeny LIMS and that have an annual support agreement and in addition to trial users. Contact Progeny Genetics, LLC at:
1 Enterprise, Aliso Viejo, CA 92656
561-859-0773 International 800-776-4369 (US/CAN)
support@progenygenetics.com
http://www.progenygenetics.com

Generating SSL Certifications for Progeny Web

1. Open an elevated command prompt with administrative permissions.
2. Type **path** and hit enter.
3. Look at the output to see if it includes a reference to the bin folder within the 64-bit version of the Java application. If it does, move to step 4. If not, you will need to enter the path for the exact Java bin location. See an example below:
    - path=%path%;C:\Program Files\Java\jre1.8.0_231\bin
4. Create a folder in C:\ (name does not matter) then in your elevated command prompt target that folder.
5. Run the following script to generate the keystore file. You will need to replace {CERTNAME} with whatever name you want to use for the certificate, {ALIAS} with a common-name alias for the certificate and {PASSWORD} with whatever password you want to use for the keystore.

    keytool -genkeypair -keyalg RSA -keystore {CERTNAME}.jks -alias {ALIAS} -storepass {PASSWORD}

6. You will then be prompted for six pieces of information. The following table will detail what information is required for each prompt.

What is your first and last name?	Common Name (URL) of the domain/subdomain to be secured.
What is the name of your organizational unit?	Your group's name or department within the organization
What is the name of your organization?	Your company name (will be checked against the domain registration)
What is the name of your city or Locality?	City where the company is located (will be checked against the domain registration)
What is the name of your State or Province	Full name (not 2 letter code) of the State where the company is located (will be checked against the domain registration)
What is the two-letter country code for this unit?	US (if in the USA) or the code for your company's location
Is … correct?	Summarizes your previous entries, type 'yes' to confirm

7. When prompted for a password just hit enter (you have already created a password in Step 3).
8. Run the following script to generate your certificate request. You will replace {CERTNAME}, {ALIAS} and {PASSWORD} with the information used in Step 3. Replace {REQUESTNAME} with whatever name you want to use for the certificate request file.

    keytool -certreq -keystore {CERTNAME}.jks -alias {ALIAS} -storepass {PASSWORD} > {REQUESTNAME}.csr

9. Open the .csr file you just created and submit this information to your certificate authority.
10. Once you receive your validated SSL certificate, you will need to import this into your keystore. Save the files you receive from your certificate authority to the same folder where you generated the CSR and keystore.
    - You may receive multiple certificates from your authority, including a root and/or intermediate certificate.

- The certificates you receive need to be in text format to properly import into your keystore – acceptable certificate formats include .cer, .crt and .pem file extensions.

11. Repeat step 1 once you have the certificates saved to your SSL folder (the one created in Step 2), then run the following commands after changing the command prompt directory to your SSL folder:

    - Import the secondary certificates (Primary Root, Intermediate CA, Chain CA) – you may need to do this 2-3 times to import all the certificates received from your authority:

    ```
    keytool -import -trustcacerts -file {CERTIFICATE FILENAME}.cer -alias {CREATE NEW ALIAS} -keystore
    {CERTNAME}.jks –storepass {PASSWORD}
    ```

    - Import the server certificate:

    ```
    keytool -import -trustcacerts -file {CERTIFICATE FILENAME}.cer -alias {ALIAS FROM STEP 3} -keystore
    {CERTNAME}.jks –storepass {PASSWORD}
    ```

12. Once complete you will need to edit the server.xml file located within the **conf** folder of your Apache Tomcat installation (make sure the service is stopped before editing).

    - Uncomment the following XML string below in the server.xml file. Replace {KEYSTOREPATH} with the full file path to your keystore file and {PASSWORD} with the password you created in Step 2:

      NOTE: It is important to use a corresponding secure port – if your HTTP/1.1 port is 80 then your SSL port will be 443 and if your HTTP/1.1 port is 8080 then your SSL port will be 8443.

    Tomcat 9

    ```
    <Connector port="443" protocol="org.apache.coyote.http11.Http11NioProtocol" maxThreads="150"
    SSLEnabled="true" >
            <SSLHostConfig>
                    <Certificate certificateKeystoreFile="{KEYSTOREPATH}.jks"
                    certificateKeystorePassword="{PASSWORD}" type="RSA" />
            </SSLHostConfig>
    </Connector>
    ```

    TOMCAT 8

    ```
    <Connector port="443" protocol="org.apache.coyote.http11.Http11NioProtocol" maxThreads="150"
    SSLEnabled="true" scheme="https" secure="true" clientAuth="false" sslProtocol="TLS"
    keystoreFile="{KEYSTOREPATH}.jks" keystorePass="{PASSWORD}" />
    ```

13. Start the Tomcat service and test the connection to Progeny web manually using http:// and https://

    - If both connections are successful, then stop the Tomcat service and continue to the next step.
    - If either connection fails to display the login screen, stop and contact Progeny Support.

14. The last step is to configure Apache Tomcat to automatically redirect all HTTP requests to the SSL-secured port. Edit the web.xml file located within the conf folder of your Apache Tomcat installation. Go to the bottom of the document and add the XML below just before </web-app>. When complete start your Tomcat service and verify that the automatic redirection is working:

```
<security-constraint>
        <display-name>RPC Request Encryption</display-name>
        <web-resource-collection>
                <web-resource-name>RPCRequests</web-resource-name>
                <description>RPC Requests</description>
                <url-pattern>/*</url-pattern>
                <http-method>GET</http-method>
                <http-method>POST</http-method>
        </web-resource-collection>
        <user-data-constraint>
                <description>Encrypt all RPC request data destined for server</description>
                 <transport-guarantee>CONFIDENTIAL</transport-guarantee>
        </user-data-constraint>
</security-constraint>
```