How to get started with SSL/TLS

Introduction

For establishing and maintain a secure TLS connection each of the parties (Server and Clients) have pairs of Private key + Public certificate.

- **The private key** that should be kept in a secret place
- **The public certificate** can be shared between the parties and used for validating the encrypted messages.

The list of the core artifacts that will be used for establishing a secure TLS connection is the following:

- **Server's public key (certificate)** - it is located on the server-side and is shared with the counterparty during the handshake procedure.
  It is mandatory for FIX acceptors.
- **Server's private key** - it is located on the server-side and shouldn't be shared. The key is used for encrypting the message.
  It is mandatory for FIX acceptors.
- **Client's public key (certificate)** - it is located on the client-side and is shared with the counterparty during the handshake procedure.
  Optional for initiators. Without it, the client will be able to establish a secure connection (1-way TLS) and verify only the server.
  It is specified the initiator can use 2-way TLS authentication and the server will be able to verify the client.
- **Client's private key** - it is located on the client-side and shouldn't be shared.
  Optional for initiators. Should be used in case of 2-way TLS authentication so will be able to verify the client.
- **CA certificate** - Certificate signed by 3rd party and trusted by both Server and Client.
- **Certificate Signing Request (CSR)** - the file that the applicant sends Certificate Authority for getting the CA certificate.
- **Certificate stores** - various containers for certificates. Usually protected with passwords
  Java products expect certificates and keys in their own Java containers (*.jks) and do not use plaintext files (for example in *.pem format).
  - **Java Key Store** - store for certificates containing certificate-keys pairs.
  - **Java Trust Store** - storage for all trusted certificates. Usually, it holds CA certificates.

Supported SSL protocols

There are a lot of SSL protocols. FIXEdge supports SSL v2.0, SSL v3.0, TLS v1.0, TLS v1.1, TLS v1.2, TLS v1.3. By default only TLS v1.0, TLS v1.1, TLS v1.2,TLS v1.3 protocols are enabled in FIX Edge.

It's highly recommended to use TLS v1.1, TLS v1.2, or TLS v1.3 protocol because SSL protocol versions were deprecated (see: Deprecating SSL v3.0)

Recommended TLS v1.1, TLS v1.2 or TLSv1.3 protocols are supported only by OpenSSL version 1.0.1 and higher (see: https://www.openssl.org/news/changelog.html).

How to get a TLS certificate and private key

There are two ways to get the SSL certificate and the private key:

1. Create the self-signed SSL certificate and the private key via OpenSSL.
   The instruction about how to create such a certificate and key can be found here: SSL with self-signed certificates
   This type of certificate is only suitable for the testing period because most clients don't trust self-signed certificates.
2. Prepare a Certificate Signing Request (CSR) from a Certificate Authority (CA), for example, "DigiCert".
   In order to do it, you should create the private key file and then generate CSR.
   You can find the instruction here: https://www.ssl.com/how-to/manually-generate-a-certificate-signing-request-csr-using-openssl/
   During this process you have to provide the following information for CA:
   a. commonName (for example, “CN=B2BITS”)
   b. organizationalUnitName (for example, “OU=Dev”)
   c. organizationName (for example, “O=EPAM Systems”)
   d. localityName (for example, “L=Newtown”)
   e. stateOrProvinceName - should not be abbreviated (for example, “S=Pennsylvania”)
   f. countryName - two-letter ISO code for the country where your organization is located (for example, “C=US”)
   g. emailAddress - an email address to contact the organization. Usually the email address of the certificate administrator or IT department
   CA will create the SSL certificate based on provided information. After that you should install this certificate on your server.

Do not forget to back up your private key and certificate in a secure place. Share private key only with entitled personnel.

SSL with self-signed certificates

Self-signed certificates can be used in some cases (for testing purposes). Before creating certificates, make sure that OpenSSL Toolkit is installed.

The OpenSSL project does not distribute any code in binary form, and does not officially recommend any specific binary distributions. An informal list of third party products can be found on the wiki.

First of all you have to create Certificate Authority (CA) certificate along with its private key. Use the following command file to generate the Private key (line 1) and SSL (Certificate Authority) CA root certificate (line 2) files that can be used to generate further certificates:

gen_ca_cert.bat

```
openssl genrsa -out %1.key 2048
openssl req -x509 -new -key %1.key -days 10000 -out %1.crt -subj "/C=RU/ST=%1/O=EPAM/CN=%2"
```

where:

%1 - file name for certificate (.crt) and private key (.key);

%2 - common name (CN), can match the organization name for the CA certificate.

CA certificate allows is used to create and sign other certificate.

Use the following command file to generate

- Certificate (.crt),
- Private key (.key),
- Encrypted Private key (-enc.key),
- Certificate and Private key in .pfx format.

gen_ss_cert.bat

```
openssl genrsa -out %1.key 2048
openssl rsa -in %1.key -out %1-enc.key -aes256 -passout pass:%1
openssl req -new -key %1.key -out %1.csr -subj "/C=RU/ST=%1/O=EPAM/CN=%3"
openssl x509 -req -in %1.csr -CA %2.crt -CAkey %2.key -CAcreateserial -out %1.crt -days 5000
del %1.csr
del %2.srl
openssl pkcs12 -inkey %1.key -in %1.crt -export -out %1.pfx -password pass:%1
```

where:

%1 - file name for certificate (.crt), private key (.key) and password used for encryption where required;

%2 - file name of CA certificate (.crt) and its private key (.key);

%3 - common name (CN) is used to define the server name which will be used for secure SSL connection. Your SSL certificate is valid only if hostname matches the CN.

Server verification by the client and client verification by the server are possible. Two CA certificates and two certificates signed with that CA certificates respectively are required two utilize both sides verification at the same time. Both types of verification are supported by FIXEdge.

How to validate the SSL certificate

After you got the certificate you can check all information about it using the command line opened in the same folder where the certificate is placed:

```
openssl x509 -in cert.pem -text
```

where *cert.pem* is a certificate name.

You will see the following information:

```
openssl x509 -in cert.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 11159143049149737040 (0x9add3d7ac256e850)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, ST=PA, L=Newtown, O=EPAM Systems, OU=Dev
        Validity
            Not Before: Oct 19 14:46:36 2015 GMT
            Not After : Oct 18 14:46:36 2016 GMT
        Subject: C=US, ST=PA, L=Newtown, O=EPAM Systems, OU=Dev
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    ...
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                61:6F:72:A0:D0:99:36:0D:E3:89:8B:53:7A:4F:12:75:01:CA:E4:B3
            X509v3 Authority Key Identifier:
                keyid:61:6F:72:A0:D0:99:36:0D:E3:89:8B:53:7A:4F:12:75:01:CA:E4:B3
            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha1WithRSAEncryption
         ...
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

How to validate the private key

In order to check the private key you can use the following command:

```
openssl rsa -in key.pem -check
```

where *privateKey.key* is a private key name.

You will see the following information:

```
openssl rsa -in key.pem -check
RSA key ok
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
<actual key here>
-----END RSA PRIVATE KEY-----
```


The certificate and the private key can be merged into the one file (for example, *cert.pem*).

How to configure FIXEdge

The article which describes how to configure FIXEdge for SSL connection can be found here: How to configure built-in SSL support for FIX session in FIXEdge