ONBOARDING GUIDE

eInvoice Webservice

23.08.2023 (VERSION 1.0.1)

Contents

INTRODUCTION

SCOPE

This Document details the technical onboarding process for eFaktura Webservice APIs. It does not detail the use of any webservice APIs.

TARGET AUDIENCE

This document is meant for technical resources onboarding to an eFaktura Wevservice API.

ENVIRONMENTS

All eFaktura Webservice API's are available in two versions. One version for Test and one version for Production. These versions run in their own separate environments to ensure that no data is shared between the versions.

Both Test and Production environments utilizes the same security scheme of firewall restrictions, mTLS authentication and user-bound rights and accessibility.

TEST

The Test environment contains all eFaktura Webservice API's in test-versions, uses data from the test-database and runs on the test-version of the core eFaktura systems. It is wholly separate from the production environment.

As is the case in the Production environment, all data is shared by all users with access to the environment, but processing of the data is restricted by the Users access to the different webservice API's. This enables similar functionality and behaviour in the Test environment compared to the Production environment.

The underlying philosophy of the Test environment is to provide an environment that is as similar to Production-environment as possible to be able to support a wide variety of testing needs. From basic sanity-testing to simulating end-to-end-testing.

PRODUCTION

This is the live version of eFaktura.

USER

The user carries the access to, and the rights within, the different eFaktura Webservice API's. The user will be created and administered by Mastercard Payment Services.

USER CONFIGURATION

EXAMPLE: SOAP-HEADER USING WSSE SECURITY

```
<soapenv:Header>

        <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">

            <wsse:UsernameToken wsu:Id="UsernameToken-1" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">

                <wsse:Username>USERNAME</wsse:Username>

                <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">PASSWORD</wsse:Password>

            </wsse:UsernameToken>

        </wsse:Security>

    </soapenv:Header>
```

FIREWALL WHITELISTING

Access to the eFaktura Webservice API require that we add the source IP-address of any requests to the service in our firewall whitelist. For this purpose, you must provide us with all possible source IP-addresses for requests sent to eFaktura Webservice API.

Test- and Production-environments do not share firewalls. A list of source IP-addresses to be whitelisted must be provided separately for each environment.

IP-ADDRESS RESTRICTIONS

Some restrictions apply to IP-addresses that is to be added to our firewall whitelist.

1. Must not be in any of the private IP-ranges. Read more about private IP-ranges here: https://en.wikipedia.org/wiki/Private_network

2. Very large IP-ranges is not permitted.

3. Shared IP-ranges is not permitted. I.e. where the clients source IP-address may be re-distributed to another party, such as in a cloud-based system.

MUTUAL AUTHENTICATION SCHEME

In eFaktura all Webservice API's use a mutual authentication scheme. Mutual authentication, or two-way authentication, refers to two parties authenticating each other at the same time in an authentication protocol. This is achieved using mTLS.

mTLS

The TLS protocol is mostly used for server-to-client authentication where a server identifies itself using a X.509 certificate on the server-side. In mTLS a client-to-server authentication using a X.509 certificate on the client-side is also required.

This is achieved using a PKI setup where the server-side acts as the Certificate Authority (CA) and Intermediate CA, and issues a signed X.509 certificate to the client based on a Certificate Signing Request (CSR) from the client

The complete trust-chain for the PKI setup is CA Root Certificate -> Intermediate CA Certificate -> Client Certificate where each subsequent certificate after the CA Root Certificate is signed by the previous certificate in the chain. It is important to maintain this chain as you build the trust-store to be used for the client application.

ORDERING A CLIENT CERTIFICATE

The mTLS scheme requires certain information about each party to be exchanged. In order for MPS, acting as Certificate Authority (CA), to issue a client-certificate, the client needs to present the Client attributes and the Clients public key. This must be added to a Certificate Signing Request (CSR) and sent to MPS.

CSR SPECIFICATIONS

The Certificate Signing Request (CSR) must have the following attributes.

1. Must be a PKCS#10 standard Certificate Signing Request
2. Must be an ASCII Base64 PEM-encoded file
3. Keylength of public key must be 2048bit
4. Common Name must be unique and descriptive of the client

NOTE ON USE OF COMMON NAME

The Common Name (CN) is used to easily identify a specific certificate. A well formatted CN will make any handling of the certificates that much easier if we do not have to rely on more hidden attributes of the certificate, for instance the certificate thumbprint, when trying to identify the client certificate.

CONTENTS OF CERTIFICATE FILE FROM MPS

Based on the provided CSR, MPS will provide all the certificates in the trust-chain. This will be provided either as a single .p7b-file. The in the trust-chain can also be provided in separate files contained in a .zip file.

Certificate Type	Common Name	Filename in zip-file
CA Root Certificate	NETS AS Root CA	TrustedRoot.crt
CA Intermediate Root Certificate	NETS AS Intermediate CA	DigiCertCA.crt
Client Authentication Certificate	<client_comon_name>	<client_comon_name>.crt

USE-CASE EXAMPLES

There are several tools available to create and edit ssl/tls files such as key-stores, certficates and certificate requests. The two most commonly used are OpenSSL and Java Keytool. Both of these come as command-based tools that can be installed on your OS.

It is also worth mentioning that Windows has built inn tools and wizards handling many different operations with ssl/tls-files.

In the examples below we will mainly concentrate on how to use OpenSSL and Java Keytool.

In the examples below we have used placeholders whenever the command refers to a filename that you are asked to define. I.e. PRIVATEKEY.key is the placeholder for the name you choose to set for your private-key file.

EXAMPLE: Create a Certificate Signing Request (CSR-file)

A Certificate Signing Request can be created using both OpenSSL and Java Keytool, though the process varies somewhat between the two tools.

OpenSSL

Run the following command in OpenSSL to create a CSR-file and a Private key-file.

>openssl req -newkey rsa:2048 -keyout PRIVATEKEY.key -out MYCSR.csr

Breakdown of command

- openssl is the command for running openssl
- req refers to a certificate signing request
- -newkey creates a new key pair
- rsa:2048 donates the keylength in bits
- -keyout creates file containing the private key

- PRIVATEKEY.key is the filename of the created key-file.
- -out creates the output from the req command, i.e. the CSR-file.
- MYCSR.csr is the filename of the created CSR-file.

When running the command you will be prompted for a number of properties to be tied to your private-key and CSR-file.

Create and confirm passphrase for your private-key.

You should now have two files: PRIVATEKEY.key and MYCSR.csr

Java Keytool

Before you can create a CSR-file using Java Keytool you will have to first create a Java Key Store file. This file will contain your private key that is later used to sign the public key that is part of the CSR-file.

In Java Key Tool, run the following command.

>keytool -genkey -alias YOUR_ALIAS -keyalg RSA -keysize 2048 -keypass YOUR_PASSWORD -keystore YOUR_KEYSTORE_FILE.jks

You will be prompted to add some further properties to the keystore.

- First and Last name: This is the CN of your client. This should be the same as YOUR_ALIAS
- Name of your organizational unit: Name of your department, or similar, within your organization.
- Name of your organization: Name of your organization
- Name of your City or Locality: The name of your City
- Name of your State or Province: The name of the State/Province of your City
- Country code: Two-letter country code

You will be prompted to verify the properties. Reply Yes to confirm the properties. Reply No to change the properties.

After confirming the properties you should have a file named YOUR_KEYSTORE_FILE.jks. This is the file containing your private key and forms the basis for further operations in Java Keytool.

Note: It is a good idea to make a back-up of the original YOUR_KEYSTORE_FILE.jks before you perform further operations on this file. The file is quite sensitive to tampering and having a back-up of the original file will be helpful if you at any point you need a do-over of further operations.

Use the YOUR_KEYSTORE_FILE.jks to create a Certificate Signing Request by running the following command in Java Keytool.

>keytool -certreq -v -alias YOUR_ALIAS -file YOUR_ALIAS_CSR.pem -keystore YOUR_KEYSTORE_FILE.jks

You will be prompted for the password of YOUR_KEYSTORE_FILE.jks.

You should now have a new file: YOUR_ALIAS_CSR.pem. This is the Certificate Signing Request required by MPS to issue your client-certificate.

EXAMPLE: Extract Certificates from P7B file

MPS will supply the certificates in a P7b-file. This is a PKCS#7 standard certificate bundle that will contain the CA Root-, CA Intermediate- and Client Certificates issued by MPS. This file cannot be used directly with OpenSSL or Java Keytool to install the certificates in a keystore. You will need extract each certificate individually using OpenSSL or the built-in tools in Windows to extract and save the certs in .cer-file.

OpenSSL:

Run the following command in OpenSSL.

>openssl pkcs7 -inform PEM -outform PEM -in YOUR_CLIENT.p7b -print_certs > CERTIFICATES.cer

This will create a file; CERTIFICATES.cer, that contains all certificates from the p7b-file in a concatenated file with each certificate in order. Copy each certificate into separate text-files and save them as .cer-files.

Windows 10:

Open the file YOUR_CLIENT.p7b and navigate to the folder Certificates within the file. A list of certificates in the bundle will appear.

Double-click the certificate you wish to export and choose the pane named Details. Click the Copy to File... button.

This will open the Certificate Export Wizard. Select to export as Base64 endoced X.509. Click Next.

Select export location and click Next.

You will be presented with the Completing the Certificate Export Wizard-page. Click Finnish.

The certificate is now exported into the specified export location.

EXAMPLE: Create a PKCS#12 .pfx-file using OpenSSL

Many applications can use a PKCS#12 standard .pfx-file as a local keystore. The file must contain the clients private key, the CA Trust-chain of certificates and the client certificate.

The private-key has already been during the process that created the CSR-file. The Client Certificate is supplied in the p7b-file from MPS. The CA Trust-chain of certificates must be created by concatenating the CA Root Certificate and the CA Intermediate Certificate. Both of these certificates are part of the p7b-file from MPS.

Open the CA Root Certificate and the CA Intermediate Certificate in a text editor. Create a new blank file in a text editor and copy-paste the text in the CA Intermediate file and the text in the CA Root Certificate file into the new blank file. Make sure that the CA Intermediate Certificate is first in the file, followed by the CA Root Certificate. Save this file as CA_TRUST.cer.

You should now have all the files you need to create the PKCS#12 file.

Run the following command in openssl

>openssl pkcs12 -export -in YOUR_CLIENT_CERTIFICATE.cer -inkey PRIVATEKEY.key -out YOUR_CLIENT_CERTIFICATE.pfx -certfile CA_TRUST.cer

You will be prompted for your private key password.

You should now have a new file: YOUR_CLIENT_CERTIFICATE.pfx. This file can be used as a local keystore by some applications as well as be used to install your private key and certificates in Windows Keystore.

EXAMPLE: Building a keystore using Java Keytool

First create a trustchain-file that contains all certificates in the trustchain; CA Root Certificate, CA Intermediate Certificate and Client Certificate. Concatenate the three certificates into one file as described below.

Open the CA Root Certificate, the CA Intermediate Certificate and the Client Certificate in a text editor. Create a new blank file in a text editor and copy-paste the text in the Client Certifcate, then the CA Intermediate file and lastly the text in the CA Root Certificate file into the new blank file. Save this file as CA_TRUST.pem.

Next step is to import trustchain file into the keystore-file that was used to create the CSR-file. Run the following command in Java Keytool.

Import Trustchain file

Note that the -alias property must be the same as the alias used when initially creating the keystore-file.

>keytool -import -keystore YOUR_KEYSTORE_FILE.jks -alias YOUR_ALIAS -file CA_TRUST.pem

Now import each of the certificates CA Root Certificate, CA Intermediate Certificate and Client Certificate separately using the commands below.

Import CA Root Certificate

>keytool -import -alias trustedroot -file TrustedRoot.crt -keystore YOUR_KEYSTORE_FILE.jks

Import CA Intermediate Certificate

>keytool -import -alias intermed -file DigiCertCA.crt -keystore YOUR_KEYSTORE_FILE.jks

Import Client Certificate

>keytool -import -alias client -file YOUR_ALIAS.crt -keystore YOUR_KEYSTORE_FILE.jks

TESTDATA

SHARED ENVIRONMENT

The TEST ENVIRONMENT resources, including data, is shared by all users with access to the ENVIRONMENT. Please take care to not alter any pre-existing data as these might be part of another user's test-cases.

When adding data to the test-environment using the eFaktura API's, take care to not add production data.

EXISTING TESTDATA

There are some pre-existing datasets that are tailored to specific use-cases of a Webservice API. See the user guide for each Webservice API for more information of datasets and use-cases for testing purposes.

ORDERING NEW TESTDATA

If specific test-data is required, please contact your INTEGRATION CONSULTANT and ask for help.