## SSL For Tomcat.

### Certificates

* For IIS we use .pfx files and this must be supplied by the customer.
* For Tomcat we require a p7b certificate and if it is not received from the customer, you can create own.
1. Launch IIS and select the server name, then Server Certificates.



1. Right-click on the certificate and select Export.
2. Export the .pfx file to the root of the drive where ProjectDox resides, choose a password and confirm the password. You will need this password during the Tomcat SSL process.



1. Launch the mmc.exe on the web server.
2. Choose File, Add/Remove Snap-in and choose Certificates.



1. Click the Add button in the middle of the screen.
2. Choose the third radio button, Computer account then click Next.



1. Continue with the default radio, Local computer (the computer this console is running on), then click Finish.



1. Expand Personal, select Certificate, right mouse click on the certificate.
2. Choose All Tasks, Export.



1. Click Next on the Welcome to the Certificate Export Wizard.
2. Click the radio for Cryptographic Message Syntax Standard for PKCS #7 Certificates (.P7B)
3. Place a checkmark for Include all certificates in the certification path if possible, click Next.



1. Use the Browse button and navigate to the root drive where ProjectDox is installed and save the file as a .p7b file.



1. You will be returned to the File name screen, click Next.
2. Once the process is complete, you will see the settings you specified, click Finish.



You now have a .pfx with a known password and a .p7b certificate to continue on to the next step.

## Installation DOS Commands

### Create the Keystore for Tomcat

1. Convert .pfx certificate file to JKS:
	1. Launch a command prompt on the web server.
	2. From Java folder (eg. C:\Program Files\Java\jre7\bin) run the following command:

**Sample:** keytool -importkeystore -srckeystore e:\certificate.pfx -srcstoretype pkcs12 -destkeystore destcert.jks -deststoretype JKS -storepass <yourpassword>

**Actual:** keytool -importkeystore -srckeystore e:\incoming\lorassl\avolvecloudsslcom.pfx -srcstoretype pkcs12 -destkeystore pdtomcert.jks -deststoretype JKS -storepass MyPassword

The result will give you the keyAlias value.

Entry for alias le-c5a08b67-625c-4184-8769-59eca73b6b55 successfully imported.

Import command completed: 1 entries successfully imported, 0 entries failed or

cancelled

In the above command, we did not provide a path for the –destkeystore parameter therefore, the JKS file is created in Java bin folder.

1. Copy the C:\Program Files\IGC\Tomcat 8 For Brava\conf\your\_p7b\_file.p7b into C:\Program Files\Java\jre7\bin.
2. Import certificate into JKS file (which is the keystore) using this command:

**Sample:** keytool -import -trustcacerts -alias alias\_from\_jks\_file -file your\_p7b\_file.p7b -keystore destcert.jks -storepass <yourpassword>

**Actual:** keytool -import -trustcacerts -alias le-c5a08b67-625c-4184-8769-59eca73b6b55 -file ssl\_certificate.p7b -keystore pdtomcert.jks -storepass <yourpassword>



## Edit server.xml

1. Copy destination JKS file to C:\Program Files\IGC\Tomcat 8 For Brava\conf folder.



1. Before changes are made to the server.xml file, save the original file as server.xmlOrig.
	1. If for any reason you need to revert to non SSL, activating the original file will make it very easy.
2. Edit C:\Program Files\IGC\Tomcat 8 For Brava\conf\server.xml file. You can skip a-d if you copy the server.xml file from \ProjectDox\Integration\BravaPlugin\JP folder and edit the following 2 values, this example is configured for SSL.

keystorePass="<<KEYSTOREPASSWORD>>"
keyAlias="<<KEYALIAS>>"

* 1. Uncomment the following section:

<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"

 maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

 clientAuth="false" sslProtocol="TLS" />

* 1. Update Tomcat to use newer TLS protocols by replacing *sslProtocol=”TLS”* with
	 *sslEnabledProtocols="TLSv1, TLSv1.1, TLSv1.2"*
	2. Add new keys to that section:

 <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"

             maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

clientAuth="false" sslEnabledProtocols="TLSv1, TLSv1.1, TLSv1.2"

keystoreFile="${catalina.base}/conf/pdtomcert.jks"

keystorePass=”XXXXXXXXXX”

keyAlias=”XXXXXXXXXXXXXXXXXXXX”

ciphers="TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256,

                    TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA,

                    TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA384,

                    TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA,

                    TLS\_ECDHE\_RSA\_WITH\_RC4\_128\_SHA,

                    TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256,

                    TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA,

                    TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256,

                    TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA,

                    SSL\_RSA\_WITH\_RC4\_128\_SHA"

 />

* 1. To turn off the logs in Tomcat that create excessive large files named localhost\_access\_log, comment out the following line:

<Valve className=”org.apache.catalina.valves.AccessLogValve”  directory=”logs” prefix=”localhost\_access\_log.” Suffix=”.txt” pattern=”%h %l %u %t &quot;%r&quot; %s %b” />

* 1. Save the server.xml file.