EEN-305: PIV/CAC support to Ephesoft

Starting Ephesoft v4.1.0.0, Personal Identity Verification (PIV)/Common Access Card (CAC) authentication solution is now supported.

Common Access Card (CAC) is a smart card used by employees and other personnel in the United States Department of Defense (DoD). CAC includes a picture of the user along with other information such as their name. DoD employees wear the CAC as a badge and can show it to security personals to prove their identity. They can also use it as a smart card to log onto systems.

Personal Identity Verification (PIV) card is a specialized smart card used by personnel in the United States federal agencies. Just as a CAC does, the PIV card includes a picture of the user along with their name. A PIV can be used for visual verification of users, and then as a smart card when users log onto their computers.

Client Certificate (aka. CLIENT-CERT) Authentication is used to enable PIV card based authentication in Ephesoft Transact application. A client digital certificate or client certificate is basically a file, usually protected with a password and loaded onto CAC/PIV based cards (usually as PKCS12 files with the .p12 or .pfx extension).

A client certificate typically contains pertinent information like a digital signature, expiration date, name of the client, name of CA (Certificate Authority), SSL/TLS version number, serial number, and possibly more, all structured using the X.509 standard.

At the start of a SSL or TLS session, the server requires the client application to submit a client certificate for authentication. Upon receiving the certificate, the server uses it to identify the certificate's source and determine whether the client should be allowed access.

Popular web browsers like Firefox, Chrome, Safari, and Internet Explorer support client certificates. If a server is enabled with client certificate authentication, only users who attempt to connect from clients loaded with the right client certificates will succeed. Even if a legitimate user attempts to connect with the right username and password, if that user is not on a client application loaded with the right client certificate, that user will not be granted access.

In case of PIV card, user has to connect the card to a system, so that browser can read the certificate from the card and send to the server. If the card is PIN enabled, then browser always prompts to enter the PIN before reading the certificate from the card.

Prerequisites

Following items are required to set up PIV authentication with Ephesoft Transact.

1. In case user has access to a trusted Certificate Authority (CA), then the user should go through the CA process to get a CA certificate, server certificate and server private key. In case user does not have a trusted Certificate Authority (CA), in such a case, one can create dummy CA

certificates to test the set up in LAB/TEST Environment. In an ideal scenario, user should have a trusted Certificate Authority (CA).

2. PKCS#12 format client certificates loaded in PIV cards; in case same is not there then user can create client certificates and load the certificates to PIV cards.
3. Active Directory/LDAP configurations for configuring Ephesoft realm.

Tools Used

1. OpenSSL (In case trusted CA is not available), you can download OpenSSL on windows from
   https://slproweb.com/products/Win32OpenSSL.html
   https://slproweb.com/download/Win32OpenSSL-1_0_2g.exe

You need install PERL on the system before using OpenSSL.

2. vSEC_CMS_K2.0 (For loading certificate to the card.), you can use other ways as well to create Client Certificate PIV Cards.

   http://www.ephesoft.com/Ephesoft_Product/pkadmin.zip

Setup

This section gives you a detailed walkthrough of the steps that you need to follow to complete the PIV/CAC integration process.

Generating Server Certificates to Set Up SSL/TLS

Ideally, in a production environment, you must make use of certificates issued by trusted CA.

But for lab/test environment, you can generate own dummy self-signed certificates for testing the PIV/CAC integration.

You can generate self-signed certificates using OpenSSL.

Creating Self-Signed Certificates Using OpenSSL

OpenSSL is open source general purpose cryptography library, which implementation of SSL and TLS.

To create self-signed certificates using OpenSSL

In case of **cacert.pem**, **servercert.pem** and **serverkey.pem** files are already available, then you can directly proceed to **Step 7** below.

1. Locate **OpenSSL CA.pl** file as this file is required to create dummy CA certificate file.
2. Create a directory to store certificates mkdir certificates.

3. On Linux, execute the command `/usr/lib/ssl/misc/CA.pl -newca`

   OR,

   On Windows, execute the above command replacing path of CA.pl with Windows path.

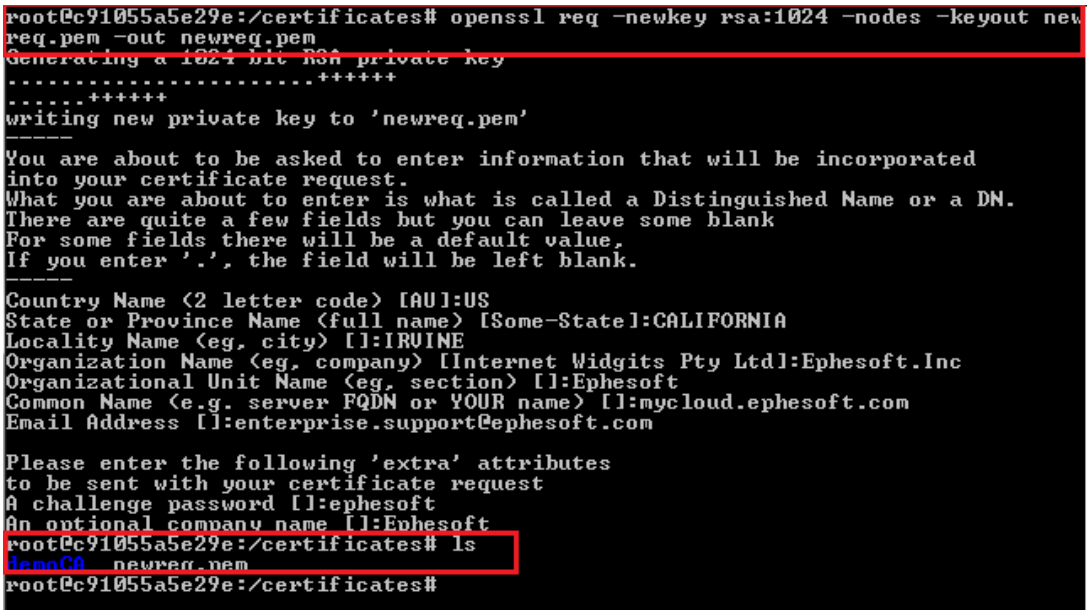
   This creates demoCA/cacert.pem **(CA Certificate) and** demoCA/private/cakey.pem **(private key).**

The generated **cacert.pem** is located inside **demoCA** folder.

```
root@c91055a5e29e:/certificates# /usr/lib/ssl/misc/CA.pl -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 2048 bit RSA private key
.....................................+++
.....................................................................................++
+
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CALIFORNIA
Locality Name (eg, city) []:IRVINE
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Ephesoft.Inc
Organizational Unit Name (eg, section) []:Ephesoft
Common Name (e.g. server FQDN or YOUR name) []:mycloud.ephesoft.com
Email Address []:enterprise.support@ephesoft.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:ephesoft
An optional company name []:Ephesoft
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 11265914040040247845 (0x9c5891214aaf5225)
        Validity
            Not Before: Apr 20 14:24:53 2016 GMT
            Not After : Apr 20 14:24:53 2019 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = CALIFORNIA
            organizationName          = Ephesoft.Inc
            organizationalUnitName    = Ephesoft
            commonName                = mycloud.ephesoft.com
            emailAddress              = enterprise.support@ephesoft.com
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                4A:8B:F2:49:A3:F5:A2:CF:6E:A8:D9:9F:BC:95:00:69:33:F7:56:04
            X509v3 Authority Key Identifier:
                keyid:4A:8B:F2:49:A3:F5:A2:CF:6E:A8:D9:9F:BC:95:00:69:33:F7:56:0
4

            X509v3 Basic Constraints:
                CA:TRUE
Certificate is to be certified until Apr 20 14:24:53 2019 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated
root@c91055a5e29e:/certificates#
```

4. Make a server certificate signing request (CSR) using the following command:

```
openssl req -newkey rsa:1024 -nodes -keyout newreq.pem -out newreq.pem
```

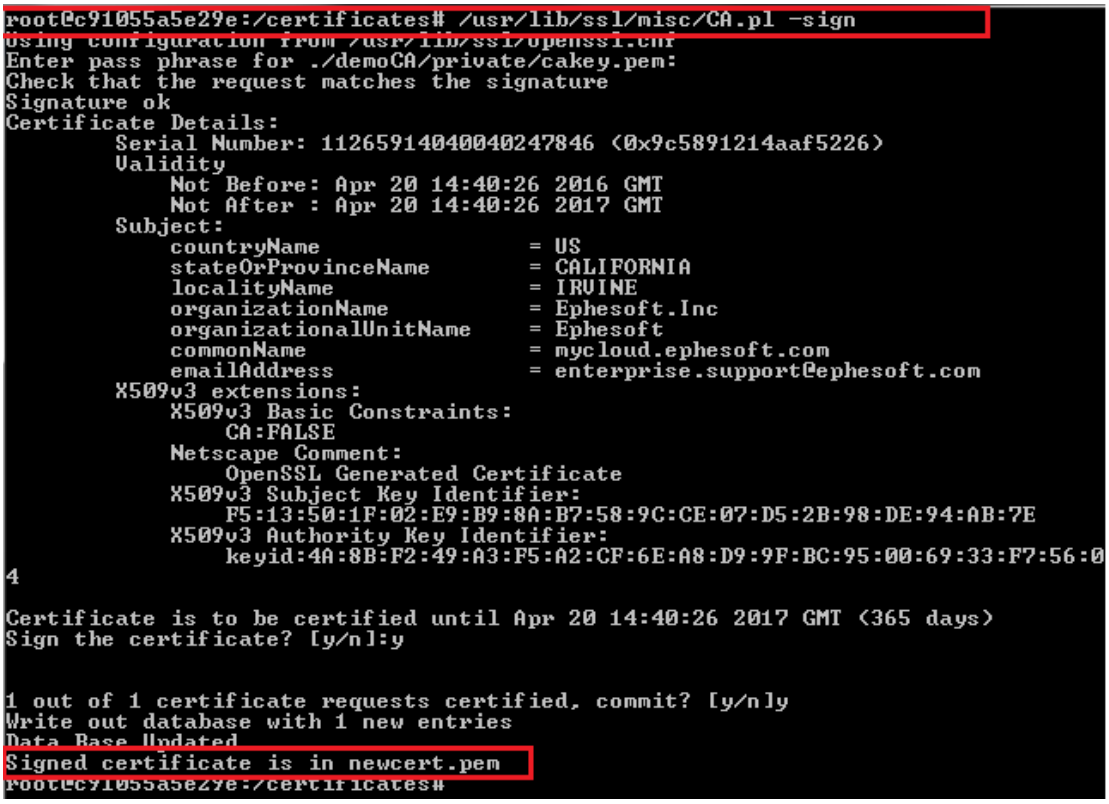


Note*: Make sure to use same name/value in Common Name as that of servername/hostname. Otherwise, the browser may complain while accessing that name does not match the hostname of the server. Adding to this make sure to access the server with the same hostname as mentioned here.

5. Create Server Certificate i.e. sign the certificate CSR (certificate signing request) with CA using the following command:

```
/usr/lib/ssl/misc/CA.pl -sign
```

Replace path of **CA.pl** file according to your operating system (Windows/Linux).



Following files shall be created after above mentioned steps have been executed:

- **cacert.pem** (CA certificate) created in Step 3
- **newreq.pem** (Server key) created in Step 4
- **newcert.pem** (Server certificate or signed certificate by CA) created in Step 5 above

6. For better clarity, rename these files:

   - Rename **newreq.pem** to **serverkey.pem**
   - Rename **newcert.pem** to **servercert.pem**

   Following PEM files will be available after renaming the original files:

   a. **cacert.pem**

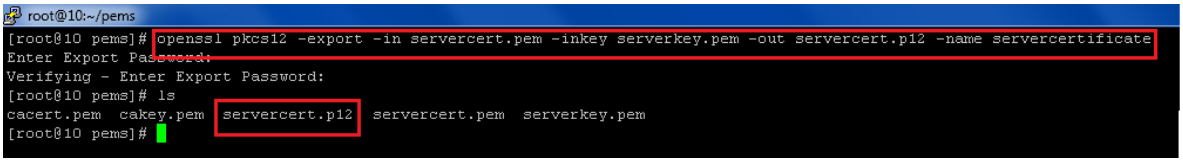
   b. **servercert.pem**

   c. **serverkey.pem**

7. Convert the **servercert.pem** file to PKC12 format (***.p12**) using the following command:

```
openssl pkcs12 -export -in servercert.pem -inkey serverkey.pem -out
servercert.p12 -name servercertificate
```

The converted file (**servercert.p12**) acts as a server certificate and is used to generate keystore. When prompted for **`Export Password`**, enter a password and keep the password safe.

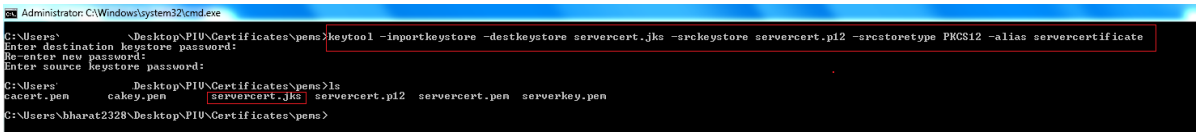


8. Create a java keystore file by converting the **servercert.p12** file to Java Keytool format by using the following command:

```
keytool -importkeystore -destkeystore servercert.jks -srckeystore
servercert.p12 -srcstoretype PKCS12 -alias servercertificate
```

When prompted for **`destination keystore password`**, enter a password and keep it safe. It will be used as **`keystore password`** in **server.xml** file.

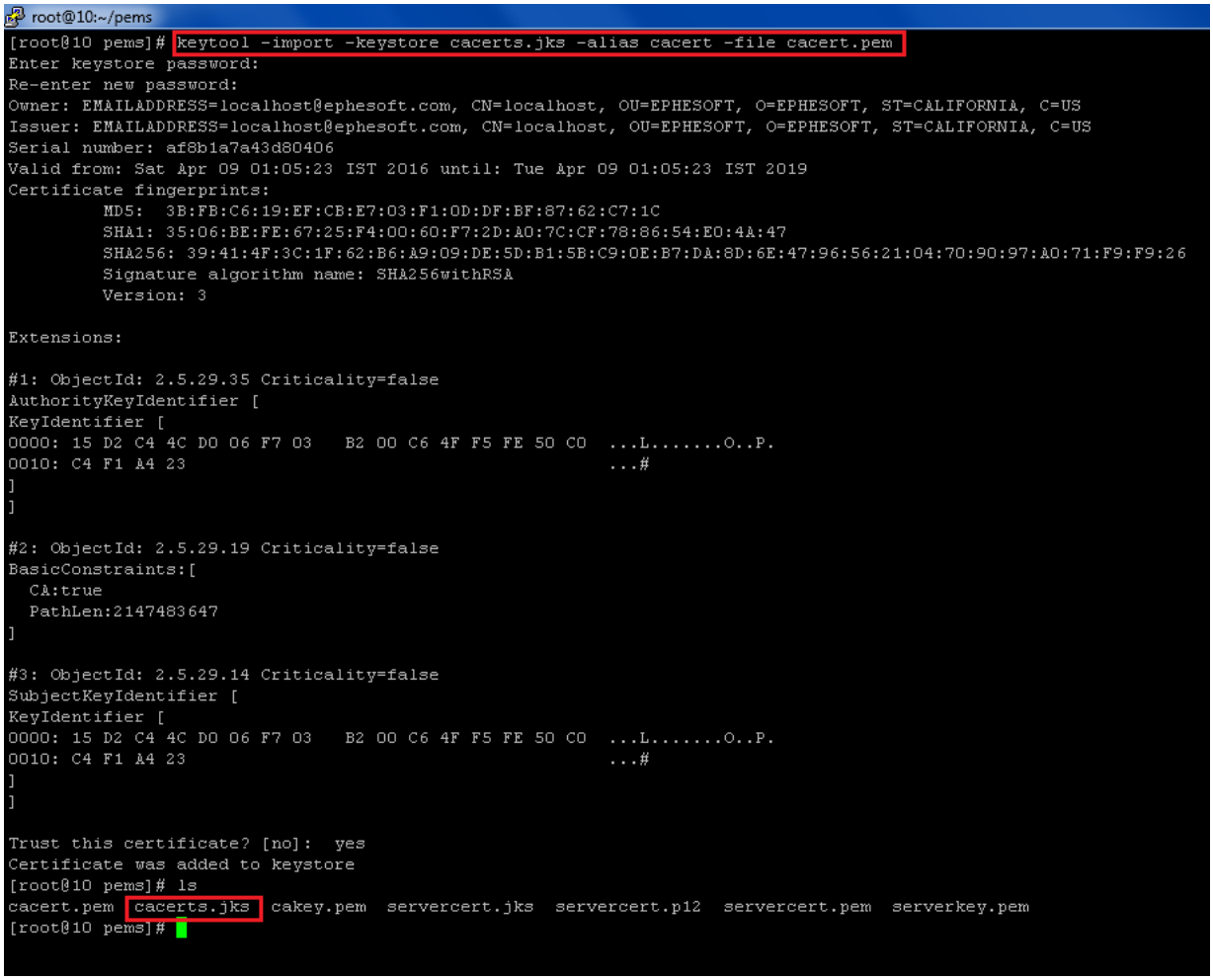
Also, when prompted for **`source keystore password`**, enter the export password for input **servercert.p12** file created in the previous step (Step 7).



9. Create a java truststore file by converting the **cacert.pem** file to Java Keytool format by using the following command:

```
keytool -import -keystore cacerts.jks -alias cacert -file cacert.pem
```

When prompted for **`keystore password`**, enter a password and keep the password safe. It will be used as **`truststore password`** in **server.xml**.



These files will be used while configuring SSL/TLS on Ephesoft Server.

Configuring SSL/TLS on Ephesoft JavaAppServer (Tomcat) Using Generated Certificates.

The following steps are involved in configuring SSL/TLS on Ephesoft Server.

Configuring Tomcat Connector Using Generated Keystore and Truststore Certificates

To configure Tomcat Connector

1. Take a backup of existing **server.xml** file located at **<Ephesoft>/JavaAppServer/conf** folder.
2. Open **server.xml** in edit mode, and locate existing HTTP/HTTPS connector as shown in snapshot.

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="900000" redirectPort="8443" compression="on"
noCompressionUserAgents="gozilla, traviata"
compressableMimeType="text/html,text/xml,text/css,text/javascript,image/tiff,image/jpg,image/ico,image/png,image/jpeg,application/pdf,application/x-pdf"
maxThreads="2000" maxKeepAliveRequests="200"/>
```

Existing HTTP/HTTPS Connector

```
<!-- Connector for enabling PIV/CAC configuration (with secure ciphers to prevent attacks like bar-mitzwah), please make sure to comment out HTTP connector before un-commenting PIV/CAC Connector-->
<!--
<Connector protocol="org.apache.coyote.http11.Http11NioProtocol"
port="8080" maxThreads="2000" clientAuth="false" scheme="https"
keepAliveTimeout="-1" connectionTimeout="900000" compression="on"
noCompressionUserAgents="gozilla, traviata"
compressableMimeType="text/html,text/xml,text/css,text/javascript,image/jpg,image/ico,image/png,image/jpeg,image/tiff,image/tif"
secure="true" SSLEnabled="true" sslProtocol="TLS" sessionTimeout="30"
truststoreFile="enter trust store complete path (/opt/Ephesoft/certs/trustore.jks)" truststorePass="enter truststore password"
keystoreFile="senter key store complete path (/opt/Ephesoft/certs/trustore.jks)" keystorePass="enter keystore password"
maxKeepAliveRequests="200" sslEnabledProtocols="TLSv1,TLSv1.1,TLSv1.2"
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"/>
-->
```

Commented PIV/CAC HTTPS Connector

Note:* For ease of configuration commented connector tag for configuring PIV/CAC is available.

Comment the existing connector (default port 8080) and create a new connector either by un-commenting the commented PIV/CAC HTTPS connector (shown in snapshot above) or by manually adding the following properties in server.xml:

- protocol: **org.apache.coyote.http11.Http11NioProtocol**
- port: **change port number if required (not mandatory)**
- maxThreads: **2000**
- client-auth: **false**
- schme: **https**
- keepAliveTimeout: **-1**
- connectionTimeout: **900000**
- compression: **on**
- noCompressionUserAgents: **gozilla, traviata**
- compressableMimeType: **text/html,text/xml,text/css,text/javascript,image/jpg,image/ico,image/png,image/jpeg,image/tiff,image/tif**
- secure: **true**
- SSLEnabled: **true**
- sslProtocol: **TLS**
- sessionTimeout: **30**
- truststoreFile: **cacerts.jks** (specify path of file generated in Step 9 above)
- truststorePass: **<password for cacerts.jks>** (generated in Step 9 above)
- keystoreFile: **servercert.jks** (specify path of file generated in Step 9 above)
- keystorePass: **<password for servercert.jks>** (generated in Step 8 above)
- maxKeepAliveRequests: **200**
- sslEnabledProtocols: **TLSv1,TLSv1.1,TLSv1.2**
- ciphers:**TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA38**

4,TLS_ECDHE_RSA_WITH_AES_256_CBC_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

This is a list of strong ciphers

<Connector protocol="org.apache.coyote.http11.Http11NioProtocol" port="8080" maxThreads="2000" clientAuth="false" scheme="https" keepAliveTimeout="-1" connectionTimeout="900000" compression="on" noCompressionUserAgents="gozilla, traviata" compressableMimeType="text/html,text/xml,text/css,text/javascript,image/jpg,image/ico,image/png,image/jpeg,image/tiff,image/tif" secure="true" SSLEnabled="true" sslProtocol="TLS" sessionTimeout="30" truststoreFile="enter trust store complete path (/opt/Ephesoft/certs/trustore.jks)" truststorePass="enter truststore password" keystoreFile="senter key store complete path (/opt/Ephesoft/certs/trustore.jks)" keystorePass="enter keystore password" maxKeepAliveRequests="200" sslEnabledProtocols="TLSv1,TLSv1.1,TLSv1.2" 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_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"/>

Make sure you replace the path of certificates with actual location of certificates.

```
<!-- Connector for enabling PIV/CAC configuration (with secure ciphers to prevent attacks like bar-mitzwah), please make sure to comment out HTTP connector before un-commenting PIV/CAC Connector-

<Connector protocol="org.apache.coyote.http11.Http11NioProtocol"
port="8080" maxThreads="2000" clientAuth="false" scheme="https"
keepAliveTimeout="-1" connectionTimeout="900000" compression="on"
noCompressionUserAgents="gozilla, traviata"
compressableMimeType="text/html,text/xml,text/css,text/javascript,image/jpg,image/ico,image/png,image/jpeg,image/tiff,image/tif"
secure="true" SSLEnabled="true" sslProtocol="TLS" sessionTimeout="30"
truststoreFile="enter trust store complete path (/opt/Ephesoft/certs/trustore.jks)" truststorePass="enter truststore password"
keystoreFile="senter key store complete path (/opt/Ephesoft/certs/trustore.jks)" keystorePass="enter keystore password"
maxKeepAliveRequests="200" sslEnabledProtocols="TLSv1,TLSv1.1,TLSv1.2"
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_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"/>
```

Modifying Existing EphesoftAuthenticator Valve in Server.XML

Change it to make use of EphesoftSSLAuthenticator (for using Web Services with basic authentication).

Note:* For ease of configuration commented valve tag for configuring PIV/CAC is available.

Please modify valve className to **`com.ephesoft.dcma.authenticator.EphesoftSSLAuthenticator`** and add new parameter **`cache="false" and changeSessionIdOnAuthentication="false"`**.

```
<Valve className="com.ephesoft.dcma.authenticator.EphesoftSSLAuthenticator"
cache="false" changeSessionIdOnAuthentication="false"/>
```

Or comment out the existing EphesoftAuthenticator valve and un-comment the commented PIV/CAC EhesoftSSLAuthenticator valve as shown in snapshot.

```
<Realm className="org.apache.catalina.realm.MemoryRealm" />
<!-- Comment EphesoftAuthenticator valve when using PIV/CAC authetication configuration -->
<!-- <Valve className="com.ephesoft.dcma.authenticator.EphesoftAuthenticator" /> -->
<!-- Un comment EphesoftSSLAuthenticator valve for PIV/CAC configuration and comment out  EphesoftAuthenticator valve -->

<Valve className="com.ephesoft.dcma.authenticator.EphesoftSSLAuthenticator" cache="false" changeSessionIdOnAuthentication="false"/>
```

Comment out EphesoftAuthenticator Valve

Un-comment EphesoftSSLAuthenticator valve

Configuring JavaAppServer MSAD/LDAP Realm Settings

Next step is to configure the JNDI realm.

Note: * For ease of configuration, commented Realm tag for configuring PIV/CAC realm configuration has been included in server.xml, same can be used to configure PIV/CAC realm configuration

```
<Manager pathname="" />
<!-- LDAP realm configuration for PIV/CAC based authetication configuration, please uncomment the realm and make necessary configuartion changes as per environment,
make sure to comment the existing realm setting before uncommenting the PIV/CAC realm configuration -->
<!--
<Realm className="org.apache.catalina.realm.JNDIRealm"
connectionURL="ldap://localhost:389"
connectionName="cn=Manager,dc=ephesoft,dc=com"
connectionPassword="secret"
userBase="ou=people,dc=ephesoft,dc=com"
userSearch="cn={0}"
roleBase="ou=groups,dc=ephesoft,dc=com"
roleName="cn"
roleSearch="uniqueMember={0}"
X509UsernameRetrieverClassName = "UserCert"/>
-->
```

Commented out PIV/CAC realm configuration

Please include the following parameters in case they are not present by defaults as per Microsoft Active Directory/LDAP:

- userBase: User base defines where to look for a user
- userSearch: search string for searching user
- roleBase: User base defines where to look for a role corresponding to a user
- roleName: defines which attribute is used for role
- roleSearch: search string for searching role
- X509UsernameRetrieverClassName: is a class which reads username from the client certificate and passes it for authentication/authorization to MSAD/LDAP

```
<Realm className="org.apache.catalina.realm.JNDIRealm"
connectionURL="ldap://localhost:389"
connectionName="cn=Manager,dc=ephesoft,dc=com" connectionPassword="secret"
userBase="ou=people,dc=ephesoft,dc=com" userSearch="cn={0}"
roleBase="ou=groups,dc=ephesoft,dc=com" roleName="cn"
roleSearch="uniqueMember={0}" X509UsernameRetrieverClassName = "UserCert"/>
```

Instead of adding these attributes manually, user can also comment existing realm configuration and uncomment the commented PIV/CAC Realm configuration and configure the attributes accordingly.

```
<Realm className="org.apache.catalina.realm.JNDIRealm"
connectionURL="ldap://localhost:389"
connectionName="cn=Manager,dc=ephesoft,dc=com"
connectionPassword="secret"
userBase="ou=people,dc=ephesoft,dc=com"
userSearch="cn={0}"
roleBase="ou=groups,dc=ephesoft,dc=com"
roleName="cn"
roleSearch="uniqueMember={0}"
X509UsernameRetrieverClassName = "UserCert"/>
```

If **userBase** or **roleBase** is set to parent e.g. **dc=ephesoft,dc=com** instead of **ou=people,dc=ephesoft,dc=com**, then user may need to configure realm with **userSubtree** and **roleSubtree** parameter as **true**.

- **roleSubtree** - the search scope. Set to true to search the entire subtree rooted at the roleBase entry. The default value of false requests a single-level search including the top level only.
- **userSubtree** - the search scope. Set to true to search the entire subtree rooted at the userBase entry. The default value of false requests a single-level search including only the top level.

```
<Realm className="org.apache.catalina.realm.JNDIRealm"
connectionURL="ldap://10.1.0.107:3268"
        connectionName="CN=JOHN,OU=Users,OU=Ephesoft,DC=Corp,DC=ephesoft,DC=com"
        connectionPassword="P@ssw0rd"
        userBase="DC=Corp,DC=ephesoft,DC=com"
        userSearch="cn={0}"
        roleBase="DC=Corp,DC=ephesoft,DC=com"
        roleName="cn"
        roleSearch="member={0}"
        userSubtree="true"
        roleSubtree="true"
        X509UsernameRetrieverClassName="UserCert" />
```

Commenting Out AprLifecycleListener in Server.XML

Search for **AprLifecycleListener** in the server.xml and comment it out as follows:

```
 <!--<Listener className="org.apache.catalina.core.AprLifecycleListener"
SSLEngine="on" />-->
```

```
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
  -->
  <!--APR library loader. Documentation at /docs/apr.html -->
<!--  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />-->
  <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-howto.html -->
  <Listener className="org.apache.catalina.core.JasperListener" />
  <!-- Prevent memory leaks due to use of particular java/javax APIs-->
```

Configuring Ephesoft Application for CERT-AUTH

Please modify **web.xml** file located in **<Ephesoft>/Application/WEB-INF** folder to use certificate-based authorization.

To modify web.xml

1. Take a backup of existing **web.xml** file.
2. Modify the following **<context param>** entries:

Note* : Comments have been added in web.xml to assist the user to configure these values.

Parameter Name	Updated Value
port	Enter value to match value server.xml. <context-param> <param-name>port</param-name> <!-- change port number if changed in server.xml, please change this port in case PIV/CAC configuration has been done for other port <param-value>8080</param-value> </context-param>
protocol	Enter **HTTPS**. <context-param> <param-name>protocol</param-name> <!-- change protocol to https when using PIV/CAC based authetication --> <param-value>http</param-value> </context-param>

3. Locate **<login-config>** entry and modify the **<auth-method>** value from **FORM** to **CLIENT-CERT**.

Note* : Commented login-config tag for configuring PIV/CAC has been included in web.xml, one can just comment the existing tag and uncomment the commented PIV/CAC configuration.

<login-config>
<auth-method>CLIENT-CERT</auth-method>
</login-config>
<security-role>
<role-name>*</role-name>
</security-role>

```
<!-- PIV/CAC configutaion, please un-comment when configuring PIV/CAC, make sure to comment FORM auth login config -->
<!--
<login-config>
    <auth-method>CLIENT-CERT</auth-method>
</login-config>
<security-role>
    <role-name>*</role-name>
</security-role>
-->
```

Configuring Ephesoft Application Properties

1. Open **dcma-batch.properties** located at **<ephesoft.install.dir>/Application/WEB-INF/classes/META-INF/** folder and **modify batch.base_http_url** to make use of https protocol, correct port and hostname.

```
batch.base_http_url=https\://localhost\:8080/dcma-batches
```

2. Open **dcma-workflows.properties** located at **<ephesoft.install.dir>/Application/WEB-INF/classes/ META-INF/dcma-workflows** folder and change the port for property **wb.hostURL** to make use of https protocol, correct port and correct host.

```
wb.hostURL=https://localhost:8080/dcma/rest
```

3. Configure Ephesoft to use MSAD/LDAP by configuring **user-connectivity.properties** file in **<ephesoft.install.dir>/Application/WEB-INF/classes/META-INF/dcma-user-connectivity** folder. Make sure configuration is in accordance with **Server.xml** configuration.

4. Restart Ephesoft Server.

Creating Client Certificates

The process of creating client certificates is similar to the process of Creating Self-Signed Certificates Using OpenSSL.

To create client certificates

1. Make client certificate signing request (CSR) using the following command:

```
openssl req -newkey rsa:1024 -nodes -keyout newreq.pem -out newreq.pem
```

Enter client information.

```
root@0a40443e5bdd:~/certificates# openssl req -newkey rsa:1024 -nodes -keyout ne
wreq.pem -out newreq.pem
Generating a 1024 bit RSA private key
...++++++
.....................................................++++++
writing new private key to 'newreq.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:NewYork
Locality Name (eg, city) []:QUEENS
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Ephesoft.Inc
Organizational Unit Name (eg, section) []:Ephesoft
Common Name (e.g. server FQDN or YOUR name) []:JOHN
Email Address []:john@ephesoft.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:ephesoft
An optional company name []:Ephesoft
root@0a40443e5bdd:~/certificates#
```

2. Create Client Certificate i.e. sign the certificate CSR (certificate signing request) with CA using the following command:

   ```
   /usr/lib/ssl/misc/CA.pl -sign
   ```

Replace path of **CA.pl** file according to your operating system (Windows/Linux).

```
root@0a40443e5bdd:~/certificates# /usr/lib/ssl/misc/CA.pl -sign
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 10075278562431066926 (0x8bd29478aecfe32e)
        Validity
            Not Before: Apr 20 19:45:08 2016 GMT
            Not After : Apr 20 19:45:08 2017 GMT
        Subject:
            countryName               = US
            stateOrProvinceName       = NewYork
            localityName              = QUEENS
            organizationName          = Ephesoft.Inc
            organizationalUnitName    = Ephesoft
            commonName                = JOHN
            emailAddress              = john@ephesoft.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                D3:DB:49:8A:80:15:E1:76:06:17:74:9D:D2:B6:6B:CA:79:B1:B0:98
            X509v3 Authority Key Identifier:
                keyid:A9:53:24:D4:BB:D4:7E:EC:07:A9:3F:37:18:C8:2A:66:2B:67:95:9
0

Certificate is to be certified until Apr 20 19:45:08 2017 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Signed certificate is in newcert.pem
root@0a40443e5bdd:~/certificates#
```

A Client Certificate file in PEM format (newcert.pem**) is generated.**

3. Convert the **newcert.pem** file to PKC12 format (*****.**p12**) using the following command:

   ```
   openssl pkcs12 -export -in newcert.pem -inkey newreq.pem -out John.12
   ```

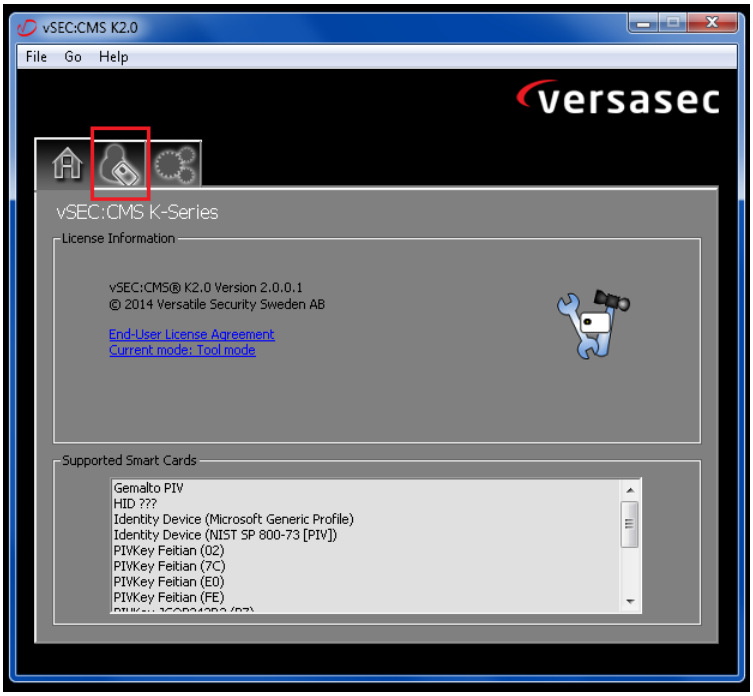
```
root@0a40443e5bdd:~/certificates# openssl pkcs12 -export -in newcert.pem  -inkey
 newreq.pem  -out John.p12
Enter Export Password:
Verifying - Enter Export Password:
root@0a40443e5bdd:~/certificates#
```

Loading Client Certificates into PIV Cards

To load client certificates into PIV cards

1. Install vSEC_CMS_K2.0 and PIV installer admin utility.
2. Attached the card reader and the PIV card to the system and open vSEC_CMS_K2.0 utility.

The following screen displays.

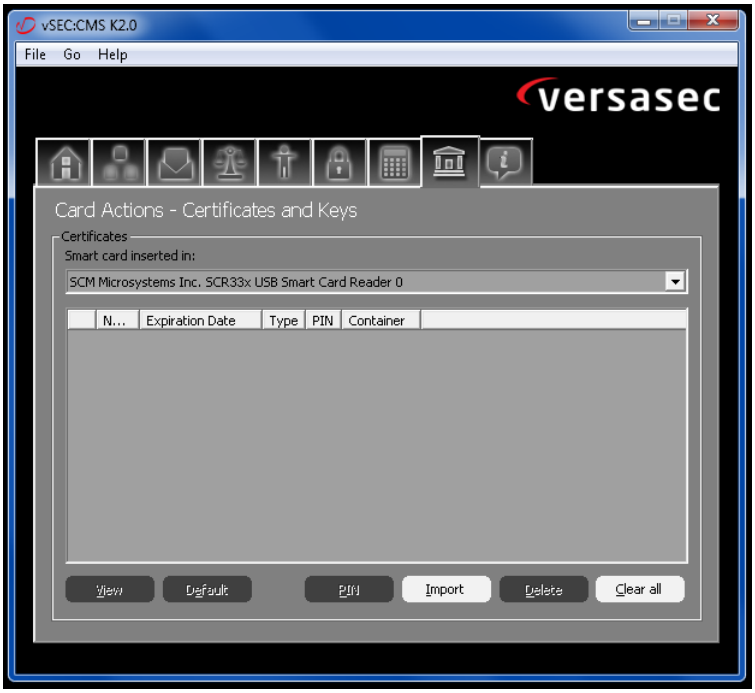


3. Select the **CARD ACTIONS** tab.

   The Card Actions-Online Unblock **tab displays.**

4. Select the **CERTIFICATES AND KEYS** tab.

The Card Actions-Certificates and Keys **tab displays.**



5. Click **IMPORT** to import the client certificate to the PIV card.

   The following screen displays.

6. Click **BROWSE** to locate and select the **.p12** format certificate from the file system.

7. Enter the **.p12** certificate password in **Password** text box.
8. Click **IMPORT**.

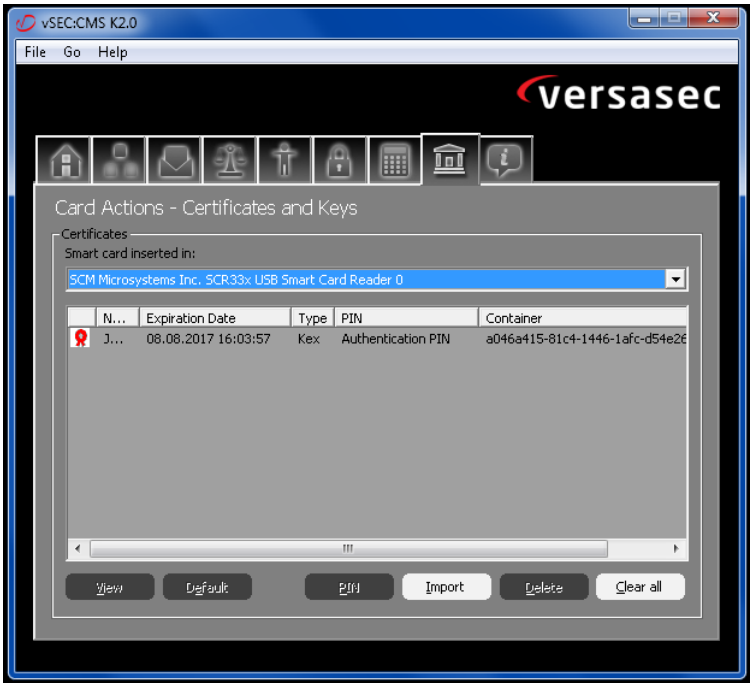
The following screen displays.

9. Enter a passcode in the **Passcode** text box.

⚠ It is important to note the passcode entered above will be used by Ephesoft during authentication and authorization

---

The Card Actions-Certificates and Keys **tab is updated displaying the certificate imported into the PIV card.**



10. Close the vSEC_CMS_K2.0 utility.

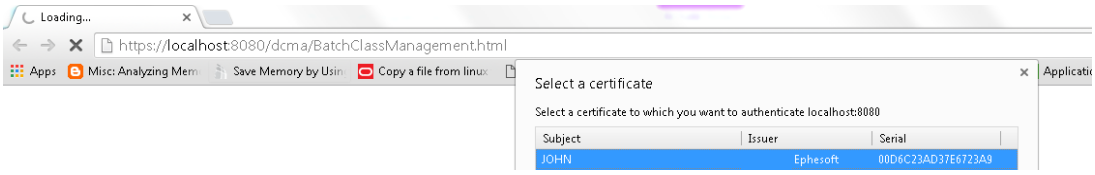
Testing PIV Cards Authentication/Authorization

This section describes a quick way to test PIV cards authentication/authorization.

To test PIV cards authentication/authorization

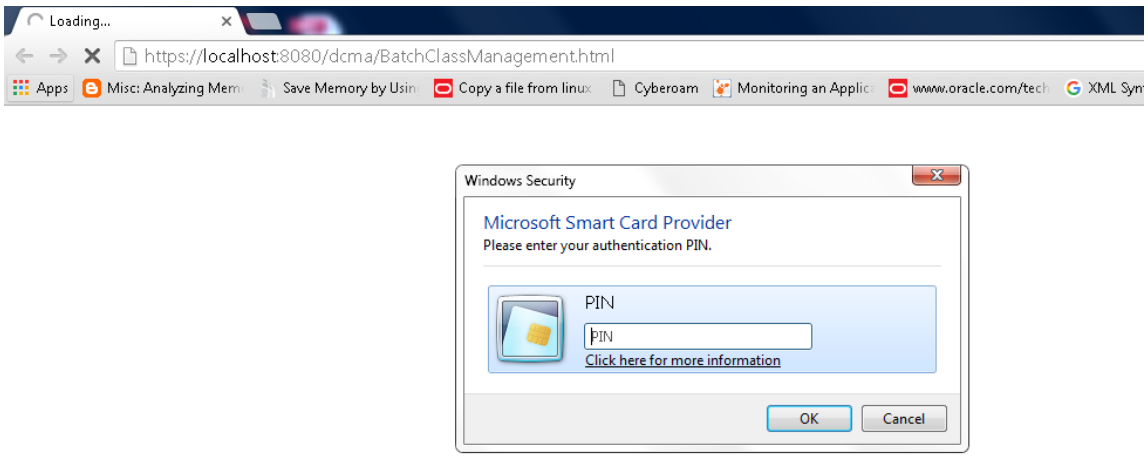
1. Start Ephesoft server.
2. Open a browser window on a client system.
3. Open **Batch Class Management** screen using an HTTPS URL.

   If card is attached to the client system, then browser prompts you to select a certificate.



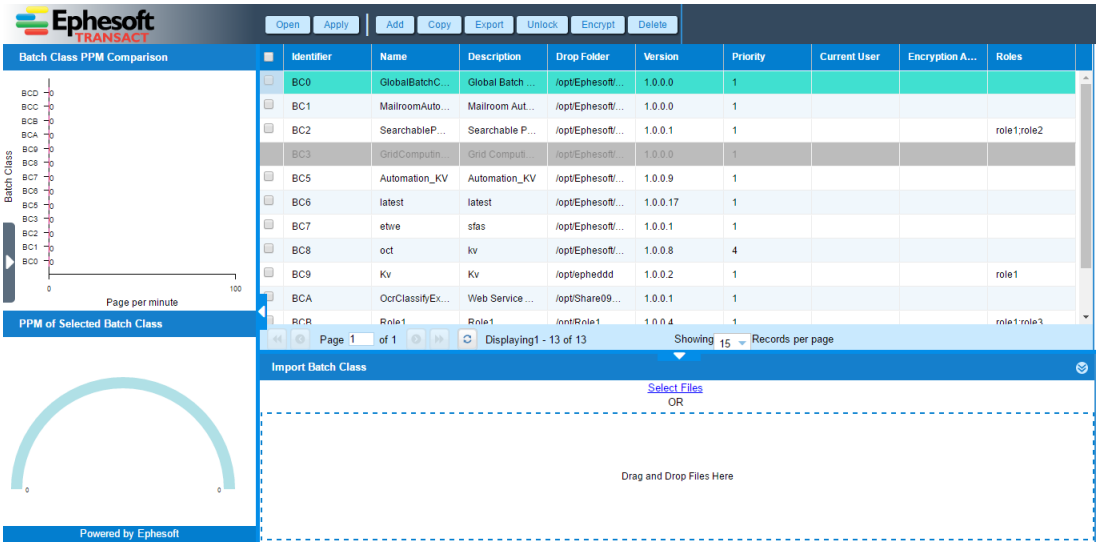
4. Select a certificate to which you want to authenticate.

   The browser prompts you to enter the authentication PIN.



5. Enter the correct authentication PIN in the **PIN** text box.

   The Batch Class Management screen displays.



⚠ Please make sure that user corresponding to the certificate is present in MSAD/LDAP with proper roles i.e. roles valid for Ephesoft Transact application. If the user does not exist in MSAD/LDAP or correct roles are not assigned, then the user will not be authenticated/authorized.

Troubleshooting & FAQs

Enabling Web Services – With Basic Authentication

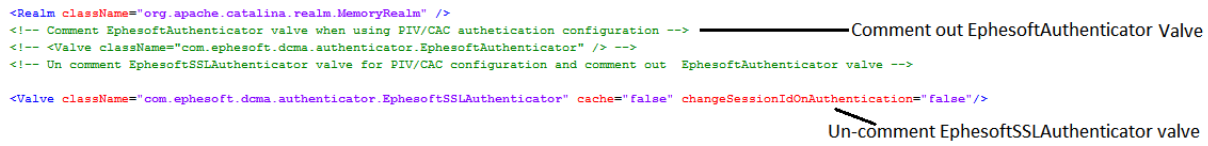
Change it to make use of EphesoftSSLAuthenticator (for using Web Services with basic authentication).

Note:* For ease of configuration commented valve tag for configuring PIV/CAC is available.

Please modify valve className to **`com.ephesoft.dcma.authenticator.EphesoftSSLAuthenticator`** and add new parameter **`cache="false" and changeSessionIdOnAuthentication="false"`**.

```
<Valve className="com.ephesoft.dcma.authenticator.EphesoftSSLAuthenticator"
cache="false" changeSessionIdOnAuthentication="false"/>
```

Or comment out the existing EphesoftAuthenticator valve and un-comment the commented PIV/CAC EhesoftSSLAuthenticator valve as shown in snapshot.



Issues with File Uploads

In case issues are observed with file uploads, please comment out **AprLifecycleListener** in the **server.xml**

```
 <!--<Listener className="org.apache.catalina.core.AprLifecycleListener"
SSLEngine="on" />-->
```

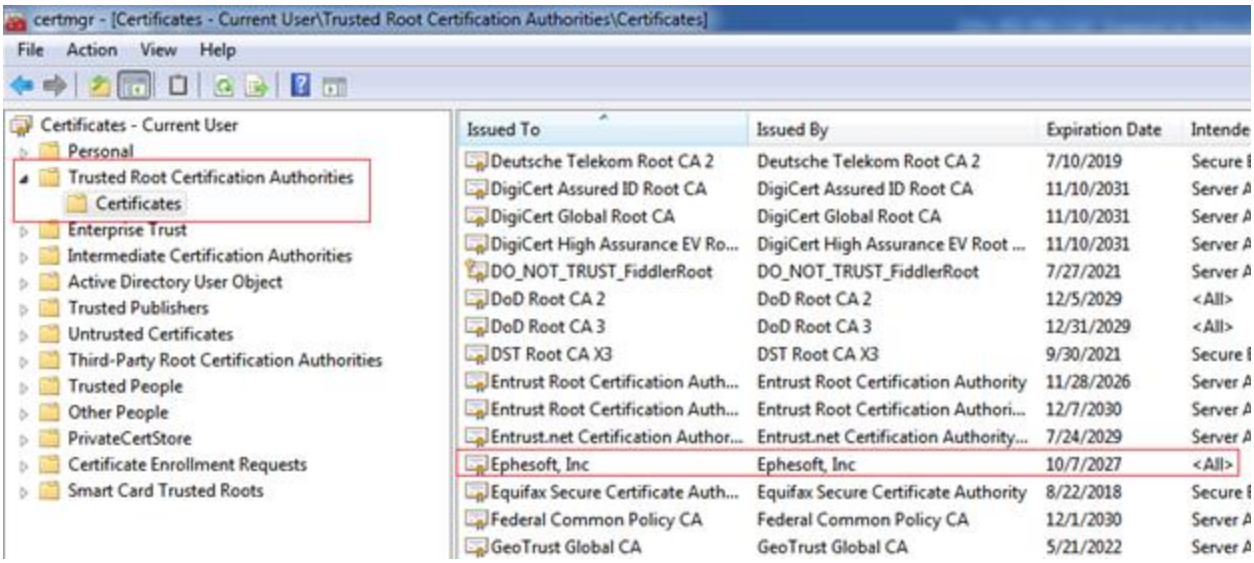
Testing Web Scanner

Since self-signed certificates are being used for configuring the application a few extra steps are required to configure Web Scanner.

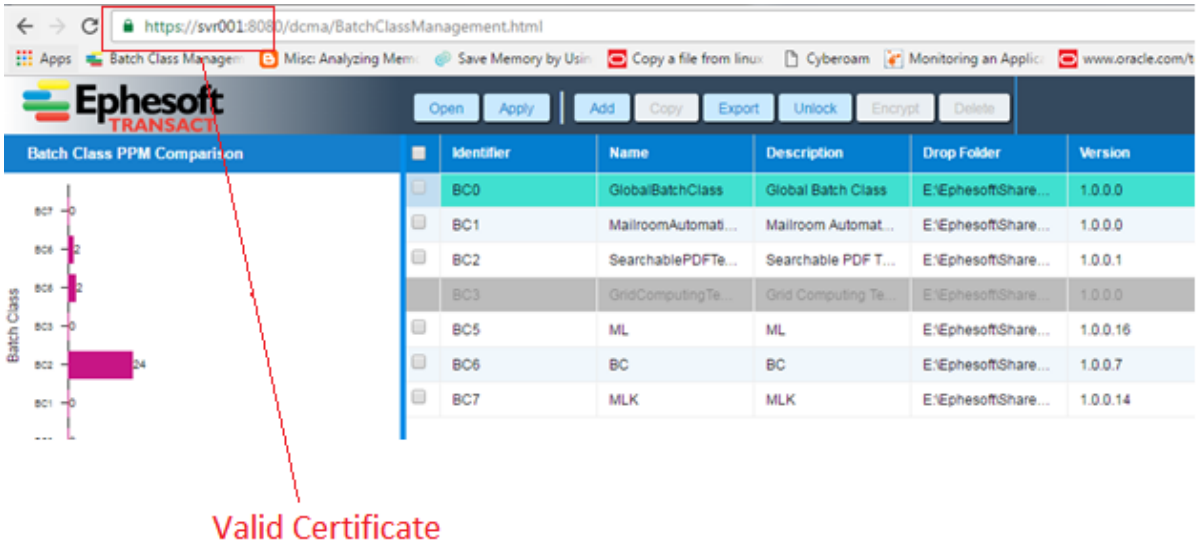
In case certificates signed by valid authority are being used then these steps may not be required.

Since certificates being used are self-signed certificates thus these certificates must be imported as Trusted Root Certificates on Web Scanning Machine (where Ephesoft Web Scanner Service is executing)

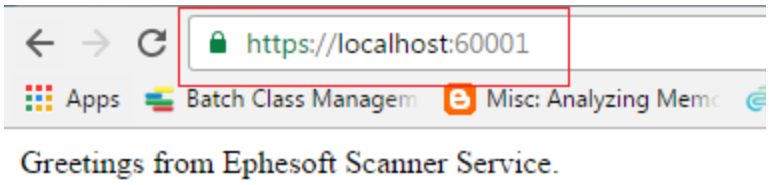
1. Open certmgr.msc and select Trusted Root Certification Authorities, please import following certificates:



a. Cacert.pem generated while generating server certificates. (CACERT is certificates which represents a Certificate Signing Authority). Certificates being used for configuring Ephesoft Server are signed by this generated Certificate Authority (CACERT), thus importing this certificate as trusted root certificate on client machine makes it valid for communication with the client and the server.
Please refer snapshot certificate and HTTPS appears to be valid to the browser.



b. Import Ephesoft-Browser-Cert.crt present in EphesoftScannerService installation folder as trusted root certificate. This step is also required to make communication between Web Scanner Client (browser) and Ephesoft Web Scanner Server (internal) valid.
Please refer snapshot certificate and HTTPS appears to be valid to the browser.



2. Next step is to import self-signed CA (CACERT) generated above in EphesoftWebScannerService JRE Keystore.

   Locate cacerts located at < EphesoftScannerService>\jre\lib\security.

   keytool -importcert -keystore <EphesoftScannerService>\jre\lib\security\cacerts -storepass changeit -alias ephesoftrootCA -file cacert.pem

   Please substitute the path of cacerts and cacert.pem accordingly.

3. Restart Ephesoft Web Scanner Service.
4. Please make sure to access Ephesoft Server with the same hostname as was used while generating the server certificate otherwise communication will become insecure and the browser may not allow insecure communication.