Welcome to ZATCA E-Invoice Java SDK (CLI)

This cli is a Java based tool that utilizes ZATCA’s SDK jar to perform several tasks. Basically, you can use it to validate different e-invoice documents including Standard Invoice, Standard Debit Note, Standard Credit Note, Simplified Invoice, Simplified Debit Note, and Simplified Credit Note. Once validation has run, the tool will display the validation results. This can be a PASS if all validations have passed, otherwise a NOT PASS messf.net
age is displayed along with a list of errors indicating what has happened. Moreover, this tool allows signing invoices, generating invoice hash, generating api requests, generating certificate signing requests (CSR), and generating QR code .

SDK Prerequisite

The SDK requires JAVA to run, please make sure you have a valid Java JRE or SDK, the version must be between 11 and 15.

SDK Installation Steps

Install/Update for Windows

1. Download the sdk.zip file
2. Unzip the sdk.zip file
3. Now, open a command line and point to the root folder of the sdk.
4. Run the ***install.bat***
5. Now, you can start using the fatoora cli. Please run ***fatoora -help*** to get all supported commands

Install/Update for Linux

1. Install [jq] from https://stedolan.github.io/jq/
2. Download the sdk.zip file
3. Run command ***sh ~/.bash_profile***
4. Unzip the sdk.zip file
5. Now, open a command line/terminal and point to the root folder of the sdk.
6. Run command ***install.sh***
7. *Run* ***sh ~/.bash_profile***
8. *Run command* ***cd $FATOORA_HOME***
9. *Run command* ***chmod +x fatoora***

10. Now, you can start using the fatoora cli. Please run ***fatoora -help*** to get all supported commands

● Configuration

fatoora [command] [options]

No.	Command	Description
1	[-csr]	A flag used to generate csr and private key.
2	[-pem]	A flag used to generate csr and private key in pem format.
3	[-privateKey fileName]	The name of the private key output file.
4	[-generatedCsr fileName]	The name of the csr output file.
5	[-csrConfig fileName]	The name of the csr configuration file.
6	[-invoice fileName]	The name of the invoice file.
7	[-qr]	A flag used to generate qr.
8	[-sign]	A flag used to sign an invoice.
9	[-signedInvoice fileName]	The name of the signed invoice output file.
10	[-invoiceRequest]	A flag used to generate an invoice API request.
11	[-apiRequest fileName]	The name of the invoice json request output file
12	[-validate]	A flag used to validate invoice.
13	[-generateHash]	A flag used to generate new hash for the provided invoice.
14	[-nonprod]	A flag pointing to use the csr and private key on a non production server.
15	[-sim]	A flag pointing to use the csr and private key on a simulation server.

16	[-help]	A flag used to display this help menu and exit.

● Generating a Certificate Signing Request (CSR):

Prerequisite:

- **Input:**

There is a CSR configuration properties file which contains all the input needed to generate a CSR. Those include the below:

- common name
- serial number
- organization identifier
- organization unit name
- organization name
- country name
- invoice type
- location address
- industry business category

Also, please note that:

1. You can follow the structure of the properties file by checking the template provided in:

   ***/Data/Input/csr-config-template.properties*.**

2. Also an example is provided in:

/Data/Input/csr-config-template.properties/csr-config-example.properties

Once done you can generate a csr using the config properties file by providing it as an option to the -csrConfig argument. Please see Arguments and Options below.

Arguments and Options:

fatoora -csr -csrConfig fileName -privateKey fileName -generatedCsr fileName -pem

No.	Argument(s)/Option(s)	Description
1	[-csr]	A flag used to generate csr and private key.
2	[-csrConfig <filename>]	This is a mandatory argument to provide location and name of the csr configuration file.
3	[-privateKey] <filename>	This is an optional argument to provide location and name of generated private key output file
4	[-generatedCsr] <filename>	This is an optional argument to provide location and name of generated csr output file
5	[-pem]	This is an optional argument used to generate csr and private key in pem format

Notes:

1. If the user does not provide [-privateKey] argument then the private key file will be generated in the path of a running command prompt with file convention name pattern generated-private-key-yyyyMMddhhmmss.key
2. If the user does not provide [-generatedCsr] argument then the csr file will be generated in the path of a running command prompt with file convention name pattern generated-csr-yyyyMMddhhmmss.csr
3. If the user does not provide [-pem] argument then the csr file will be generated encoded base64 and private key file will be generated without header "-----BEGIN EC PRIVATE KEY-----" and footer "-----END EC PRIVATE KEY-----"

● Signing and Generating Invoice Hash:

Prerequisite:

Please refer back to the prerequisites section of generating a CSR above.

- Input: The E-Invoice XML file full path.
- Output: an object of type "Result" that contains
  - IsValid : The final Status of the process.
  - ResultValue: XML Hashed Generated if "IsValid = True". It should be the same as the one already exists in the E-Invoice XML file.
  - ErrorMessage: The error message if "IsValid = false".
- Integration with other systems
  - Declare object from Class "HashGenerationService"
    - Declare object from class “SigningService”
      - Call function "generateInvoiceHash"
        - Parameter 1 (`xmlFilePath`) : E-Invoice XML file full path

Arguments and Options:

fatoora -sign -invoice <filename> -signedInvoice <filename>

No.	Argument(s)/Option(s)	Description
1	[-sign]	A flag used to sign an invoice.
2	-invoice <filename>	This is an optional argument to provide the name of the invoice file.
3	[-signedInvoice <fileName>]	This is an optional argument to provide the name of the signed invoice output file.

Notes:

1. If the user does not provide [-signedInvoice] argument then the signed invoice file will be generated in the path of a running command prompt with file convention name pattern <same input file name>_signed.xml

● Generating JSON API Request:

Prerequisite:

None.

Arguments and Options:

fatoora -invoice <filename> -invoiceRequest -apiRequest <fileName>

No.	Argument(s)/Option(s)	Description
1	-invoice <filename>	This is a mandatory argument used to provide the name of the invoice file.
2	[-invoiceRequest]	A flag used to generate an invoice API request.
3	[-apiRequest <fileName>]	This is an optional argument used to provide the name of the invoice json request output file

Notes:

1. If the user does not provide [-apiRequest] argument then the invoice request file will be generated in the path of a running command prompt with file convention name pattern generated-json-request-yyyyMMddhhmmss.json

● Generating QR Code:

Prerequisite:

Replace SDK default private key and certificate with the custom values, as mentioned below in section “Adding the private key and certificate”.

Arguments and Options:

fatoora -qr -invoice <filename>

No.	Argument(s)/Option(s)	Description
1	[-qr]	A flag used to generate qr code.
2	-invoice <filename>	This is a mandatory argument used to provide the name of the invoice file.

● Validating an invoice:

Prerequisite:

Replace SDK default private key and certificate with the custom values, as mentioned below in section “Adding the private key and certificate”.

Arguments and Options:

fatoora -validate -invoice <filename>

No.	Argument(s)/Option(s)	Description
1	[-validate]	A flag used to validate the invoice.
2	-invoice <filename>	This is a mandatory argument used to provide the name of the invoice file.

● Generating Invoice Hash:

Prerequisite:

None.

Arguments and Options:

fatoora -generateHash -invoice <filename>

No.	Argument(s)/Option(s)	Description
1	[-generateHash]	A flag used to generate new hash for the provided invoice.
2	-invoice <filename>	This is a mandatory argument used to provide the name of the invoice file.

● Adding the Private Key and Certificate:

1. Replace Data/Certificates/ec-secp256k1-priv-key.pem with your private key using EC secp256k1 algorithm without header "-----BEGIN EC PRIVATE KEY-----" and footer "-----END EC PRIVATE KEY-----"

2. Place yourCertificate generated by ZATCA into Data/Certificates/cert.pem *without* header, footer and without any new lines "\n".

3. Once done, now you can sign your xml invoice using your private key and matching it on validation with the zatca certificate.

Please note that the provided certificate
and private key in the SDK are dummy and for testing purposes only.

● Technical FAQs:

- **What JAVA version should I install before using the SDK?**

The prerequisite is using the Java SDK (JAR) versions >=11 and <15.

- **What should the user do when faced with a JAVA error?**

When faced with a JAVA error, the user needs to install JAVA (versions >=11 and <15) before running and using the SDK.

Jar File

Target Java: jdk version start from 11 till 15

1- Generating CSR

- Input: The config properties file path contains following:
  - common name
  - serial number
  - organization identifier
  - organization unit name
  - organization name
  - country name
  - invoice type
  - location address
  - industry business category
- Output: an object of type " CsrGeneratorDto" contains that:
  - csr: The final file will be generated from process
  - privateKey: The final file will be generated from process

2- Generating Hashing

- Input: The E-Invoice XML file full path.
- Output: an object of type "Result" that contains
  - IsValid : The final Status of the process.
  - ResultValue: XML Hashed Generated if "IsValid = True". It should be the same as the one already exists in the E-Invoice XML file.
  - ErrorMessage: The error message if "IsValid = false".
- Integration with other systems
  - Declare object from Class "HashGenerationService"
    - Declare object from class “SigningService”
      - Call function "generateInvoiceHash"
        - Parameter 1 (`xmlFilePath`) : E-Invoice XML file full path

3- Generating QR

- Input: The E-Invoice XML file full path.
- Output: an object of type "Result" that contains
  - IsValid : The final Status of the process.
  - ResultValue: QR code Generated if "IsValid = True". It should be the same as the one already exists in the E-Invoice XML file.
  - ErrorMessage: The error message if "IsValid = false".
- Integration with other systems
  - Declare object from Class "QRCodeGeneratorService "
  - Call function " generateQrCode"
    - Parameter1 (`values`): info needed to create qr from E-Invoice XML (certificate, sellerName, vatRegistrationNumber, timestamp, invoiceTotal, vatTotal, signature, invoiceType) with data type hashMap.
    - Parameter2 (`hashedXml`): hash extracted from E-Invoice XML.
    - Parameter3 (certificate): Certificate String

4- Validating QR (Containing Generating QR and validate it)

- Input: The E-Invoice XML file full path.
- Output: an object of type "Result" that contains
  - IsValid : The final Status of the process it should be true if the generated QR Code is the same as the one already exists in E-Invoice XML.
  - ResultValue: Empty
  - ErrorMessage: The error message if "IsValid = false".
- Integration with other systems
  - Declare object from Class " QrCodeValidator"
  - Call function " validate"
    - Parameter1 (`xmlFilePath`): E-Invoice XML file full path

5- Validating E-Invoice

- There are two types of E-Invoices "Simplified" & "Standard"
- Validation of "Simplified" E-Invoice contains the next steps
  - Validate XSD
  - Validate EN Schema Tron
  - Validate KSA Schema Tron
  - Validate Signature
  - Validate QR
  - Validate PIH
- Validation of "Standard" E-Invoice contains the next steps
  - Validate XSD
  - Validate EN Schema Tron
  - Validate KSA Schema Tron
  - Validate PIH
- Input:
  - The E-Invoice XML file full path
  - The content of certificate file.
  - The Current PIH as string .
- Output: an object of type "Result" that contains
  - IsValid : The final Status of the process it should be true if all steps of E-Invoice validation "IsValid"
  - ResultValue: Empty
  - ErrorMessage: Empty.
  - List of Steps each step is an object of type "Result" that contains if this step IsValid or not and the error message if it is not valid.
- Integration with other systems
  - Declare object from Class " XsdValidator"
  - Call function " validate"
    - Parameter 1 (`xmlFilePath`): E-Invoice XML file full path
  - Declare object from Class “SchematronValidator”
  - Call function " validate"
    - Parameter 1 (`xmlFilePath`): E-Invoice XML file full path

6- Signing E-Invoice

- Signing E-Invoice contains the next steps
  - Generating Hashing
  - Generating Signature
  - Populating Data
  - Generating and populating QR
- Input:
  - The E-Invoice XML file full path.
  - The content of certificate file as string
  - The content of private key as string (generated by OpenSSL)
- Output: an object of type "Result" that contains

- IsValid : The final Status of the process it should be true if all steps of E-Invoice Signing "IsValid"
- ResultValue: XML content after signing if it is valid
- ErrorMessage: Empty.
- List of Steps each step is an object of type "Result" that contains if this step IsValid or not and the error message if it is not valid.

- Integration with other systems
  - Declare object from Class " SigningService"
  - Call function " SignDocument"
    - Parameter 1 (`xmlFilePath`): E-Invoice XML file full path
    - Parameter 2 (`certificateContent`): Certificate file content as string
    - Parameter 3 (`privateKeyContent`): Private key file content as string

7- Generating E-Invoice Request

- Generating E-Invoice request contains the next steps
  - Retrieve uuid from E-Invoice
  - Retrieve invoice hash from E-Invoice
- Input:
  - The E-Invoice XML file full path.
  - The content of certificate file as string
  - The content of private key as string (generated by OpenSSL)
- Output: an object of type " Properties" that extends from Hash Table
- Integration with other systems
  - Declare object from Class " InvoiceRequestGenerationService "
  - Call function " getInvoiceData "
    - Parameter 1 (`invoice`): E-Invoice XML file as string

Running the SDK inside your application

• Prerequisites:

Make sure to Install the latest SDK version (see section **SDK Installation Steps).**

• Running SDK inside a java application:

To run any function inside your java application, use the following code snippet:

```
Runtime rt = Runtime.getRuntime();
Process pr = rt.exec("fatoora  <command> <args> ");
```

where <command> is any sdk command. For example, **-generateCsr** or **-generateHash** as well as any arguments associated as mentioned previously in the documentation.