Enabling SSL

SSL is enabled by default and is recommended to ensure secure, encrypted communication between the browser and DTP. If you are migrating from Concerto (version 4.x), you may need to enable SSL manually. DTP version 5.1.3 and later ships with Java 1.8.0_102-b14, which disables SSLv3 by default (see SSL for additional information).

In this section:

SSL for Enterprise Pack Application

If you are installing Extension Designer, you will either need to enable SSL for those applications or disable SSL in DTP so that the entire system is uses the same protocol (HTTP or HTTPS). See Getting Started with Enterprise Pack for instructions on enabling SSL.

Enabling SSL for DTP

Stop Parasoft services before making changes related to SSL. See Stopping DTP Services for instructions.

Step 1: Keystore Generation and Certificates

A .keystore file with signed certificate is required to enable SSL. DTP ships with a default .keystore file in the $DTP_HOME/tomcat/conf directory. The default .keystore file contains a self-signed certificate. You can replace the default .keystore file with your organization's .keystore file, but your file must contain a signed certificate.

If you do not already have a .keystore file available, you can also generate one by executing the following command:

```
keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks -storepass password -keysize 2048
```

This will create a keystore containing a private key and a self-signed certificate named keystore.jks with the password `password`. The `-keysize` setting is optional. The default keysize is `1024`.

You will be prompted to enter your organization information. When it asks for your first and last name, you typically enter the domain name of the server to be accessed. This is especially important if you are going to use a commercially-signed certificate. For a self-signed certificate, you could enter anything for first and last name (even your real first and last name). The prompt will also ask for a password for the generated key. The password can be the same as the password used for the keystore. In this case, the alias for the private key is `selfsigned`.

Obtaining a Commercial Certificate

You can obtain commercial certificates from a certificate authority (CA), such as verisign.com or thawte.com by submitting a certificate signing request (CSR) to the CA.

1. Use the following command to create the CSR:

   ```
   keytool -certreq -alias selfsigned -keystore keystore.jks -file cer- treq.csr
   ```

   You will be prompted to enter the keystore password. A certreq.csr CSR file is created for the key with the alias `selfsigned`.
2. The CA will return a Root or Chain certificate and the newly-signed certificate—both of which must be imported into your keystone. Use the following command to import your root certificate:

   ```
   keytool -import -alias root -keystore keystore.jks -trustcacerts - file
   <filename_of_the_chain_certificate>
   ```

3. Use the following command to import the new certificate:

   ```
   keytool -import -alias dtp -keystore keystore.jks -file <your_certificate_filename>
   ```

Step 2: Tomcat Configuration

Edit the server.xml configuration file located in the $DTP_HOME/tomcat/conf/ directory to configure Tomcat. Locate the <Connector port="80 or 8080" . . .> node in the <Service name="PST"> and add the following code after it:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true" maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" keystoreFile="$KEYSTORE_HOME/keystore.jks" keystorePass="$PASSWORD"
keyAlias="$ALIAS"/>
```

In this example, $KEYSTORE_HOME should be the directory of the keystone described in Keystore Generation and Certificates and keystore.jks is the file name. $PASSWORD is the user password specified when the keystore was created. $ALIAS is the alias assigned to the desired certificate in the keystore.

If the above snippet of code is already in server.xml, comment it out and add the keystoreFile and keystorePass.

The server.xml file will also contain the connector that specifies the port where DTP is already running. For example, if DTP runs on port 80, the connector you are looking for may look like this:

```
<Connector port="80" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />
```

Ensure that redirectPort points to the SSL connector specified previously. DTP can also be configured to run in a reverse proxy environment, which may require additional Tomcat configuration. See Reverse Proxy Support.

Step 3: DTP Configuration (Optional)

Perform this step if you want to deploy DTP solely on SSL (i.e., no HTTP port will be open).

Open the web.xml file located in $DTP_HOME/tomcat/webapps/grs/WEB-INF/ and uncomment out the following lines of code:

```
<security-constraint>
        <display-name>Security Constraint</display-name>
        <web-resource-collection>
                <web-resource-name>Protected Area</web-resource-name>
                <url-pattern>/*</url-pattern>
        </web-resource-collection>
        <user-data-constraint>
                <transport-guarantee>CONFIDENTIAL</transport-guarantee>
        </user-data-constraint>
</security-constraint>
```

If the snippet of code is not present, then add it manually under the web-app node.

In the $DTP_HOME/conf directory, open the PSTRootConfig.xml file and change <ssl-enabled>false</ ssl-enabled> to <ssl-enabled>true</ ssl-enabled>.