**TASK/Requirement**

**encode Tag-Length-Value (TLV) string as per KSA into base64 and generate QR code**

=====================================

Pre-requisites

=====================================

\*) R&D for required format as per govt rule.

\*) R&D for tlv and base64 encoding.

\*) Process design for development.

\*) workflow design for process.

====================================

Development

\*) first, I was trying it with plsql but after conversion of tlv formatted base64 string occurred an issue in generating qr image with the encoded string generated using plsql. Then decided to go with java for this.

\*) written a java program for tlv formatted base64 string which is required for e-invoicing.

\*) written a java program to create qr image from output string of first program.

\*) downloaded and configured build path for java lib required for qr and tlv&base64 conversion.

\*) combined both java programs in single code package.

\*) Testing the whole logic as per ksa document standards for e-invoice.

\*) creating a jar file with variables for bundling all subprograms and libraries and required fields as input.

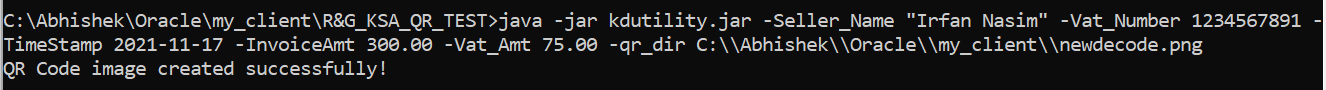
===================================

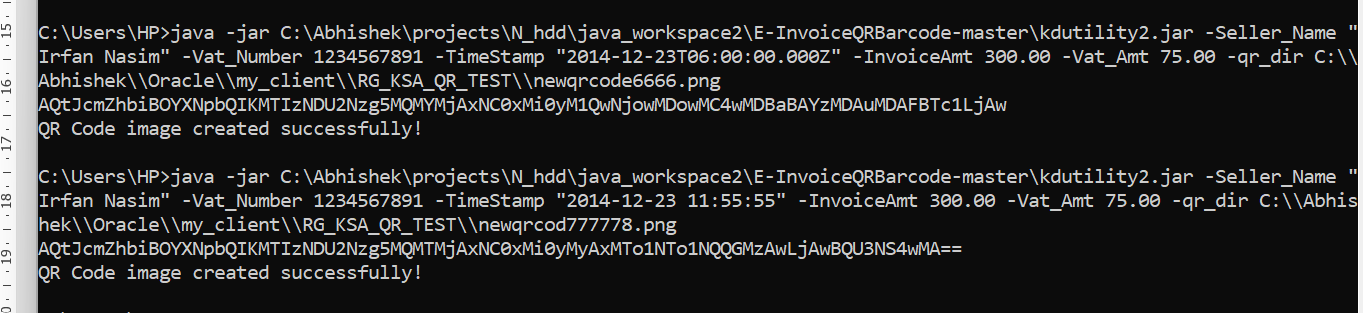
RESULT OUTPUT

===================================

you can use this jar file to get required ksa formatted qr image using below command by passing the parameter values:

java -jar kdutility.jar -Seller\_Name "Irfan Nasim" -Vat\_Number 1234567891 -TimeStamp 2021-11-17 -InvoiceAmt 300.00 -Vat\_Amt 75.00 -qr\_dir C:\\Abhishek\\Oracle\\newqrcode.png





java -jar C:\Abhishek\projects\N\_hdd\java\_workspace2\E-InvoiceQRBarcode-master\kdutility2.jar -Seller\_Name "Irfan Nasim" -Vat\_Number 1234567891 -TimeStamp "2014-12-23T06:00:00.000Z" -InvoiceAmt 300.00 -Vat\_Amt 75.00 -qr\_dir C:\\Abhishek\\Oracle\\my\_client\\RG\_KSA\_QR\_TEST\\newqrcode6666.png

java -jar C:\Abhishek\projects\N\_hdd\java\_workspace2\E-InvoiceQRBarcode-master\kdutility2.jar -Seller\_Name "Irfan Nasim" -Vat\_Number 1234567891 -TimeStamp "2014-12-23 11:55:55" -InvoiceAmt 300.00 -Vat\_Amt 75.00 -qr\_dir C:\\Abhishek\\Oracle\\my\_client\\RG\_KSA\_QR\_TEST\\newqrcod777778.png

QR - newqrcode6666



QR- 777778 - 

====================================

Benefits

====================================

\*) now the process done by java program so it took some extra effort but it is beneficial in many aspects of development.

\*) As we are using java code so the utility is platform independent i.e., windows, Unix, Solaris etc...

\*) And we can use the qr image to print on oracle reports, crystal reports etc...

\*) we can also create a simple gui tool in forms ERP application to generate qr code on click event.

\*) We can also replace third party utility or tools which we are currently using in applications for QR codes with our own created utility by separating qr related code from our utility.

\*) if those third-party tools stop working and as we don't have their source code so it will impact the work. and in respect to this it will also reduce the dependency of any other third-party tools.

\*) I Will give you the source code with comments on each step so that you can have full control of the utility in future.

===================================

EXTRA Efforts.

===================================

involved a java developer for a day

Attachment



Source – java

**import** com.github.busaeed.einvoice.qrbarcode.QRBarcodeEncoder;

**import** com.github.busaeed.einvoice.qrbarcode.tag.InvoiceDate;

**import** com.github.busaeed.einvoice.qrbarcode.tag.InvoiceTaxAmount;

**import** com.github.busaeed.einvoice.qrbarcode.tag.InvoiceTotalAmount;

**import** com.github.busaeed.einvoice.qrbarcode.tag.Seller;

**import** com.github.busaeed.einvoice.qrbarcode.tag.TaxNumber;

**import** com.google.zxing.BarcodeFormat;

**import** com.google.zxing.EncodeHintType;

**import** com.google.zxing.MultiFormatWriter;

**import** com.google.zxing.client.j2se.MatrixToImageWriter;

**import** com.google.zxing.common.BitMatrix;

**import** com.google.zxing.qrcode.decoder.ErrorCorrectionLevel;

**import** java.io.File;

**import** java.nio.charset.StandardCharsets;

**import** java.util.HashMap;

**import** java.util.Map;

**public** **class** Test {

**static** String *name*;

**static** String *text*;

**static** String *text1*;

**static** String *text2*;

**static** String *text3*;

**static** String *out*;

**public** **static** **void** main(String[] args) {

**if** (args[0].equals("-Seller\_Name")) {

*name* = args[1];

**if** (args[2].equals("-Vat\_Number"))

*text* = args[3];

**if** (args[4].equals("-TimeStamp"))

*text1* = args[5];

**if** (args[6].equals("-InvoiceAmt"))

*text2* = args[7];

**if** (args[8].equals("-Vat\_Amt"))

*text3* = args[9];

**if** (args[10].equals("-qr\_dir"))

*out* = args[11];

}

String qrBarcodeHash = QRBarcodeEncoder.*encode*(

**new** Seller(*name*),

**new** TaxNumber(*text*),

**new** InvoiceDate(*text1*),

**new** InvoiceTotalAmount(*text2*),

**new** InvoiceTaxAmount(*text3*)

);

System.*out*.println(qrBarcodeHash);

**try** {

String qrCodeData = qrBarcodeHash;

String filePath = *out*;

String charset = "UTF-8"; // or "ISO-8859-1"

Map < EncodeHintType, ErrorCorrectionLevel > hintMap = **new** HashMap < EncodeHintType, ErrorCorrectionLevel > ();

hintMap.put(EncodeHintType.*ERROR\_CORRECTION*, ErrorCorrectionLevel.*L*);

BitMatrix matrix = **new** MultiFormatWriter().encode(

**new** String(qrCodeData.getBytes(charset), charset),

BarcodeFormat.*QR\_CODE*, 200, 200, hintMap);

MatrixToImageWriter.*writeToFile*(matrix, filePath.substring(filePath

.lastIndexOf('.') + 1), **new** File(filePath));

System.*out*.println("QR Code image created successfully!");

} **catch** (Exception e) {

System.*err*.println(e);

}

}

}