Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Rev.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 18, 2018</td>
<td>4.5.0.x</td>
<td>1.0</td>
<td>Initial publication of the document, for manually configuring SSO with ADFS and SAML 2.0 (not automated configuration). This document is also available in the Ephesoft Wiki at the following URL:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <a href="https://wiki.ephesoft.com/configuring-saml-sso">https://wiki.ephesoft.com/configuring-saml-sso</a></td>
</tr>
<tr>
<td>November 1, 2018</td>
<td>4.5.0.x</td>
<td>1.1</td>
<td>Added Transport Layer Security (TLS) support information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1.3. SSO Support Information for Ephesoft Transact</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Added a new issue resolution to the “Troubleshooting” chapter:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 5. No redirection to the SSO Login screen upon login to Transact</td>
</tr>
<tr>
<td>April 2, 2019</td>
<td>4.5.0.x 2019.1</td>
<td>1.1b</td>
<td>Updated hypertext links to support Ephesoft.com/Docs.</td>
</tr>
</tbody>
</table>

Contents
Revision History .......................................................................................................................... 2
1. Introduction ........................................................................................................................... 4
   1.1. SSO Overview ............................................................................................................ 4
   1.2. Benefits ...................................................................................................................... 4
   1.3. SSO Support Information for Ephesoft Transact .................................................... 4
   1.4. SSO Configuration Overview on Ephesoft Transact ............................................... 5
2. Prerequisites ......................................................................................................................... 6
   2.1. Prerequisite Components ......................................................................................... 6
3. Obtaining and Installing OpenSSL — Required .................................................................. 6
4. Configuring SSL/TLS — Required ....................................................................................... 7
   4.1. Introduction ............................................................................................................... 7
   4.2. SSL/TLS Setup: Server Certificates ......................................................................... 7
      4.2.1. Production Environment: CA Certificates ......................................................... 7
      4.2.2. Testing Environment: Self-Signed Certificates ................................................. 7
5. Configuring SSL/TLS for Ephesoft Transact — as Needed ............................................. 13
   5.1. Introduction ............................................................................................................. 13
   5.2. SSL/TLS Setup: Configuring the Ephesoft Transact JavaAppServer (Tomcat) . . . . . . 13
   5.3. SSL/TLS Setup: Configuring Ephesoft Transact Files ............................................. 14
      5.3.1. dcma-batch.properties ................................................................................... 15
      5.3.2. dcma-workflows.properties ............................................................................ 15
      5.3.3. web.xml ......................................................................................................... 15
6. Configuring Ephesoft Transact for SAML SSO ................................................................. 16
   6.1. Introduction to SAML .............................................................................................. 16
6.2. SAML v2 Support in Ephesoft Transact.................................................................16

7. Configurations for Integrating SAML-Enabled Spring Security Framework..............17

7.1. Introduction .............................................................................................................17

7.2. applicationContext.xml ...........................................................................................17

7.3. web.xml ....................................................................................................................17

7.3.1. Multiple Group Support in web.xml .................................................................19

7.4. samlKeystore.jks .....................................................................................................19

7.4.1. Create a private key .......................................................................................19

7.5. applicationContext-security.xml ............................................................................20

7.6. Testing at the SSOCircle .........................................................................................24

7.7. Configuring Ephesoft Transact and ADFS ............................................................24

7.7.1. Configuring Ephesoft Transact for the ADFS Server ...................................... 24

Obtain the ADFS Metadata ..................................................................................25

Import the ADFS Certificates to Ephesoft Transact ............................................25

7.7.2. Configuring the ADFS Server for Ephesoft Transact ...................................... 25

7.7.3. Exporting the Certificate from the ADFS Server .............................................29

7.7.4. Successful Configuration ...............................................................................31

7.8. Configuring Okta with Ephesoft Transact .............................................................31

7.8.1. Introduction .................................................................................................... 31

7.8.2. Creating the Okta Developer Account ............................................................ 31

7.8.3. Assigning People in the Okta Account to Ephesoft Transact .......................... 38

7.8.4. Adding People to the Okta Account ............................................................... 39

7.8.5. Configuring Ephesoft Transact to Operate with Okta .....................................40

8. Exporting the Self-signed Certificate from Keystore ....................................................43

9. Troubleshooting SSO in Ephesoft Transact .............................................................43

9.1. Introduction ..........................................................................................................43

9.2. General Troubleshooting Tasks ...........................................................................43

9.3. Logging for SAML Requests .................................................................................45

9.4. Getting the Fiddler Trace .......................................................................................45
1. Introduction

1.1. SSO Overview

Single sign-on (SSO) is a mechanism of access control that can be applied on multiple related, but independent software systems. With this mechanism a user logs in once and gains access to multiple systems without being prompted to log in again for each individual application. Conversely, single sign-off is a property mechanism whereby a single action of signing out terminates access to multiple software systems.

As different applications and resources support different authentication mechanisms, single sign-on internally translates to and stores different credentials, compared to what is used for initial authentication.

1.2. Benefits

SSO has several general benefits, to include the following:

- Reducing password fatigue from different user name and password combinations
- Reducing time spent re-entering passwords for the same identity
- Reduced logins for discreet systems
  - Corporate systems
  - Shared intranet/web applications
  - Web logon aggregators
- Reduced cost to reset a password
- Reduced time spent logging into multiple systems each time
- Reduces multiple authentication, unnecessary user clicks, forgotten passwords, multiple profiles
- Limited time and resources to develop IT solutions

1.3. SSO Support Information for Ephesoft Transact

This document describes how to make manual SAML 2.0 SSO configurations during a new installation of Ephesoft Transact 4.5.0.x or 2019.1.

Note: Automated SSO configuration for Ephesoft Transact does not have scheduled availability.

Ephesoft Transact has been tested with the following SSO-related components:

- **SAML 2.0** — An XML-based protocol that entails security tokens for authentication and authorization. Ephesoft Transact supports SAML 2.0, and has tested and certified the following identity providers that are compatible with Ephesoft Transact:
  - ADFS
  - Okta
  - PingFederate
  - SSOCircle

- **Transport Layer Security (TLS)** — Ephesoft Transact 4.5.0.x and 2019.1 support TLS versions 1.0, 1.1 and 1.2.

There are multiple additional identity providers in the market that support SAML 2.0. Ephesoft has not tested every available identity provider or security product.
1.4. SSO Configuration Overview on Ephesoft Transact

This document describes how to make manual SAML 2.0 SSO configurations during a new installation of Ephesoft Transact 4.5.0.x or 2019.1.

Note: This document emphasizes tasks for operational deployment. At some time in the future, Ephesoft may provide a separate document that describes security and SSO concepts.

This document contains the following sections and purposes:

<table>
<thead>
<tr>
<th>Section Title</th>
<th>Purpose and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Prerequisites</td>
<td>Required — This section identifies the prerequisite conditions that must be in place in order to configure SSO for use with Ephesoft Transact.</td>
</tr>
<tr>
<td>3. Obtaining and Installing OpenSSL</td>
<td>Required — This section describes the steps for downloading and installing OpenSSL. You must complete these steps prior to performing the additional SSO configurations in this document.</td>
</tr>
<tr>
<td>4. Configuring SSL/TLS</td>
<td>Required — This section describes the steps for configuring multiple components of the Secure Sockets Layer (SSL) protocol or the Transport Layer Security (TLS) protocol. One or more of these components may need to be configured, depending on your current requirements and status. Not all identity providers (IdP) will require SSL/TLS configuration, but this section provides instructions for those IdPs that do require it.</td>
</tr>
<tr>
<td>5. Configuring Ephesoft Transact for SSL/TLS</td>
<td>Required — This section describes how to configure Ephesoft Transact over SSL/TLS.</td>
</tr>
<tr>
<td>5.1. Introduction</td>
<td></td>
</tr>
<tr>
<td>5.2. SSL/TLS Setup: Configuring the Ephesoft Transact JavaAppServer (Tomcat)</td>
<td></td>
</tr>
<tr>
<td>5.3. SSL/TLS Setup: Configuring Ephesoft Transact Files</td>
<td></td>
</tr>
<tr>
<td>6. Configuring Ephesoft Transact for SAML SSO</td>
<td>Required — This section describes the steps for configuring SAML SSO within Ephesoft Transact.</td>
</tr>
<tr>
<td>7. Configurations for Integrating SAML-Enabled Spring Security Framework</td>
<td>Required — This section describes the steps for configuring the various components of the SAML-enabled Spring Security framework. There are several elements to configure in this phase of SSO setup.</td>
</tr>
<tr>
<td>7.1. Introduction</td>
<td></td>
</tr>
<tr>
<td>7.2. applicationContext.xml</td>
<td></td>
</tr>
<tr>
<td>7.3. web.xml</td>
<td></td>
</tr>
<tr>
<td>7.4. samlKeystore.jks</td>
<td></td>
</tr>
<tr>
<td>7.5. applicationContext-security.xml</td>
<td></td>
</tr>
<tr>
<td>7.6. session-timeout.js</td>
<td></td>
</tr>
<tr>
<td>7.7. Testing at the SSO Circle</td>
<td></td>
</tr>
<tr>
<td>7.8. Configuring Ephesoft Transact and ADFS</td>
<td></td>
</tr>
<tr>
<td>7.9. Configuring Okta with Ephesoft Transact</td>
<td></td>
</tr>
<tr>
<td>8. Exporting the Self-signed Certificate from Keystore</td>
<td>As Needed — This short section describes how to export the self-signed certificates, if you need these for the additional configurations in this document.</td>
</tr>
<tr>
<td>9. Troubleshooting SSO in Ephesoft Transact</td>
<td>As Needed — This section provides a reference monitoring and troubleshooting SSO configuration issues in Ephesoft Transact.</td>
</tr>
<tr>
<td>9.1. Introduction</td>
<td></td>
</tr>
<tr>
<td>9.2. General Troubleshooting Tasks</td>
<td></td>
</tr>
<tr>
<td>9.3. Logging for SAML Requests</td>
<td></td>
</tr>
<tr>
<td>9.4. Obtaining the Fiddler Trace</td>
<td></td>
</tr>
</tbody>
</table>
2. Prerequisites

2.1. Prerequisite Components

The following items are required to set up SAML SSO authentication with Ephesoft Transact.

1. **CA Certificate** — If you have access to a trusted Certificate Authority (CA), you should complete the CA process to get a CA certificate, server certificate and server private key.

   If you do not have a trusted Certificate Authority (CA), you can create dummy CA certificates to test the setup in a lab or test environment.

   In an ideal scenario, you should have a trusted Certificate Authority (CA).

2. **Identity Provider** — You must have an installed identity provider (IdP) server that supports SAML 2.0. Ephesoft has tested and certified Ephesoft Transact with the following four identity providers:
   - Active Directory Federation Services (ADFS)
   - Okta
   - PingFederate
   - SSOCircle

3. **Active Directory** — This document does not provide step-by-step instructions for installing or configuring Active Directory. If you need to set up Active Directory in preparation for configuring SSO, refer to one or more of the following articles:
   - *How to: Configure Active Directory*
     [https://ephesoft.com/docs/configurations/user-connectivity/how-to-configure-active-directory/](https://ephesoft.com/docs/configurations/user-connectivity/how-to-configure-active-directory/)
   - *KB0007802 Active Directory Configuration/Setup: Sample File Changes for Proper Setup*
   - *Multiple Groups as Roles in Active Directory*
   - *How to Administer Ephesoft Users & Groups*

3. Obtaining and Installing OpenSSL — Required

OpenSSL is a robust, commercial-grade, and full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. Use OpenSSL in the case that a trusted Certificate Authority (CA) certificate is not available.

You can download OpenSSL on Windows from the following location:

- [https://slproweb.com/products/Win32OpenSSL.html](https://slproweb.com/products/Win32OpenSSL.html)
  [https://slproweb.com/download/Win32OpenSSL-1_0_2g.exe](https://slproweb.com/download/Win32OpenSSL-1_0_2g.exe)

You need to install PERL on the system before using OpenSSL. Refer to the PERL website.
4. Configuring SSL/TLS — Required

4.1. Introduction

This chapter provides detailed steps for setting up certificates for the Secure Sockets Layer (SSL) or Transport Layer Security (TLS).

This chapter contains the following topics for configuring SSL certificates:

- 4.2. SSL/TLS Setup: Server Certificates
  - 4.2.1. Production Environment: CA Certificates
  - 4.2.2. Testing Environment: Self-Signed Certificates

If you already have CA certificates, and the keystore is ready for SSL setup on the server, navigate to the chapter titled 5. Configuring SSL/TLS for Ephesoft Transact — As Needed.

If the server is already set up for certificates, keystore and SSL, navigate to the chapter titled Configuring Ephesoft Transact for SAML SSO.

4.2. SSL/TLS Setup: Server Certificates

To set up SSL/TLS protocols, you need to use either certificates issued by CA or self-signed certificates, which can be used in TEST environment as mentioned above.

4.2.1. Production Environment: CA Certificates

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

4.2.2. Testing Environment: Self-Signed Certificates

For a lab or test environment, you can generate your own dummy self-signed certificates for testing the Ephesoft SSO integration. This can be done using OpenSSL tool, as described below.

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

Perform the following steps to create self-signed certificates using OpenSSL:

1. Locate the OpenSSL CA.pl file, as this file is required to create the dummy CA certificate file.
2. Create a directory to store certificates:
   ```
   mkdir certificates
   ```
3. In Linux, execute the following command:
   ```
   /usr/lib/ssl/misc/CA.pl -newca
   ```
4. In Windows, execute the above command, replacing the path of CA.pl with the Windows path.

This creates demoCA/cacert.pem (CA Certificate) and demoCA/private/cakey.pem (private key).
The generated **cacert.pem** is located inside the **demoCA** folder.

```
post&39;975e0a79e/certificates/[usr]/lib/ss1/misc/cac.pem -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'
Please enter 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): EPM
Organization Name (eg, company): [Internet Vidgits Pty Ltd]:Ephesoft Inc
Organizational Unit Name (eg, section): Epm
Common Name (eg, server FQDN or YOUR name): [Innopcloud.ephesoft.com]
Email Address [emailaddress@ephesoft.com]: 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/ss1/openssl.cnf
Enter pass phrase for `./demoCA/private/cakey.pem'
Check that the request matches the signature
Signature ok, Certificate Details:
  Serial Number: 1126591400006247095 (8b9c5891214aaf5225)
  Validity
    Not Before: Apr 20 14:24:53 2016 GMT
    Not After : Apr 20 14:24:53 2019 GMT
  Subject
    C: US
    ST: CALIFORNIA
    O: Ephesoft Inc
    OU: Innovation
    CN: Innopcloud.ephesoft.com
    emailAddress: enterprise.support@ephesoft.com
  X509v3 extensions:
  X509v3 Authority Key Identifier:
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
Base Updated
post&39;975e0a79e/certificates/
```

**Viewing the cacert.pem (CA Certificate)** [migrated]

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

    `openssl req -newkey rsa:1024 -nodes -keyout newreq.pem -out newreq.pem`
Using the openssl reg Command [migrated]

Note: Make sure to use the same name/value in Common Name as that of the servername/hostname. Otherwise, when trying to access the server, the browser might warn the user that the name does not match the hostname. Also, make sure to access the server with same hostname as mentioned here.

5. Create Server Certificate — that is, sign the certificate CSR (certificate signing request) with CA using the following command:

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

Replace the path of the CA.pl file according to your operating system (Windows/Linux).
Creating the Server Certificate [migrated]

The following files will be created after executing the previous steps.

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

6. For improved clarity, rename these files:

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

The following PEM files will be available after renaming the original files:

- `cacert.pem`
- `servercert.pem`
- `serverkey.pem`

7. Convert the `servercert.pem` file to PKCS12 format (*.p12) using the following command:

   ```bash
   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.

Converting the servercert.pem File to PKCS12 Format

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 the destination keystore password, enter a password and keep it safe. It will be used as keystore password in the 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).

Keytool Command

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.
Creating a java truststore file

These files will be used while configuring SSL/TLS on the Ephesoft Transact server.
5. Configuring SSL/TLS for Ephesoft Transact — as Needed

5.1. Introduction

Use the steps in this section to configure SSL/TLS on Ephesoft, after you have generated the certificates with the previous chapter.

This chapter describes the following topics:

- 5.2. SSL/TLS Setup: Configuring the Ephesoft Transact JavaAppServer (Tomcat)
- 5.3. SSL/TLS Setup: Configuring Ephesoft Transact Files
  - 5.3.1. dcma-batch.properties
  - 5.3.2. dcma-workflows.properties
  - 5.3.3. web.xml

If you do not need to perform any of these SSL/TLS setup tasks, please proceed to the next chapter titled 6. Configuring Ephesoft Transact for SAML SSO.

5.2. SSL/TLS Setup: Configuring the Ephesoft Transact JavaAppServer (Tomcat)

As an example of SSL and TLS setup, this section describes how to use self-signed certificates from the previous chapter and to generate them for testing purposes. If you are using CA certificates in a production environment, provide the details shared by your Certifying Authority.

To configure the Tomcat Connector using CA/Self-Signed Keystore and Truststore certificates, perform the following steps:

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

   Existing HTTP/HTTPS Connector

Comment the existing connector (default port 8080) and create a new connector by manually adding the following properties in the server.xml file:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>port:</td>
<td>8443 (change port number if required)</td>
</tr>
<tr>
<td>maxThreads:</td>
<td>2000</td>
</tr>
<tr>
<td>clientAuth:</td>
<td>false</td>
</tr>
<tr>
<td>scheme:</td>
<td>https</td>
</tr>
<tr>
<td>keepAliveTimeout:</td>
<td>-1</td>
</tr>
<tr>
<td>connectionTimeout:</td>
<td>900000</td>
</tr>
<tr>
<td>secure:</td>
<td>true</td>
</tr>
</tbody>
</table>
**SSLEnabled:** true
**sslProtocol:** TLS
**sessionTimeout:** 30
**truststoreFile:** (truststore file generated previously)
**truststorePass:** <password for truststore>
**keystoreFile:** (keystore files generated previously)
**keystorePass:** <password for keystore>
**maxKeepAliveRequests:** 200

```xml
  port="8443" maxThreads="2000" clientAuth="false" scheme="https"
  keepAliveTimeout="-1" connectionTimeout="900000" compression="on"
  noCompressUserAgents="gozilla, travista"
  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" truststorePass="enter truststore password"
  keystoreFile="enter key store complete path" keystorePass="enter keystore password"
  maxKeepAliveRequests="200" maxPostSize="4194304" />
```

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

**Certificate Settings**

5.3. SSL/TLS Setup: Configuring Ephesoft Transact Files

The following files need to be configured in cases where you are setting up Ephesoft Transact files for SSL/TLS:

- `EPESOFT_HOME/Application/WEB-INF/classes/META-INF/dcma-batch/dcma-batch.properties`
- `EPESOFT_HOME/Application/WEB-INF/classes/META-INF/dcma-workflows/dcma-workflows.properties`
- `EPESOFT_HOME/Application/WEB-INF/web.xml`
5.3.1. dcma-batch.properties

- Modify the `batch.base_http_url` to include the https protocol, correct port and hostname.

  ```
  batch.base_http_url=https://localhost:8443/dcma-batches
  ```

  Modifying `batch.base_http_url`

5.3.2. dcma-workflows.properties

- Modify the `wb.hostURL` to include the https protocol, correct port and correct host.

  ```
  wb.hostURL=https://localhost:8443/dcma/rest
  ```

  Modifying `wb.hostURL`

5.3.3. web.xml

- Take a backup copy of the existing web.xml file.
- Modify the following `<context-param>` entries:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Updated Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>Enter the value matching the value in server.xml. Update this only if you have updated the port number.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;context-param&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>param-name</code>port<code>/param-name</code></td>
</tr>
<tr>
<td></td>
<td><code>param-value</code>8443<code>/param-value</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/context-param&gt;</code></td>
</tr>
<tr>
<td>protocol</td>
<td>Enter HTTPS.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;context-param&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>param-name</code>protocol<code>/param-name</code></td>
</tr>
<tr>
<td></td>
<td><code>param-value</code>https<code>/param-value</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;/context-param&gt;</code></td>
</tr>
</tbody>
</table>

- The Logout URL needs to be of the following format:

  ```
  https://{hostname}:{port-number}/dcma/saml/logout
  ```

  ```
  <!-- Logout URL -->
  <init-param>
  <param-name>logoutUrl</param-name>
  <param-value>https://ephesoftvm:8443/dcma/saml/logout</param-value>
  </init-param>
  ```

  Logout URL
6. Configuring Ephesoft Transact for SAML SSO

6.1. Introduction to SAML

Ephesoft uses Spring Security SAML. Click the following link for more information:

- [https://projects.spring.io/spring-security-saml/](https://projects.spring.io/spring-security-saml/)

This section provides an overview of the Security Assertion Markup Language (SAML), with additional information where it applies to Ephesoft Transact.

- **SAML Overview** — Security Assertion Markup Language (SAML) is an open standard for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider.

- **Identity Provider Overview** — “An identity provider offers user authentication as a service. Relying party applications, such as web applications, outsource the user authentication step to a trusted identity provider. Such a relying party application is said to be federated, that is, it consumes federated identity. An identity provider is “a trusted provider that lets you use single sign-on (SSO) to access other websites.” SSO enhances usability by reducing password fatigue. It also provides better security by decreasing the potential attack surface.”

  Source — [Wikipedia](https://en.wikipedia.org) (8/28/18)

- **Service Provider Overview** — In SAML, a service provider “is a system entity that receives and accepts authentication assertions in conjunction with a single sign-on (SSO) profile of the Security Assertion Markup Language (SAML).”

  Source — [Wikipedia](https://en.wikipedia.org) (8/28/18)

6.2. SAML v2 Support in Ephesoft Transact

Ephesoft Transact supports Spring Security-based SAML SSO. The important components for configuring SSO are as follows:

- **applicationContext-security.xml**: User can configure all Spring Security-related beans from this xml.

- **SAML IdP**: A third-party IdP (Identity Provider) is required to configure the AuthenticationHandler. Metadata from Ephesoft is required to be registered with the IdP.

- **JKS Keystore**: A Keystore needs to be set up for general authentication.

- **Logout URL**: This parameter is required to redirect the user to a page when they click on the Ephesoft logout button.

- **Mode**: This parameter can have two values; **AUTHENTICATION_ONLY** and **AUTHENTICATION_AUTHORIZATION**. Based on this parameter, Ephesoft Authorization is used.

  - If the **AUTHENTICATION_ONLY** parameter is used, then the provided username is used for authorization using configured connectivity.
  - If **AUTHENTICATION_AUTHORIZATION** is used, Ephesoft looks for Roles and IsSuperAdmin attributes in the SAML Response. The attribute names that are passed through SAML response are configurable.
7. Configurations for Integrating SAML-Enabled Spring Security Framework

7.1. Introduction

This section provides the details to configure Ephesoft Transact for SAML SSO. If Ephesoft Transact has already been configured, skip to the section, Configuring the ADFS Server.

If this is a new installation of Ephesoft Transact, you must configure the following files before configuring ADFS with a later topic:

- EPHESOFT_HOME/Application/applicationContext.xml
- EPHESOFT_HOME/Application/WEB-INF/web.xml
- EPHESOFT_HOME/Application/WEB-INF/classes/security/samlKeystore.jks
- EPHESOFT_HOME/Application/WEB-INF/classes/META-INF/applicationContext-security.xml
- EPHESOFT_HOME/Application/js/sessionTimeout/session-timeout.js

7.2. applicationContext.xml

- Uncomment the applicationContext-security.xml resource in applicationContext.xml to use SAML SSO.

7.3. web.xml

- Uncomment the springSecurityFilterChain and its filter mapping.
- Comment out the following elements in the web.xml file:
  - SessionTimeoutFilter and its filter mapping
  - SessionTimeoutServlet and its filter mapping
  - All security-constraints and login-config nodes
  - The session-config node needs to be commented out in the web.xml file.
- Configure the logout URL according to the server details.
  - The logout URL needs to follow this form:
    ```
    https://{hostname}:port/dcma/saml/logout
    ```
• The **authenticationFilter** and its mapping must be prior in sequence to the **authorizationFilter** in the filter chain.

```xml
<!-- Authentication Filter for SSO -->
<filter>
  <!-- authenticationFilter -->
  <filter-name>authenticationFilter</filter-name>
  <filter-class>com.ephesoft.dcm.webapp.AuthenticationFilter</filter-class>
  <!-- Name of HTTP header containing User Name. -->
  <init-param>
    <param-name>requestUsernameHeader</param-name>
    <param-value>REMOTE_USER</param-value>
  </init-param>
  <!-- Name of HTTP header containing Group Name. -->
  <init-param>
    <param-name>requestGroupNameHeader</param-name>
    <param-value>GROUP_USER</param-value>
  </init-param>
  <!-- Name of HTTP header containing Super Admin. -->
  <init-param>
    <param-name>requestSuperAdminHeader</param-name>
    <param-value>SUPER_ADMIN</param-value>
  </init-param>
  <!-- Logout URL -->
  <init-param>
    <param-name>logoutUrl</param-name>
    <param-value>https://ephesoftvm:8443/dcm/saml/logout</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>authenticationFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
</filter>
```

**Authentication Filter**

• Select the appropriate value for **authenticationType** as per the following logic:
  o 1: If only the username will be passed along as attribute in the SAML message, and authorization will be determined by Ephesoft Transact.
  o 2: If the username, group and isSuperAdmin values will be passed along as attributes in the SAML message.

```xml
<!-- Parameter added for choosing authentication type from multiple authentication type supported by Ephesoft
  Ephesoft Authentication, 1:SSO authentication only, 2:SAML authentication and authorization with session variables -->
<context-param>
  <param-name>authenticationType</param-name>
  <param-value>1</param-value>
</context-param>
```

**Authentication Type**

If authenticationType=1 (i.e. Authentication Only), roles need to be defined as per the property defined in `user-connectivity.properties`.

(<Ephesoft_Installation_directory>\Application\WEB-INF\classes\META-
INF\dcma-user-connectivity\user-connectivity.properties). The user.connection property defines where the roles will be mapped.

If user.connection=0, roles for the user will be mapped in LDAP
user.connection=1, roles for the user will be mapped in AD
user.connection=2, roles for the user will be mapped in tomcat-users.xml(<Ephesoft_Installation_directory>\JavaAppServer\conf\tomcat-users.xml)

If authenticationType=2 (i.e. Authentication and Authorization are done by IdP) and the group is not received from ADFS server, then, by default the system will use the group defined in default_group property of application.properties
(<Ephesoft_Installation_directory>\Application\WEB-INF\classes\META-INF\application.properties).

```plaintext
# Group to be used for SSO if group name is not provided along with the request
default_group=admin
```

User Connection Properties

Note: Mapping roles for the user in AD is a feature that is only available in Ephesoft Transact 4.5.0.2 and later releases. This AD role-mapping function is not supported in releases prior to Release 4.5.0.2.

7.3.1. Multiple Group Support in web.xml

By default, when using SSO, Ephesoft Transact presumes that every user will be associated with a single group. To enable a single user with multiple groups for authentication Type = 2, certain configuration changes are required.

Follow the steps in the following Ephesoft Wiki article to make such configurations for multiple groups.

- SAML SSO | Multiple Groups Support
  https://ephesoft.com/docs/configurations/user-connectivity/saml-sso-multiple-groups-support/

7.4. samlKeystore.jks

A Java Keystore (JKS) is a repository of security certificates. SAML exchanges involve usage of cryptography for signing and encryption of data. This keystore will contain a certificate and private key that will be used to digitally sign the SAML messages and to encrypt their content.

A default keystore with the name samlKeystore.jks is provided in the application. However, the user can also create a custom keystore.

7.4.1. Create a private key

1. Open the command prompt at key tool path and run the following command for generating the private key for customKeystore.

   ```cmd
   C:\Program Files\Java\jdk1.7.XXXX\bin>keytool.exe -genkeypair -alias “alias for private key” -keypass “password for key” -keystore “path to keystore”
   ```
2. Enter “nalle123”.

3. Complete the following prompts as applied to your organization.
   After confirming all the added information, the private key will be added to the keystore.
   Place this newly created customKeystore.jks file at the following location:
   “EPESOFT_HOME/application/WEB-INF/classes/security/”
   a. Update the applicationContext-security.xml file as described below.
   
   **Note:** Ephesoft recommends that you record separately the alias for the private key created and corresponding password. These are required for an upcoming task.

### 7.5. applicationContext-security.xml

1. In the bean “epheSamlFilter”, update the value of the constructor argument which you have added in IdP, as per the below logic. The third constructor argument must be true for superadmin access, otherwise this must be false.

   If this is from IdP, a claim name can be passed to indicate whether the user is superadmin or not. That claim name can be specified here.
   - Index=0: Path of Username Attribute passed in SAML message.
   - Index=1: Path of Group Attribute passed in SAML message.
   - Index=2: Path of IsSuperAdmin Attribute passed in SAML message.
   
   **Note:** If you are using `authenticationType=2`, the constructor arg 2 should be false. Otherwise, it grants access to all users as super admin.

   ```xml
   <bean id="epheSamlFilter" class="com.ephesoft.dcma.webapp.SamlFilter">
     <constructor-arg index="0" value="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"/>
     <constructor-arg index="1" value="http://schemas.microsoft.com/ws/2008/06/identity/claims/role"/>
     <constructor-arg index="2" value="true"/>
   </bean>
   ```

2. In the beans “successRedirectHandler” and “failureRedirectHandler”, the user can set URLs for successful and failed authentications respectively. By default, it is configured as `/home.html`.

3. In the bean “keyManager”, the constructor arguments need to be populated according to the following logic:
   - **First constructor argument:** the path of keystore needs to be added here (Default: classpath:security/samlKeystore.jks). If you have created new keystore, specify that keystore.
   - **Second constructor argument:** the password of keystore needs to be added here (Default: nalle123).
   - **Third constructor argument:** the private key and corresponding password created in the previous section need to be added in the map constructor (Default: `<entry key="apollo" value="nalle123" />
              `).
   - **Fourth constructor argument:** the private key to be used needs to be provided here (Default: apollo).
Note: You can add more than one private key in the map constructor in the third argument. Just mention which key you need to use for the current scenario in the fourth argument.

Note: Java keystore Information and respective keystore passwords can be encrypted. The complete procedure is described in the following document:

- *How to Encrypt Passwords in Ephesoft Files*

To use encryption, you also need to uncomment the following commented bean set in the applicationContext-security.xml file.

```xml
<!-- Uncomment following lines to use encrypted passwords. -->
<bean id="certStorePasswordDecryptor" class="com.ephesoft.dcm.encryption.cert.PasswordDecryptor">
  <property name="encryptedString" value="EncryptedPassword"></property>
</bean>

<bean id="certPasswordDecryptor" class="com.ephesoft.dcm.encryption.core.PasswordDecryptor">
  <property name="encryptedString" value="EncryptedPassword"></property>
</bean>

<bean id="keyManager" class="org.springframework.security.saml.key.JKSKeyManager">
  <constructor-arg value="classpath:security/samlKeystore.jks" />
  <constructor-arg type="java.lang.String" value="#{certStorePasswordDecryptor.decryptedString}" />
  <constructor-arg type="java.lang.String" value="apollo" />
</bean>
```

**certStorePasswordDecryptor Bean**

Here, in the beans “certStorePasswordDecryptor” and “certPasswordDecryptor”, replace the string “EncryptedPassword” with your encrypted password value, keeping the quotation marks (example: “nROcbAKcSGR7CsrINVZSA==”).

Then, comment out the similar bean “keyManager” right below the above-mentioned lines:
4. Update the “metadataGeneratorFilter” bean as shown below. Perform these steps to configure the metaGeneratorFilter.

   a. This task requires the entityId, which the customer must provide to Ephesoft. The customer security administrator typically provides the entityId.

   An entityId is a globally unique name for a SAML entity. Specific requirements for the entityId differ according to the requirements of the specific identity provider.

   Update the entityId with the value that pertains to the user's metadata, which is to be uploaded at the identity provider (IdP).

   Update the entityBaseURL with hostname and port number values. This will be the public DNS address, and contains https:// at the beginning and /dcma at the end.

```xml
<bean id="metadataGeneratorFilter" class="org.springframework.security.saml.metadata.MetadataGeneratorFilter">
  <constructor-arg>
    <bean id="keyManager" class="org.springframework.security.saml.key.JKSKeyManager">
      <constructor-arg>
        <map>
          <entry key="apollo" value="nalle123" />
        </map>
      </constructor-arg>
    </bean>
    <!-- Enter KeyStore Location -->
    <constructor-arg type="java.lang.String" value="apollo" />
  </constructor-arg>
  <!-- Enter KeyStore Location -->
  <bean id="mapKeyStore" class="org.springframework.security.saml.key.KeyStore">
    <constructor-arg>
      <map>
        <entry key="apollo" value="nalle123" />
      </map>
    </constructor-arg>
  </bean>
  <bean class="org.springframework.security.saml.metadata.MetadataGenerator">
    <property name="entityId" value="urn:super:ephesoft:ggn" />
    <property name="requestSigned" value="true" />
    <property name="entityBaseURL" value="http://{hostname}:{portNumber}/dcma" />
    <property name="extendedMetadata" class="org.springframework.security.saml.metadata.ExtendedMetadata">
      <property name="signMetadata" value="true" />
      <property name="idpDiscoveryEnabled" value="false" />
    </property>
  </bean>
</bean>
```

5. Configure the “metadata” bean. Metadata configuration can be done in one of the following ways:

   a. Save IdP metadata in a file and place it in the following folder:
b. Update the bean as shown below and specify the classpath as “/security/<filename.xml>”.

c. To disable signature verification, set the `metadataTrustCheck` property to `false`.

```xml
<bean id="metadata"
class="org.springframework.security.saml.metadata.CachingMetadataManager">
  <constructor-arg>
    <list>
      <bean class="org.springframework.security.saml.metadata.ExtendedMetadataDelegate">
        <constructor-arg>
          <bean class="org.opensaml.saml2.metadata.provider.ResourceBackedMetadataProvider">
            <constructor-arg>
              <value type="java.lang.String">/security/FederationMetadata.xml</value>
            </constructor-arg>
            <constructor-arg>
              <value type="int">5000</value>
            </constructor-arg>
          </bean>
        </constructor-arg>
        <property name="parserPool" ref="parserPool"/>
      </bean>
      <bean class="org.springframework.security.saml.metadata.ExtendedMetadata">
        <property name="requireLogoutResponseSigned" value="true"/>
      </bean>
      <property name="metadataTrustCheck" value="false"/>
    </list>
  </constructor-arg>
</bean>
```

d. Using the IdP metadata URL, modify the “`metadata`” bean and use `ExtendedMetadataDelegate` and `HttpMetadataProvider` as shown below. Also, define the `metadataTrustCheck` as `false` to skip signature validation. Update the metadata URL to the IdP metadata URL that is provided in the following code:

```xml
<bean id="metadata"
class="org.springframework.security.saml.metadata.CachingMetadataManager">
  <constructor-arg>
    <list>
      <bean class="org.springframework.security.saml.metadata.ExtendedMetadataDelegate">
        <constructor-arg>
          <bean class="org.opensaml.saml2.metadata.provider.HttpMetadataProvider">
            <constructor-arg>
              <value type="java.lang.String">https://ADFSServer/FederationMetadata/2007-06/FederationMetadata.xml</value>
            </constructor-arg>
            <value type="int">5000</value>
          </bean>
        </constructor-arg>
        <property name="parserPool" ref="parserPool"/>
      </bean>
    </list>
  </constructor-arg>
</bean>
```
6. **SHA-256 Support**: By default, the application is configured to provide SAML configuration with SHA-1. To configure SHA-256, comment the bean for SHA-1 and uncomment SHA-256 bean.

```xml
<constructor-arg>
  <bean class="org.springframework.security.saml.metadata.ExtendedMetadata">
    <property name="requireLogoutResponseSigned" value="true"/>
  </bean>
</constructor-arg>

<property name="metadataTrustCheck" value="false"/>
</bean>
</list>
</constructor-arg>
</bean>

7.6. Testing at the SSOCircle

Perform the following steps to test Ephesoft at the SSOCircle (IdP).

1. Create an account at the SSOCircle and log in.
2. Add the service provider’s metadata in the SSOCircle.

You can download Ephesoft metadata using the following path:

   - [http://{ephesoft_ip}:{port}/dcma/saml/metadata](http://{ephesoft_ip}:{port}/dcma/saml/metadata)
   - The entityID should be the FQDN that is registered at the SSOCircle.
   - The SSOCircle metadata is available from the following online resource:
     - [http://idp.ssocircle.com/idp-meta.xml](http://idp.ssocircle.com/idp-meta.xml)

7.7. Configuring Ephesoft Transact and ADFS

This section contains the following tasks or topics:

- **Configuring Ephesoft Transact for the ADFS Server**
  - Obtain the ADFS Metadata
  - Import the ADFS Certificates in Ephesoft Transact

- **Configuring the ADFS Server for Ephesoft Transact**

- **Exporting the Certificate from the ADFS Server**

- **Successful Configuration**

7.7.1. **Configuring Ephesoft Transact for the ADFS Server**

Before completing these steps, ensure that the following prerequisites are in place:

- Ensure that Ephesoft Transact has already been configured on HTTPS.
- Ensure that the Ephesoft Transact server is running prior to downloading metadata.
In the case that a broken page is displayed while downloading Ephesoft metadata, this means that the Ephesoft server did not start correctly. Resolve any errors that caused a failure by using the following log file:

- `<Ephesoft_HOME>/Application/logs/dcma-all.log`

Follow these steps to configure ADFS settings on Ephesoft Transact:

**Obtain the ADFS Metadata**

1. Download the ADFS metadata from the default URL:
   - `https://<ADFSServer>/FederationMetadata/2007-06/FederationMetadata.xml`
   No Ephesoft-related configuration is required for downloading the ADFS metadata.
2. Copy the downloaded configuration at the following path:
   - `<Ephesoft_Home>/Application/WEB-INF/classes/security`

**Import the ADFS Certificates to Ephesoft Transact**

To export the certificate from ADFS, view the section [Exporting Certificate from the ADFS Server](#).

To import ADFS certificates, use the following command:

```
keytool.exe -importcert -alias adfssigning -keystore samlKeystore.jks -file adfsCertificate.cer
```

### 7.7.2. Configuring the ADFS Server for Ephesoft Transact

Follow these steps to set up the ADFS Server for operation with Ephesoft Transact.

1. Download the Ephesoft Transact metadata from the following path:
   - `https://<ephesoft_ip:port>/dcma/saml/metadata`

2. Add the **Relying Party Trust** with the following steps:
   a. In ADFS, navigate to the following folder:
      - `ADFS\Trust Relationships`
   b. Right click the **Relying Party Trusts** folder under `ADFS\Trust Relationships`. The following snapshot illustrates this navigation.
Relying Party Trusts Folder

c. From the popup menu, click **Add Relying Party Trust...**

   The **Add Relying Party Trust Wizard** appears.
3. Choose (select) the **Import data about Relying Party from a file** option and select the metadata XML file saved in step 1.

4. Provide a display name and click **Next** by choosing the default setting.
5. Click **Finish** to complete this part of the setup. The **Edit Claim Rules** dialog should open automatically. Alternatively, you can right click the added relying party name and select **Edit Claim Rules**.

6. Add the following claims.
   a. **NameId**: Sends the **Name ID**.
      i. Rule Name: **NameId**
      ii. Choose "**SAM-Account-Name**" as the LDAP Attribute and "**Name ID**" as the Outgoing claim type.

   b. **FirstName**: Sends name details.
      i. Rule Name: **FirstName**
      ii. Choose "**SAM-Account-Name**" as the LDAP Attribute and "**Name**" as outgoing claim type.

   c. **Group**: Sends group details.
      Define this rule as per your configuration. For example, the user can be a member of multiple groups. The service provider expects a single group — namely ‘**Enterprise**’.
      Sending all groups will be irrelevant to the service provider.
      A filter can be defined for this scenario by the two following rules:
      i. This rule retrieves all groups and adds them as an incoming claim for the next rule. The output of this claim will be used by the next rule for processing. Add Rule Name as ‘**Get all groups user belongs**’. Define the Custom Rule as follows:

ii. Filtering Groups — Define a filter to restrict groups sent in claims.
   − Select the Pass Through or Filter an Incoming Claim template.
   − Add the Rule Name as Filter Groups.
   − In the field, Pass through all claims that start with specific value, specify Enterprise.

   The group claim rules need to be configured only if authenticationType=2 is set in the web.xml file.

   In the web.xml file, if the authenticationType is 1 (that is — SSO Authentication Only), then Authorization will be handled by Ephesoft Transact. ADFS just needs to send the username as a claim to Transact.

   Note: The constructor argument value, which is cited in the topic applicationContext-security.xml, can be found in ADFS. While editing claim rules, you can find the claim name by selecting the View Rule Language button. You can copy the claim name from the shown rule and paste directly from ADFS into the Ephesoft Transact epheSamlFilter to avoid mistakes.

7. Open the Relying Party Trust by double clicking it. In the Advanced tab, change Secure Hash Algorithm to SHA-1/SHA-256, to be as configured in Ephesoft Transact.

7.7.3. Exporting the Certificate from the ADFS Server

1. In ADFS, select the Certificate option under ADFS/Service.

2. Double click the certificate to export. Navigate to the Details tab, click Copy to File and save this as a DER-encoded certificate in your file system.

   Note: The exported certificates can be saved to any accessible path on your file system. This exported certificate from ADFS needs to be imported in samlKeystore.
Certificate > Details Tab

Certificate Export Wizard
7.7.4. Successful Configuration

When accessing the Ephesoft Transact page, the user will be redirected to the ADFS login screen.

![ADFS Login Screen](image)

On successful authentication, the user will be redirected to the Ephesoft Transact page.

7.8. Configuring Okta with Ephesoft Transact

7.8.1. Introduction

This section contains four tasks, as follows:

**Okta setup**
- Creating the Okta Developer Account
- Assigning People in Okta to Ephesoft Transact
- Adding People to Okta

**Ephesoft Transact setup**
- Configuration in Ephesoft Transact

7.8.2. Creating the Okta Developer Account

Perform the following steps to configure Okta for inter-operation with Ephesoft Transact.

1. Create a developer account at Okta if you have not already done so. Use the following resource:
   - [https://www.okta.com/developer/signup/](https://www.okta.com/developer/signup/)
2. Launch Okta.
3. Click the **Admin** tab and click **Add Applications**.
4. Click the Create New App button.

5. Select SAML 2.0 and then click Create.

6. Configure the settings for the new SAML 2.0 application.
   a. Enter General Settings and click Next.
7. Enter **SAML Settings** and define the following fields.

    **SAML Settings** screen

    a. **Single sign on URL** should be of the following form:
       - `http://{hostname}:{portNo}/dcma/saml/SSO`

    b. **Audience URI (SP Entity ID)** should be any unique Entity ID. Please note down its value as it will be needed while configuring Ephesoft Transact.

    c. In **Attribute Statements**, the **username** must be added. It is necessary for authorization in Ephesoft Transact. Note down the attribute name you have given in **Name tag** as it is necessary in a later task, while configuring Ephesoft Transact.
8. Click **Show Advanced Settings** and check (enable) **Enable Single Logout**. Define additional settings as follows:

![Advanced Settings in SAML 2.0 setup (deliberately enlarged)](image)

- **Single Logout URL** — must be in the following form:

- **SP Issuer** — the issuer for the service provider and must be in the following form:
  - `http://<hostname>:<port>/dcma/saml/metadata`

- **Signature Certificate** — use the following steps to upload the public key certification exported from the samlKeystore.jks file:
  
  i. Click **Next**.
  
  ii. In **Step 3 Feedback**, check (enable) the setting for:

  **I’m an Okta customer adding an internal app.**

  iii. Click **Finish**.
Select a reason for this SAML 2.0 app

9. Once configuration is completed successfully, the following screen appears with Sign On settings:

Sign On settings for the SAML 2.0 App

10. Copy the URL. This is required later when configuring Ephesoft Transact.

   Because Okta metadata is hosted at https, the public key needs to be imported in JRE cacerts. Perform the following steps to import the public key into cacerts:
   a. Navigate to the https URL of the Okta app.
   b. Press Ctrl+Shift+I (the letter I as in India) and navigate to the Security tab.
Security tab

c. Click **View Certificate**. Browse to the **Details** tab. Click **Copy to File**.

Save this as a DER-encoded certificate in your file system.
d. Import this certificate in cacerts.
   
i. Navigate to `{InstallDir}\Dependencies\jdk\bin` and open the command prompt here.
ii. Run the following command to import the Okta certificate in cacerts.

```
{InstallDir}\Dependencies\jdk\bin>keytool.exe -import -file "path to downloaded okta certificate" -keystore {InstallDir}\Dependencies\jdk\jre\lib\security\cacerts
```

### 7.8.3. Assigning People in the Okta Account to Ephesoft Transact

Perform the following steps to assign people to Ephesoft Transact, which will give them permission to use the application.

1. Click the **People** tab and click **Assign to People**.

![Admin People tab in SAML 2.0 setup](image)

The screen displays the people who are currently in your directory.
2. Click **Assign** for the user you want to assign. Enter the **username** to set the login requirement. This user is now assigned to Ephesoft Transact.

![Viewing usernames for the app](image)

3. Click **Done** to close this screen, or repeat these steps to assign additional people.

### 7.8.4. Adding People to the Okta Account

Perform the following steps to add additional people to your Okta account.

1. Click **Directory** and then select **People**.

![Directory > People navigation](image)

2. Click **Add Person** and then enter details.

![Add Person to provide details as prompted](image)
7.8.5. Configuring Ephesoft Transact to Operate with Okta

To configure Ephesoft Transact for operation with Okta, edit or set up the `applicationContext.xml` and `web.xml` files, as would be required if setting them up for other identity providers (IdPs).

1. In the `tomcat-users.xml` file, add the `username`, `password` and `role` for the user.
   - This is a required setting for authentication and authorization in Ephesoft Transact.
   - The `username` should be the first name of the Okta user.

2. In the `applicationContext-security.xml` file, make the following changes:
   - In the `epheSamlFilter` bean, add the name of the attribute that you added in Okta.
   - In the `metadataGeneratorFilter` bean, enter `entityID` as configured in Okta.
   - In the `metadata` bean, add the URL of the Okta metadata which was copied to an Ephesoft folder location while adding Ephesoft Transact in Okta (a previous procedure).
By default, the application is configured to provide SAML configuration with SHA-1.

For additional information about configuring SHA, refer to the earlier topic:

- **7.5. applicationContext-security.xml**

```
<bean id="certStorePasswordDecryptor" class="com.ephesoft.doma.encryption.core.PasswordDecryptor">
  <property name="encryptedString" value="EncryptedPassword"/>
</bean>

<bean id="certPasswordDecryptor" class="com.ephesoft.doma.encryption.core.PasswordDecryptor">
  <property name="encryptedString" value="EncryptedPassword"/>
</bean>

<bean id="keyManager" class="org.springframework.security.saml.key.PrivateKeyManager">
  <constructor-arg value="classpath:security/samlKeystore.jks"/>
  <constructor-arg type="java.lang.String" value="#(certStorePasswordDecryptor.decryptedString)!"/>
</bean>
```

**certStorePasswordDecryptor bean**

- Here, in the beans “certStorePasswordDecryptor” and “certPasswordDecryptor”, replace the string “EncryptedPassword” with your encrypted password value, keeping the quotation marks (example: “nROcbAKcSGR7CsxrINVZSA==”).

Also, comment out the similar **keyManager bean**, which is immediately below the above-mentioned lines:
3. Save all above changes in the `applicationContext-security.xml` file.

4. Start the Ephesoft Transact server. On navigation to any of the Ephesoft Transact URLs, the user gets redirected to the Okta Sign In page.

5. With successful Okta authentication, the user will be redirected to the Ephesoft Transact login page.
8. Exporting the Self-signed Certificate from Keystore

- Open the Key Tool in the command prompt. (Key tool is available in the JDK.)
- Execute the following command:

  `keytool -export -alias <name of the entry to process> -storepass <password of keystore> -file <ephesoft.cer> -keystore <path to keystore file>`

9. Troubleshooting SSO in Ephesoft Transact

9.1. Introduction

This section describes possible SSO misconfigurations that could arise during SSO setup, with instructions to resolve each issue. This chapter contains the following sections:

- **9.1. Introduction**
- **9.2. General Troubleshooting Tasks** — Use this to diagnose and resolve any one of 9 potential SSO configuration issues
- **9.3. Logging for SAML Requests** — Use this to obtain logging information about SAML requests
- **9.4. Getting the Fiddler Trace** — Follow these steps to use Fiddler, which is a free web debugging proxy from Telerik

9.2. General Troubleshooting Tasks

General troubleshooting may entail one or more of the following configuration issues or activities:

<table>
<thead>
<tr>
<th>Error:</th>
<th>Description and Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Server Startup issues</td>
<td><strong>Issue</strong> — This issue may occur on server startup. Errors are observed with the PKIX path building having failed:</td>
</tr>
<tr>
<td></td>
<td><code>sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target</code></td>
</tr>
<tr>
<td></td>
<td><strong>Resolution</strong> — This issue originates with misconfiguration of certificates.</td>
</tr>
<tr>
<td></td>
<td>• Your https endpoint certificates presented by the IdP could not be verified.</td>
</tr>
<tr>
<td></td>
<td>• Please validate if the IdP certificates have been imported properly in cacerts.</td>
</tr>
<tr>
<td>2. IdP Login Page not displaying</td>
<td><strong>Issue</strong> — With this issue, the user is not redirected to the IdP login page upon opening any screen.</td>
</tr>
<tr>
<td></td>
<td><strong>Resolution</strong> — In the <code>web.xml</code> file, determine if the <code>springSecurityFilterChain</code> is not commented. Also, verify that the filter and its mapping precede the</td>
</tr>
<tr>
<td></td>
<td><code>authenticationFilter</code>.</td>
</tr>
<tr>
<td>3. Access Denied screen shows despite successful authentication</td>
<td><strong>Issue</strong> — With this issue, the user is getting successfully authenticated with the IdP. Although the user has privilege to access the screen, the <code>Access Denied</code> page is shown in error.</td>
</tr>
<tr>
<td></td>
<td><strong>Resolution</strong> — In the <code>web.xml</code> file, verify if the <code>authenticationFilter</code> and its mapping is defined prior to the <code>authorizationFilter</code>.</td>
</tr>
</tbody>
</table>
### 4. User is redirected to a broken page

**Issue** — With this issue, the user gets successfully authenticated on the IdP, but is redirected to a broken page.

**Resolution** — The broken page is displayed in cases where there are some missing configurations.
- Verify that the `entityBaseURL` property is specified in the `metadataGeneratorFilter`.
- If ADFS is configured, verify that the NameID claim is being passed in SAML message. This is mandatory to be passed for ADFS.

### 5. No redirection to the SSO Login screen upon login to Transact, with Access Denied message

**Issue** — With this issue, and after configuring SSO, the Ephesoft Transact page is not redirected to the SSO login page. An **Access Denied** error is displayed.

**Resolution** — Follow these steps as a possible resolution to this issue:
1. Verify that the `applicationContext-security.xml` resource is uncommented in the `applicationContext.xml` file (`<Ephesoft Installation Directory>\Application`). The following snapshot illustrates this configuration:

   ```xml
   <!-- Uncomment to Use SAML SSO -->
   <import resource="classpath:/META-INF/applicationContext-security.xml"/>
   
   Uncommenting applicationContext-security.xml in applicationContext.xml file
   ```

2. Verify that the `springSecurityFilterChain` filter is uncommented in the `web.xml` file:

   ```xml
   <Ephesoft Installation Directory>\Application\WEB-INF
   
   The following snapshot illustrates this configuration:
   ```

   ```xml
   <filter>
   <filter-name>springSecurityFilterChain</filter-name>
   <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
   </filter>
   
   <filter-mapping>
   <filter-name>springSecurityFilterChain</filter-name>
   <url-pattern>/*</url-pattern>
   </filter-mapping>
   
   springSecurityFilterChain filter in web.xml file
   ```

   **Note**: The `springSecurityFilterChain` filter is responsible for redirection to the SSO login page.

3. Ensure that the `springSecurityFilterChain` filter is the first filter in the `web.xml` file. If it is placed at the end, move it to be the first filter.

### 6. Broken page does not correspond to the sent message

**Issue** — In this issue, the broken page in the `InResponseToField` of the response does not correspond to the sent message.

**Resolution** — Verify that the same URL is used for sending the request and the receiving the response. Typically, this issue arises when the authentication request is initialized from a localhost address or http scheme, while the response is received at a public host name or https scheme.

### 7. Broken page appears after idle period

**Issue** — In this issue, an application redirect to a broken page is shown after the system is idle for a period of time.

**Resolution** — Verify the following settings to resolve this behavior:
1. Comment the `session-timeout` node defined in the `web.xml` file.
2. In the applicationContext-security.xml file, add a property `maxAuthenticationAge` to the bean class `WebSSOProfileConsumerImpl`:
   - **The maxAuthentication age** (in seconds) determines the maximum session time set for the IdP.
   - The default value is 7200 seconds.
   - This value should match the value that was set at the IdP.

<table>
<thead>
<tr>
<th>8. SuperAdmin privileges are granted to non-SuperAdmin users</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Issue</strong> — In this issue, users that do not have the superadmin role may be granted superadmin privileges.</td>
</tr>
<tr>
<td><strong>Resolution</strong> — Verify the index=2 setting in the <code>epheSamlFilter</code> bean in the applicationContext-security.xml file. Determine if this value is <strong>true</strong>.</td>
</tr>
<tr>
<td>- If set to <strong>true</strong>, it grants all users with superadmin privileges.</td>
</tr>
<tr>
<td>- To resolve this issue, modify the value to either <strong>false</strong> or an empty string.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9. General Log Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>If there are any issues encountered during configuration or authentication, please check the logs in the following location:</td>
</tr>
<tr>
<td>- <code>&lt;Ephesoft-Installation-Directory&gt;/Application/logs/dcma-all.log</code></td>
</tr>
<tr>
<td>Follow the instructions defined inside the logs. If the problem persists, share the following information with Ephesoft Support:</td>
</tr>
<tr>
<td>- INFO level logs</td>
</tr>
<tr>
<td>- SAZ file — Collect the Fiddler trace while accessing your application and authenticating to Azure AD and share this SAZ file.</td>
</tr>
</tbody>
</table>

### 9.3. Logging for SAML Requests

To obtain more logging information about SAML requests, add the following lines to the `log4j.xml` file placed under directory `<Ephesoft_HOME>/Application/log4j.xml`. Adding the following lines will provide INFO-level logs for SAML requests and responses in the `dcma-all.log`.

```xml
<Logger name="org.opensaml" level="INFO" additivity="false">
    <AppenderRef ref="CORE"/>
</Logger>
<Logger name="org.springframework.security.saml" level="INFO" additivity="false">
    <AppenderRef ref="CORE"/>
</Logger>
```

### 9.4. Getting the Fiddler Trace

To use Fiddler, which is a free web debugging proxy from Telerik, navigate to the following page for download and installation:

- [https://www.telerik.com/download/fiddler](https://www.telerik.com/download/fiddler)
  
  **Note:** Account registration is required.

After you have completed Fiddler installation, perform the following steps to start the capture:

1. Go to [http://fiddler2.com/home](http://fiddler2.com/home) (.NET2 or .NET4 is fine).
2. Once installed, go to **Tools > Fiddler Options > Https tab.**
3. Check Decrypt HTTPS traffic, and click Yes for the popup screens, as prompted.
   You will receive a few pop-up screens that are required in order to install the Fiddler root certificate. The Fiddler root certificate allows Fiddler to function as an intermediate agent in the HTTPS session.

4. Click Yes to continue.
5. Click Clear Cache on the Fiddler toolbar.
6. Click Edit > Remove All Sessions.
7. Open your favorite browser. Verify it is capturing data in Fiddler.
8. Reproduce the issue in the browser.
9. Stop the Fiddler trace once it is done.
10. Save the Fiddler trace as an SAZ file.