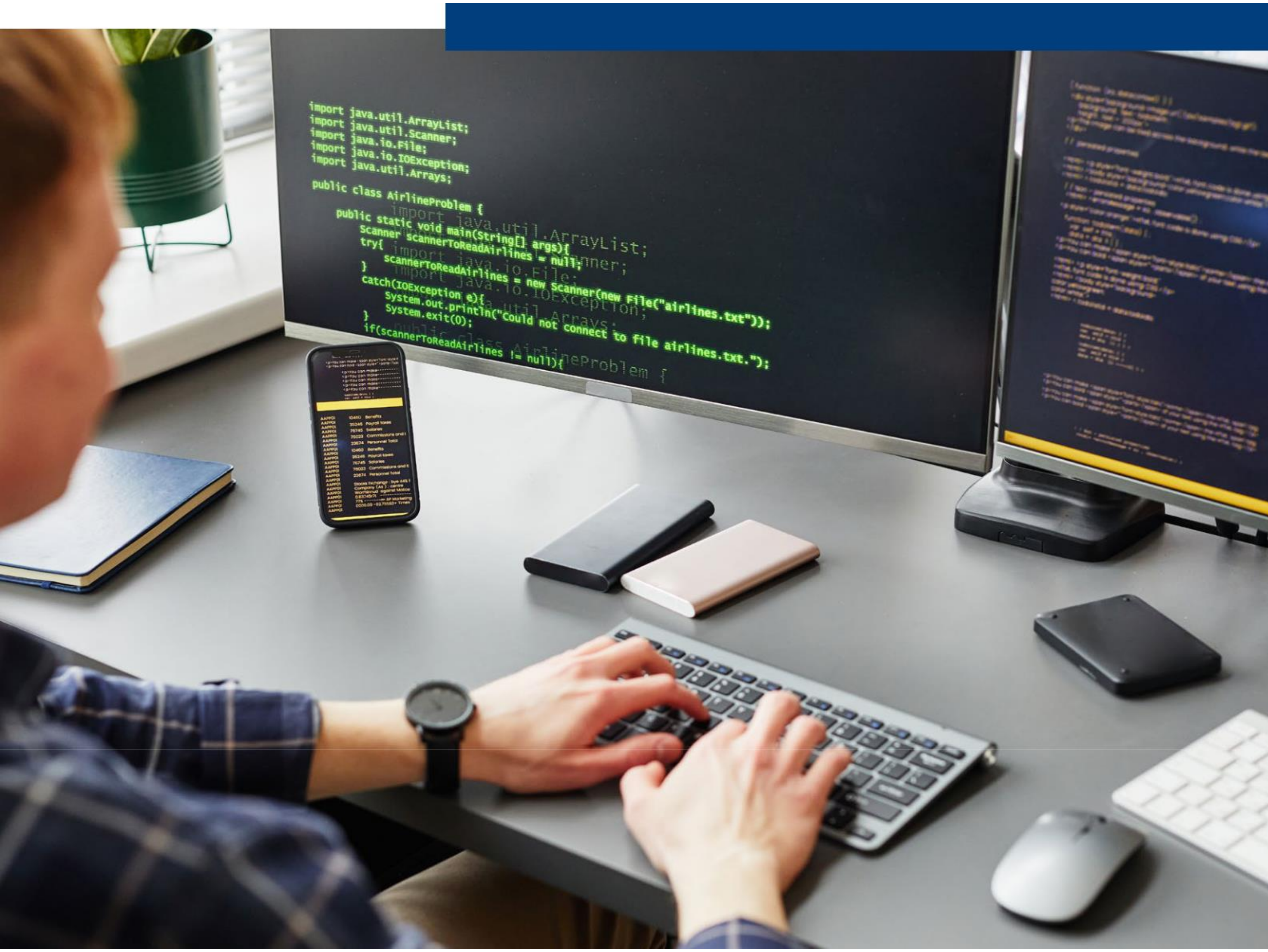
MAURITIUS
REVENUE
AUTHORITY

Technical guide to prepare and send EBS Transactions

For software Developers



```
import java.util.ArrayList;
import java.util.Scanner;
import java.io.File;
import java.io.IOException;
import java.util.Arrays;

public class AirlineProblem {
    public static void main(String[] args){
        Scanner scannerToReadAirlines = null;
        try{
            scannerToReadAirlines = new Scanner(new File("airlines.txt"));
        }
        catch(IOException e){
            System.out.println("Could not connect to file airlines.txt.");
            System.exit(0);
        }
        if(scannerToReadAirlines != null){
```

Disclaimer

This document is provided to the public for information purposes only. Information in this document is indicative and is subject to change without notice. Unless otherwise noted, the information used in examples herein is fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. It is the responsibility of the user to comply with all applicable copyright laws.

Table of Contents

1. Version History

Version	Changes	Date
v1.0	Original document	21st June 2023
v1.1	Changes in section • 7.1.2 – additional information has been provided for areaCode • 8.1.12 – additional information has been provided on how to use key for encrypting an invoice • 11 – Code snippet in Java and C# have been added	18th September 2023
v1.2	Changes in section • 8.1.8 – qrCode details has been updated • 8.1.12 – Flowchart has been added for invoice transmission • 8.1.13 – Steps to use the MRA generated QR Code and print same in receipts	16th October 2023
v1.3	Changes in section • 8.1.7.1 Hashing algorithm • 8.1.7.2 Steps to generate previous invoice note	19th October 2023
v1.3.1	Changes in section • 11 – Code snippet in PHP has added	01st December 2023
v1.3.2	Changes in section • 11 – Code snippet in PHP update	06th February 2024
V1.3.3	Changes in section • 8.1.14 – Not yet fiscalised	14th March 2024
V1.3.4	Changes in section 11	18th March 2024

2. Glossary of Terms

TERM	DESCRIPTION
API	Application Programming Interface
ECR	Electronic Cash Register
EBS	Electronic Billing System
ERP	Enterprise Resource Planning
ICT	Information and Communication Technology
JSON	JavaScript Object Notation
MRA	Mauritius Revenue Authority
OTP	One Time Password
POS	Point of Sale
QR Code	Quick Response Code
URL	Uniform Resource Locator
IRN	Invoice Registration Number

3. Introduction

The Mauritius Revenue Authority (MRA) is introducing an e-Invoicing system in Mauritius. With the advent of this e-Invoicing system at the national level, sellers will be required to fiscalise their invoices or receipts in real time with the MRA before issuing same to their customers (that is, the so-called buyers).

The following diagram gives a pictorial representation of the e-Invoicing system as implemented by MRA.

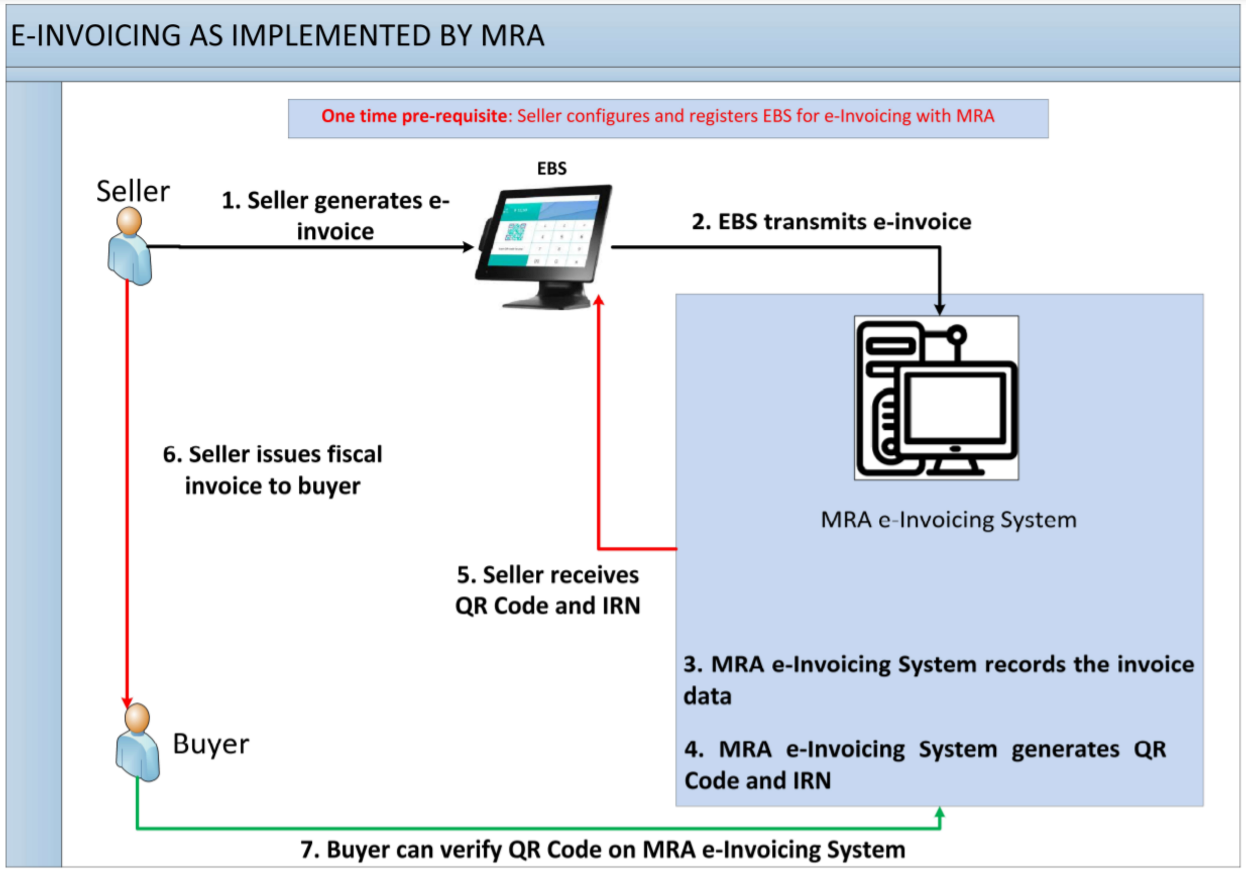

E-INVOICING AS IMPLEMENTED BY MRA
One time pre-requisite: Seller configures and registers EBS for e-Invoicing with MRA
EBS
Seller
1. Seller generates e-invoice
2. EBS transmits e-invoice
MRA e-Invoicing System
3. MRA e-Invoicing System records the invoice data
4. MRA e-Invoicing System generates QR Code and IRN
5. Seller receives QR Code and IRN
6. Seller issues fiscal invoice to buyer
Buyer
7. Buyer can verify QR Code on MRA e-Invoicing System

Figure 1: Pictorial representation of MRA e-Invoicing System

4. Purpose of this Technical Guide

This guide serves as a tool for Developers and EBS Solution Providers to make their Electronic Billing System (EBS) compliant with the MRA e-Invoicing System. EBS Software Developers and EBS Solution Providers will have to do needful to enable an EBS to generate invoice details in the format requested by MRA and submit same to MRA.

This document

- Provides guidelines on how to prepare, encrypt and transmit EBS transactions to the MRA e-Invoicing System
- Assumes that the reader has read the guidelines and has prior knowledge of JavaScript Object Notation (JSON) and JSON schema technology.

5. Prerequisites

5.1.Registration on the MRA e-Invoicing Developer Portal

The user has to

- sign up on the MRA e-Invoicing Developer Portal by creating a username and password,
- register a user profile (which will trigger a registration process at MRA),
- register EBS in order to get a unique ID known as the EBS MRA ID for each EBS registered. The EBS MRA ID will be used while transmitting invoices.

The link to access the MRA e-Invoicing Developer Portal is https://vfiscportal.mra.mu/einvoice-portal/home

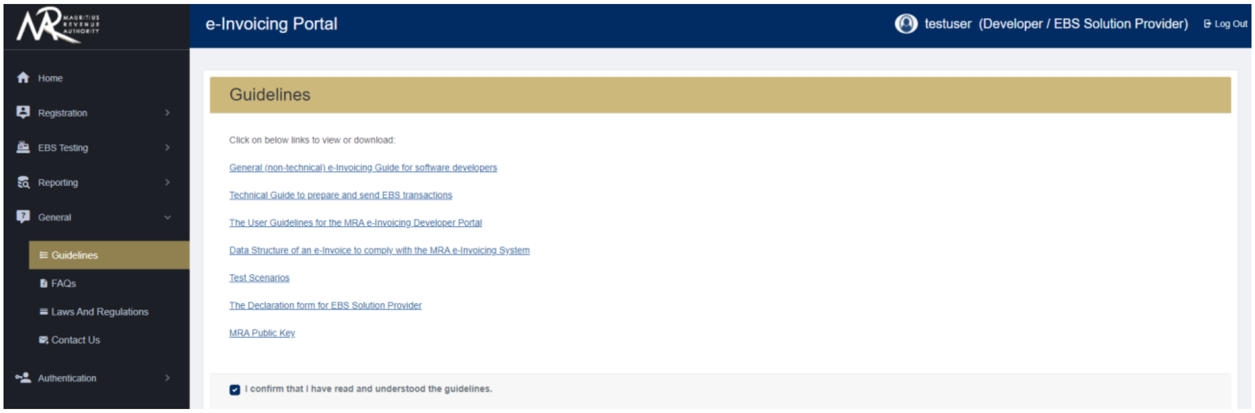
For more information on registration, refer to the MRA e-Invoicing Portal User manual on https://vfiscportal.mra.mu/einvoice-portal/guides

5.2.MRA Public Key Pre-Requisites

The user has to download and save the MRA Public Key (.crt file) / on his local computer. The MRA Public Key will be used for encrypting an AES Symmetric key prior to calling the Authentication API.

The MRA Public Key is a key which the user has to mandatorily download and save on his local computer. This key will be used for encrypting the authentication payload prior to calling the e-Invoicing Authentication API.

Login on the MRA e-Invoicing Developer Portal, go to the Guidelines Section and click the link MRA Public Key as shown below:



Guidelines

Click on below links to view or download:

General (non-technical) e-Invoicing Guide for software developers

Technical Guide to prepare and send EBS transactions

The User Guidelines for the MRA e-Invoicing Developer Portal

Data Structure of an e-Invoice to comply with the MRA e-Invoicing System

Test Scenarios

The Declaration form for EBS Solution Provider

MRA Public Key

☑ I confirm that I have read and understood the guidelines.

Figure 2: MRA Public Key for download

6. API Overview

This section describes the standards and the formats which will be used to define the APIs exposed by MRA. The MRA e-Invoicing APIs are implemented as RESTful Web Services.

The following HTTP methods are used across the APIs

HTTP Method	
POST	To authenticate an EBS and to submit EBS transaction data to MRA e-Invoicing System
GET	To fetch QR Code and IRN from MRA e-Invoicing System

6.1. Authentication of an EBS

The Authentication API is used to authenticate an EBS. Prior to calling the Transmission API or any other API

- It is necessary to authenticate an EBS with the authentication server
- The user has to call the Authentication API and request an authentication token which will be used when calling the Transmission API

6.2. Transmission of invoices

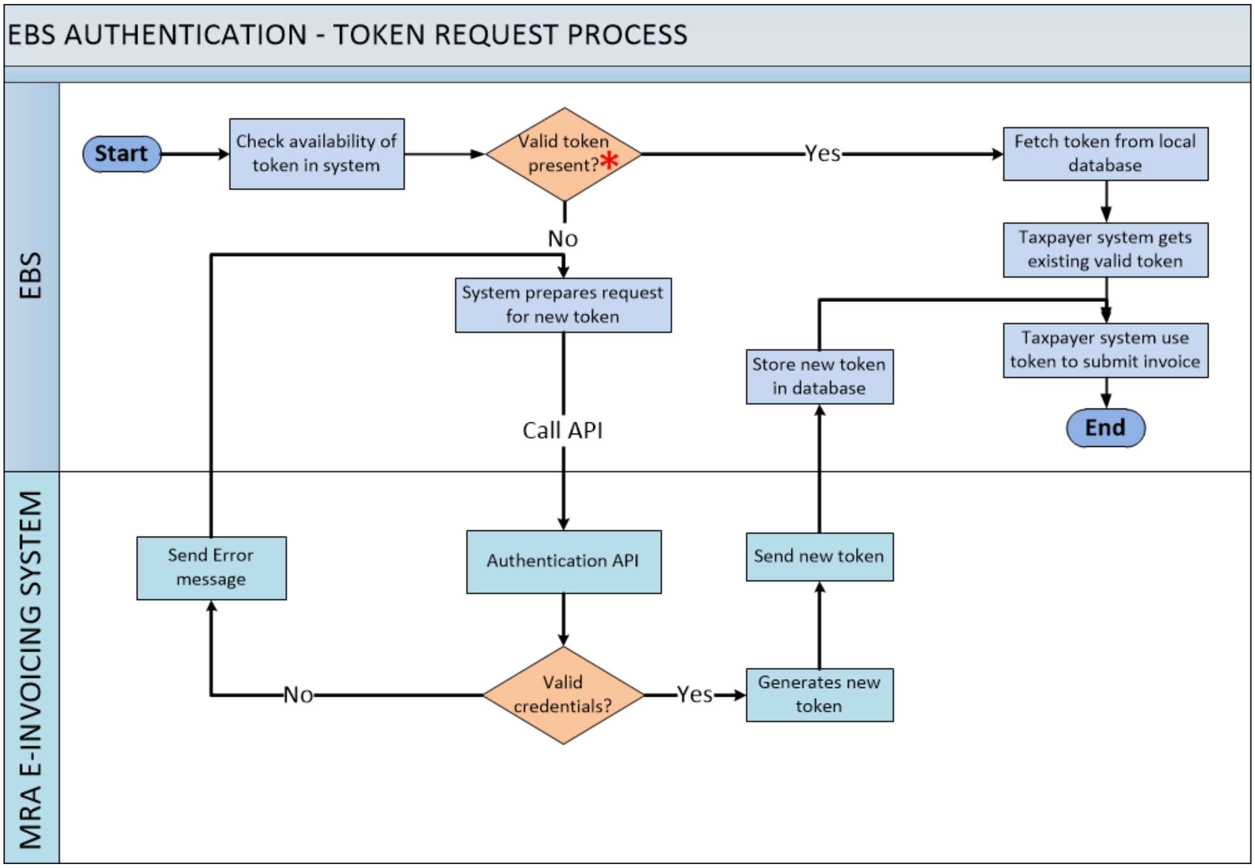
The Transmission API is used to transmit EBS transaction data on the MRA e-Invoicing System. After successful transmission of the transaction, the Transmission API will reply with a QR Code and an IRN.

7. Authentication API

Prior to calling the MRA e-Invoicing System, a registered EBS will have to call the Authentication API for authentication and request an authentication token. The authentication token will then be used when transmitting EBS transaction data to the Transmission API. The token will be valid for the day in the Live MRA e-Invoicing System.

On expiry of the authentication token, the same API needs to be invoked in order to get a new token.

The following diagram depicts the token request process on EBS system and MRA E-Invoicing System.


EBS AUTHENTICATION - TOKEN REQUEST PROCESS
EBS
MRA E-INVOICING SYSTEM
Start
Check availability of token in system
Valid token present?*
Yes
Fetch token from local database
Taxpayer system gets existing valid token
Taxpayer system use token to submit invoice
End
No
System prepares request for new token
Call API
Store new token in database
Send Error message
Authentication API
Send new token
No
Valid credentials?
Yes
Generates new token

*** An existing token is considered valid when it has more than 10 minutes before expiry**

Figure 3: EBS Authentication – Token request process

7.1.Authentication Token Request Process

When calling the Authentication API, the request header should contain the EBS MRA ID and the username of the user who registered the EBS on the MRA e-Invoicing Developer Portal.

The request payload is a JSON containing the credentials which is encrypted using the MRA e-Invoicing System Public Key (.crt file)

The below diagram describes the authentication token request flow of an EBS

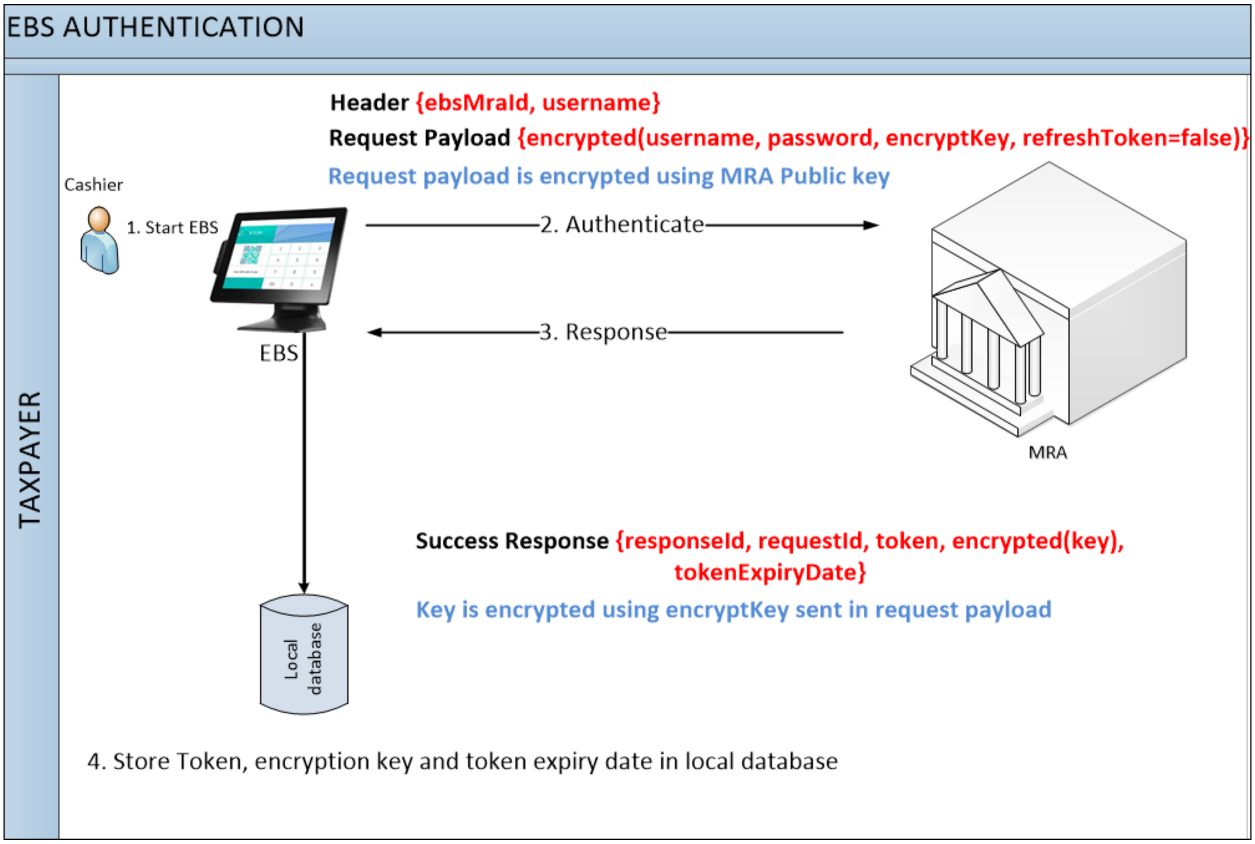

EBS AUTHENTICATION
TAXPAYER
Header {ebsMraId, username}
Request Payload {encrypted(username, password, encryptKey, refreshToken=false)}
Request payload is encrypted using MRA Public key
Cashier
1. Start EBS
2. Authenticate
3. Response
EBS
MRA
Success Response {responseId, requestId, token, encrypted(key), tokenExpiryDate}
Key is encrypted using encryptKey sent in request payload
Local database
4. Store Token, encryption key and token expiry date in local database

Figure 4: EBS Authentication – Token request flow

The format and details of the Authentication API request and response are depicted in following tables.

7.1.1. Authentication request

Format and Details of the request	
URL	https://vfisc.mra.mu/einvoice-token-service/token-api/generate-token
Content-Type	Application/JSON
Method	POST

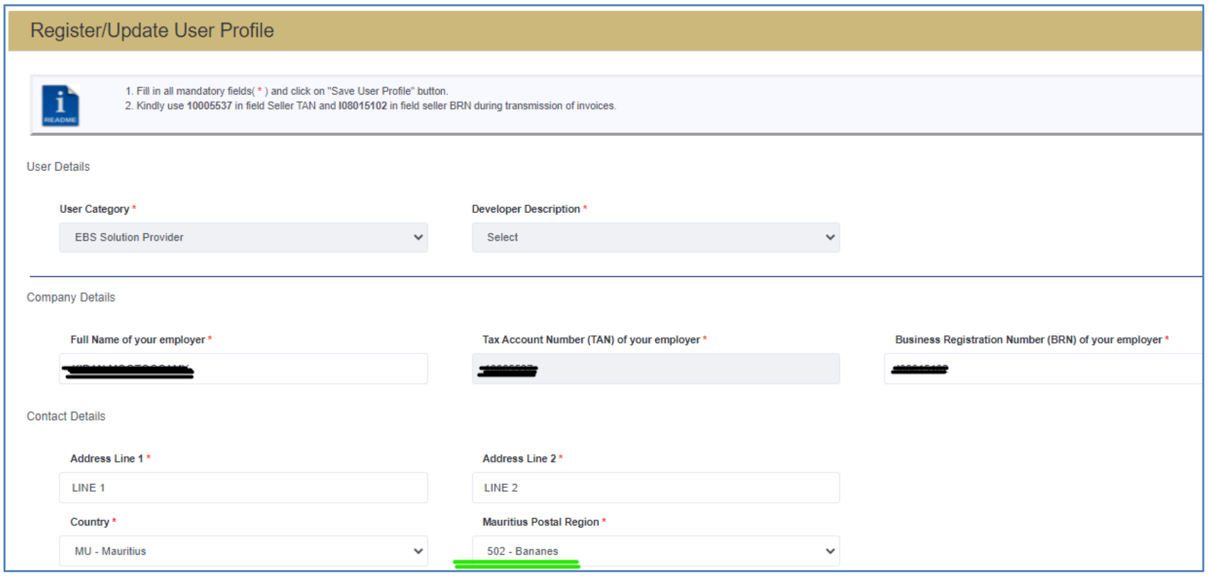
7.1.2. Request Header

The attributes of the request header are

Attributes	Description
username	The username provided at registration time on the MRA e-Invoicing Developer Portal.
ebsMraId	The unique ID provided by MRA for an EBS at registration time.
areaCode	The area code specified during the registration of the user profile (Refer to example in screen shot below, line highlighted in green. The area code to be used is 502.)

The area code can be verified on

1. **Register/Update User Profile** screen or
2. **Register/Update Electronic Billing System (EBS)** screen.



Register/Update User Profile

1. Fill in all mandatory fields(*) and click on "Save User Profile" button.
2. Kindly use **10005537** in field Seller TAN and **I08015102** in field seller BRN during transmission of invoices.

User Details

User Category *: EBS Solution Provider

Developer Description *: Select

Company Details

Full Name of your employer *: [redacted]

Tax Account Number (TAN) of your employer *: [redacted]

Business Registration Number (BRN) of your employer *: [redacted]

Contact Details

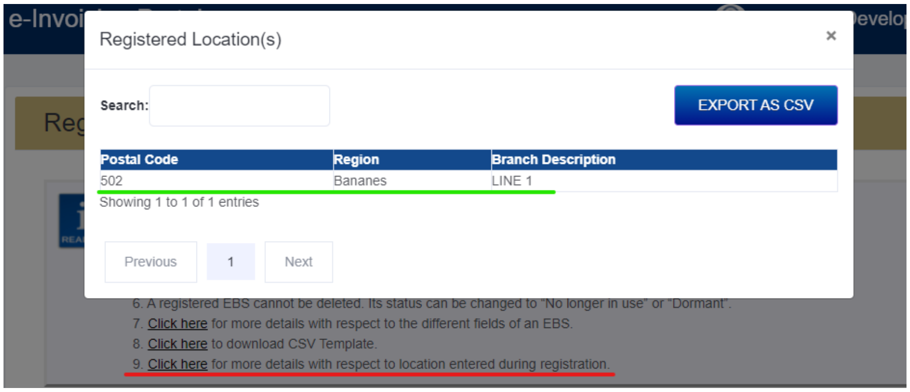
Address Line 1 *: LINE 1

Address Line 2 *: LINE 2

Country *: MU - Mauritius

Mauritius Postal Region *: 502 - Bananes

Figure 5: Registration of User Profile



Registered Location(s)

Search:

Postal Code	Region	Branch Description
502	Bananes	LINE 1

Showing 1 to 1 of 1 entries

Previous 1 Next

6. A registered EBS cannot be deleted. Its status can be changed to "No longer in use" or "Dormant".
7. Click here for more details with respect to the different fields of an EBS.
8. Click here to download CSV Template.
9. Click here for more details with respect to location entered during registration.

Figure 6: Register/Update Electronic Billing System

7.1.3. Request Payload

The attributes of the request payload are

Attributes	Description
requestId	Unique ID for each request. The requestId should be generated by the consuming client (i.e. the EBS).
payload	Payload containing the credentials which is encrypted using the MRA Public Key and then encoded using Base 64. The asymmetric algorithm (RSA/ECB/PKCS/Padding) is used to encrypt the request payloads.

7.1.4. JSON attributes corresponding to the Payload

Attributes	Description
username	The username provided at registration time on the MRA e-Invoicing Developer Portal.
password	The password set for above username at registration time.
encryptKey	Base 64 encoded string of a random 32 byte AES key (symmetric key).
refreshToken	Set to true in case a new token is required within the specified time of expiry (10 minutes before expiry time).

7.1.5. Sample Authentication request

```
{  "requestId": "20230324213055",
    "payload": "TCcvYcGczIf5pzk6RiqHO00BtjkD2pw4HC0wwPq29Wvw/T7P2cMd55RijSGQaeBIQvFufuWOo8GTBC
eQckWICKifL4/45NvuU75IqsuNHQ41iegrjp/lv+P9RWvA9Cha45GUFBnZI/lN+AUYfmdwR/SMwqXb0m7Ac/xZatBcz0pv
9C0t3IjcLLDry6wht6iF2whEtFBWltXmhH00a9BBquKqHR8H1SLX62PeCFGKsqJLHefib3ARvb8gvxUpPrIsf7gBtZeQEs
TZV6apnnkhvPJYp3gBEF14/bMpYZqtdirgFofXVsKPCHtSX2dveIqqbCD6IgsFBjgn0AfLbhoaTQ=="
}
```

JSON corresponding to the payload element of the above authentication request

```
{
    "username": "developer@xyz.mu",
    "password": "Pa$$12345",
    "encryptKey": "46REr654ds$372DSgs$&DLW58",
    "refreshToken": "false"
}
```

7.1.6. Response Payload (Success)

The attributes of a response after a successful authentication of an EBS are

Attributes	Description
responseId	Unique response ID generated by MRA for each request.
requestId	The request ID sent in the authentication request payload.
status	Status of the authentication request [SUCCESS|ERROR].
token	Authentication token to be used in invoices in order to access the transmission API.
key	Base 64 encoded string of encrypted key. Key has been generated using AES 256(AES/ECB/PKCS5Padding or AES/ECB/PKCS7Padding) algorithm and encrypted with the encryptKey present in the request payload.
expiryDate	The date and time token will expire. The format is YYYYMMDD hh:mm:ss.

7.1.7. Response Payload (Error)

The attributes of a response after a failed authentication of an EBS are

Attributes	Description
responseId	Unique response ID generated by MRA for each request.
requestId	The request ID sent in the authentication request payload.
status	Status of the authentication request [SUCCESS|ERROR].
errors	Error messages returned in case of error.

7.1.8. Sample Authentication Response (Success)

```
{
  "responseId": "TK16799824364741414703061",
  "requestId": "REQ0021",
  "status": "SUCCESS",
  "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJzaHJhdmFuaS5qZWV3b25AbXJhLm11IiwiZWJzTXJhSWQiOiIxNjc3NjAzMjY2NjA3N
09VYVZnQnYiLCJleHAiOjE2Nzk0MjA1ODQsImlhdCI6MTY3OTMzNDE4NH0.2fgbFZjA8Y7orp7flQPVNHvhJdu0RNIc7qOyjLIDqiilirB
MrrBmLY6BuxcJ4iHZiMyq0yNcEpX-cDUZEake5Q",
  "key": "TUnROQNSsuC8vdVhYoMRrJ0PW5rwVv7IGc9GZXA4osGr1jS9hwj1yEcOX61N6yO6",
  "expiryDate": "20230328 14:18:35 "
}
```

7.1.9. Sample Authentication Response (error)

```
{
 "status": "ERROR",
 "errors": [
     "Invalid Header Request"
       ]
}
```

7.1.10. Steps to produce Authentication JSON in format requested

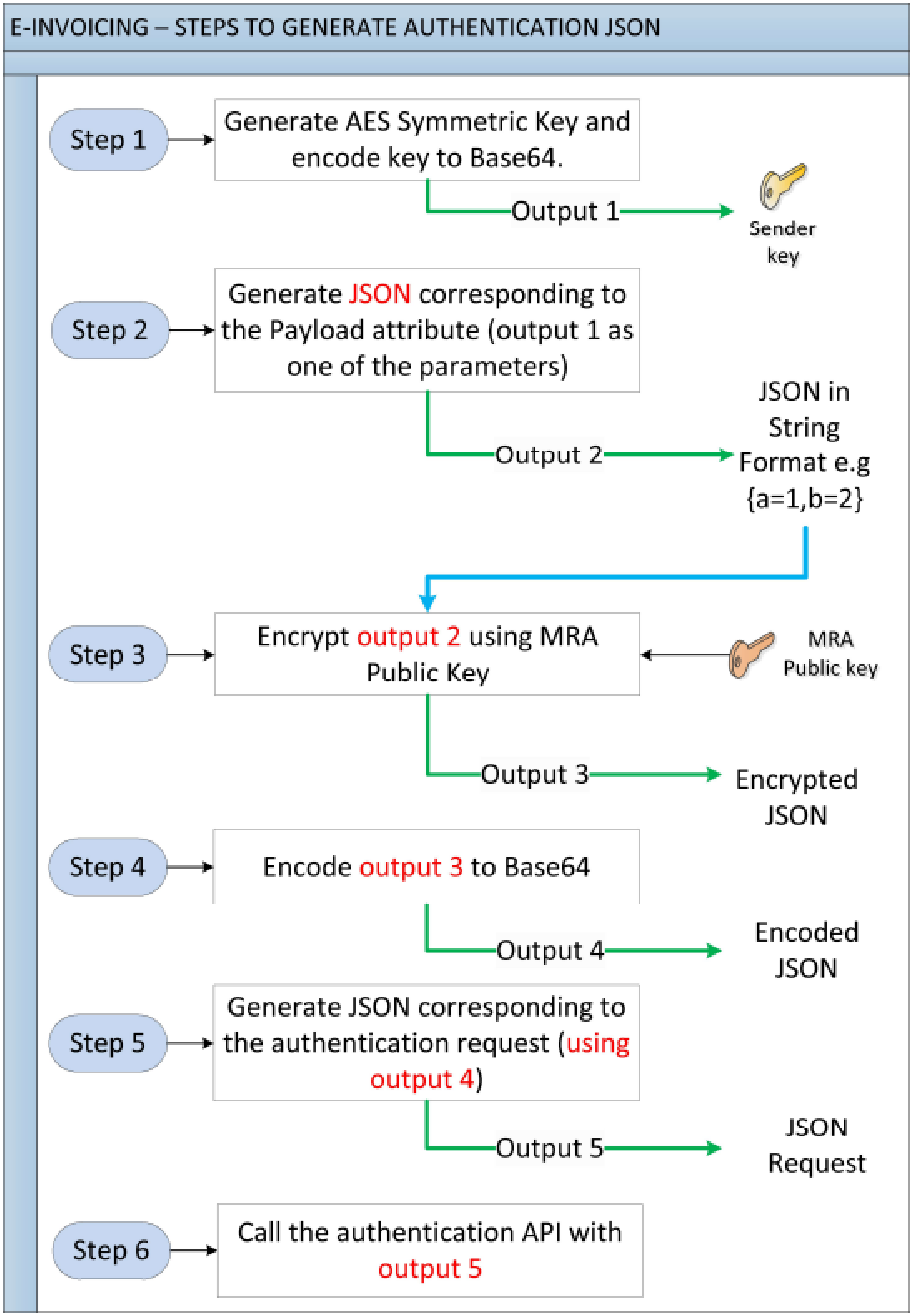

E-INVOICING – STEPS TO GENERATE AUTHENTICATION JSON
Step 1
Generate AES Symmetric Key and encode key to Base64.
Output 1
Sender key
Step 2
Generate JSON corresponding to the Payload attribute (output 1 as one of the parameters)
Output 2
JSON in String Format e.g {a=1,b=2}
Step 3
Encrypt output 2 using MRA Public Key
MRA Public key
Output 3
Encrypted JSON
Step 4
Encode output 3 to Base64
Output 4
Encoded JSON
Step 5
Generate JSON corresponding to the authentication request (using output 4)
Output 5
JSON Request
Step 6
Call the authentication API with output 5

Figure 7: Flow chart to generate authentication JSON

	Steps to produce the authentication JSON
1	Generate AES Symmetric Key (encryptKey) and encode key to Base 64.
2	Generate JSON corresponding to the "payload" attribute (username, password, encryptKey, refreshToken).
3	Encrypt JSON string from step 2 using MRA Public Key
4	Encode encrypted JSON from step 3 to Base 64
5	Generate JSON corresponding to the authentication request with a unique request ID
6	Call the authentication API with **username** and **EBS MRA ID** in the request header and JSON from 5 in the request body

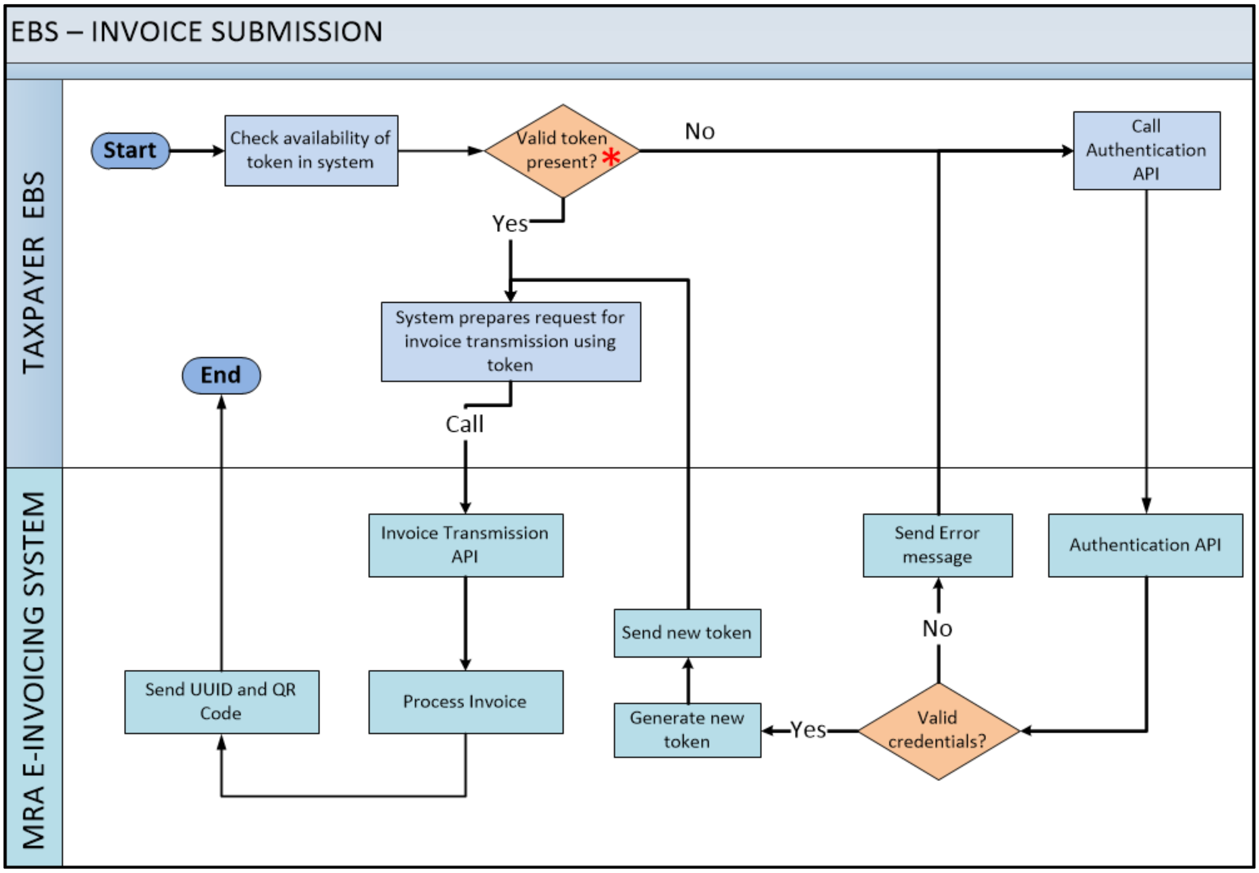
7.1.11. List of errors when calling the Authentication API

Reason	HTTP Status Code	Description
Authentication Error	400	Unauthorized request
Invalid Header Request	400	Incorrect username and/or ebsMraId in header
Decryption failed	400	Error raised during decryption
Invalid User	400	Username and/or password not match
Incorrect Payload	400	Token payload is incorrect
Incorrect Payload	400	Token payload is null
Attributes in payload is incorrect	400	Below reasons whereby a payload can be incorrect • username is mandatory • password is mandatory • encryptKey is mandatory • Refresh token should contain values: TRUE or FALSE • refreshToken is mandatory

8. Submission of an invoice

The Transmission API is used for submitting invoices on MRA e-Invoicing System.

The following diagram depicts the invoice submission process on EBS and MRA e-Invoicing System.



*** An existing token is considered valid when it has more than 10 minutes before expiry**

Figure 8: Invoice submission process

8.1.Invoice Submission Process

When calling the Transmission API, the request header should contain the **username** of the user who registered the EBS on the MRAID Portal, the **EBS MRA ID** of the registered EBS, the area code specified during registration and the valid **token** received after a successful authentication of the same EBS.

The below diagram describes the invoice transmission flow from an EBS

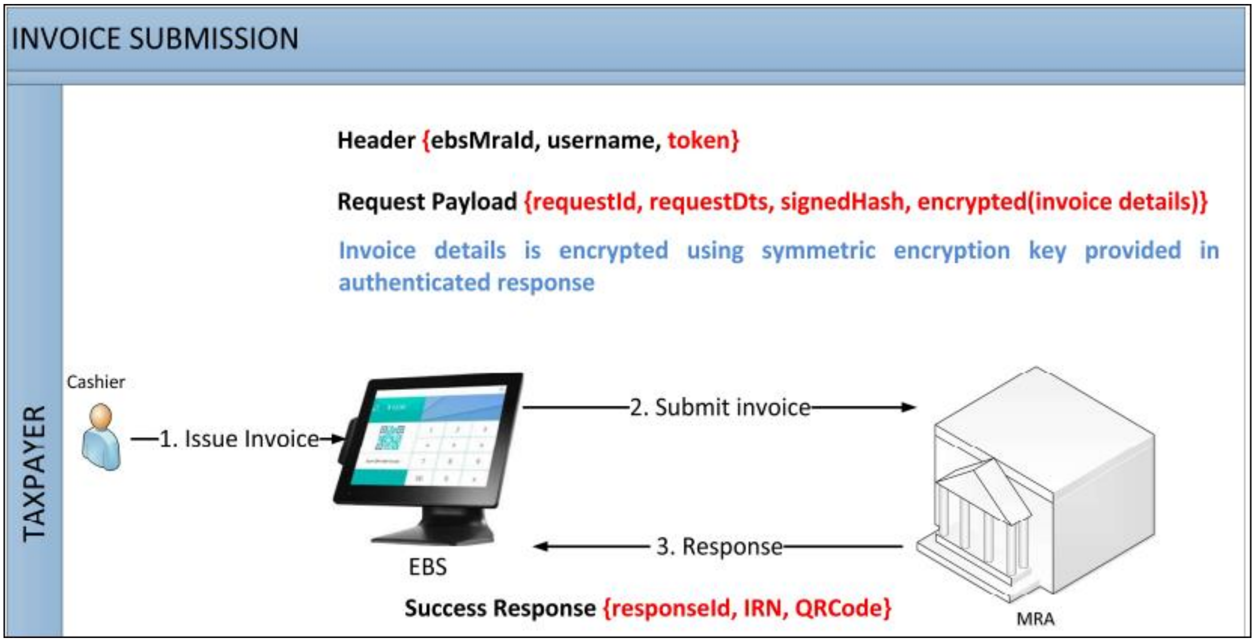

INVOICE SUBMISSION
TAXPAYER
Header {ebsMraId, username, token}
Request Payload {requestId, requestDts, signedHash, encrypted(invoice details)}
Invoice details is encrypted using symmetric encryption key provided in authenticated response
Cashier
1. Issue Invoice
2. Submit invoice
3. Response
EBS
MRA
Success Response {responseId, IRN, QRCode}

Figure 9: Invoice submission process

The format and details of the Transmission API request and response are depicted in following tables.

8.1.1. Transmission request

Format and Details of the request	
URL	https://vfisc.mra.mu/realtime/invoice/transmit
Content-Type	Application/JSON
Method	POST

8.1.2. Request Header

The attributes of the request header are

Attributes	Description
username	The username used at registration time on the MRA e-Invoicing Developer Portal
ebsMraId	The unique ID provided by MRA for an EBS at registration time
areaCode	The area code specified during the registration of the user profile
token	Authentication token returned in the authentication response

8.1.3. Request Payload

The attributes of the request payload are

Attributes	Description
requestId	Unique ID generated by user for each request
requestDateTime	Date time request was generated before calling MRA IFP
signedHash	Base 64 encoded string of the signed hash invoice (optional) (Refer to section 8.1.4)
encryptedInvoice	Base 64 encoded string of encrypted invoice details. The symmetric algorithm AES256 (AES/ECB/PKCS/Padding) is used to encrypt the invoice.

8.1.4. Signed Hash (optional)

A taxpayer may digitally sign an invoice prior sending the request payload. An attribute for the digital signature has been added in the request payload. Hence if a signed invoice is sent to the MRA e-Invoicing System, the same will be accepted. Note that this step is optional. As such the signed invoice attribute may be omitted from the request payload if not being used by the EBS.

In order to digitally sign an invoice, a taxpayer should have already uploaded a digital certificate associated with the EBS on the MRAID Portal.

The raw JSON corresponding to the attributed **encrypted Invoice** should be signed using SHA256withRSA and the Private Key of EBS of the taxpayer.

8.1.5. JSON attributes corresponding to attribute encryptedInvoice

For JSON attributes, refer to the document “**Data Structure of an e-Invoice to comply with the MRA e-Invoicing System (JSON format)”** which can be downloaded from guidelines section on the Portal.

8.1.6. Sample Invoice Transmission Request

```
{
        "requestId": "20230214_2",
        "requestDateTime": "20230221 11:27:42",
        "signedHash": "",
        "encryptedInvoice":
"4uGXAJcClVcOjy/66NdjPg5OkNXJ/hijN0mw+SeeoDTUqfSdqiI7nOXWAlAwsqOhpSWNy5GqBwGE+yJ/8OxYGeS5/5hza
lpbVLYUmqeIDnboxD0xUx8Bo26f+9/bcODpEfPyxgvkx4gIaybNnjiMajD7wvDEHdg13klY+vn7UTbTsnEe29kCCrUaLVr
J3duJJ9Nb5XPh31aLkAjdc9Cl3vvqPm7DVifUJQa8MCgtSEXUcWP/ZGw+69iUBr2rlU8AZ0sHu/LQ2Q7lPm8ZgzNiTd+LK
PQCViQxZkqCNceeILkreAIisyDoWgWUG0cdvu/kSu4GZstuLeQCNIXbjU2SXCQOSQU0gyEusXkYA3FMgRmlbHbd8vayRNn
fPeHEUys4PWlPGn+Z2rIf7rRys7Ukh4Har5pD/NsdgtRYhDOQiQdrOLUH7HAk1Sd/8yYpUdCiTSG+XJPuvYrUNkV4L8Wzl
IcMcdOYtmsq67V/XfKzgutYFGyaMgIyynBG/e9ag8JLUTnzykOQ72bijXBmlOrHxGSJ/tp3UlP24i7zqBuHr0hTVeReGbV
+eo2ugSOq3IPSpACF4Iwe99Z4t3ahLi6/K8ZVY9q2jE7mOLZTjgXeg8bVcVTI2Tg7wPyX/yd8kNL4jbpTkybHBk688iChn
7EZWFuyBB5lpoF4gpK7Pxbzzs/dI+VCXR7pGaYgSrwD08Q3Yzh3JX+iJZPhdBCUB9T7fvGHkQj51HDmNLoWsuC7BU1ewW+
+V73WQ0YrxHuoWEgZffmbHptgmQBmWwe/hrNkdKw3FJO5WhOkXadChzSfnt3HkivQVbSSCejO4Z+ymv/Jw2mpGr5WZOwSj
+7hc8Qs+31m9CgbogT8XOYQEd0zr1TaA7XZ3Kckt3seXpnOL6KRbiBPev2vBNbFyEjmt2MnWmLdIxBnaZiROWvttxluH2I
05Iqc4eOhRN5cQhtMmnkju01t/js4ZPjXsSoKmm79Vpy0g9SUoF+wkQRBmGBLq4fkPFRyMMRCeOhUB9F+wIRvodLTHu/P7
Yz0wWf9L3b0EK2Ympi2xnX+MGPbaToCphLNGiC//2Oyi7IUyS9k3Umu2MVPnBPanDDGfVZLBZLokGswoqk0JLWzn9EqpYP
A9Z1cFcRU3YmhHZ/WRE7RG06i2kxD6SSK47BYo1fIVCMTdzsl75Twx3HovU6E/E5GjCDj8RqPyAF3DNG7kRZ0z5dB74n/0
ZB1DYIb6kP2Z6GaQ3MfV63TK+NYuOLpokBuMvQZzPMu0wtwV40JX9xbBx5+7YFzfgaGE6rSXIGPty+6T15DpNkoOYT15BF
4M5cbNoXPBoKamQSMSaA+vukB6ezbwaTMqE0my15TuyP78RRgZQ=="
}
```

8.1.7. JSON corresponding to the "encryptedInvoice" is

```
    [
        {
                "invoiceCounter": "1",
                "transactionType": "B2C",
                "personType": "VATR",
                "invoiceTypeDesc": "STD",
                "currency": "MUR",
                "invoiceIdentifier": "abscs",
                "invoiceRefIdentifier": "",
                "previousNoteHash": "prevNote",
                "reasonStated": "rgeegr",
                "totalVatAmount": "60.0",
                "totalAmtWoVatCur": "310.0",
                "totalAmtWoVatMur": "310.0",
                "invoiceTotal": "370.0",
                "discountTotalAmount": "50.0",
                "totalAmtPaid": "320.0",
                "dateTimeInvoiceIssued": "20230531 10:40:30",
                "seller": {
                        "name": "Test User",
                        "tradeName": "TEST",
                        "tan": "1252XXXX",
                        "brn": "I080XXXXX",
                        "businessAddr": "Test address",
                        "businessPhoneNo": "",
                        "ebsCounterNo": "a1"
                },
                "buyer": {
                        "name": "Test user 2",
                        "tan": "12145785",
                        "brn": "CXXXXX23",
                        "businessAddr": "Test address 1",
                        "buyerType": "VATR",
                        "nic": ""
                },
                "itemList": [
                        {
                                "itemNo": "1",
                                "taxCode": "TC01",
                                "nature": "GOODS",
                                "productCodeMra": "pdtCode",
                                "productCodeOwn": "pdtOwn",
                                "itemDesc": "dILAIT CONDENc 23",
                                "quantity": "23214",
                                "unitPrice": "20",
                                "discount": "1.23",
                                "discountedValue": "10.1",
                                "amtWoVatCur": "60",
                                "amtWoVatMur": "50",
                                "vatAmt": "10",
                                "totalPrice": "60"
                        },
                        {
                                "itemNo": "2",
                                "taxCode": "TC01",
                                "nature": "GOODS",
                                "productCodeMra": "pdtCode",
                                "productCodeOwn": "pdtOwn",
                                "itemDesc": "2",
                                "quantity": "3",
                                "unitPrice": "20",
                                "discount": "0",
                                "discountedValue": "12.0",
                                "amtWoVatCur": "50",
                                "amtWoVatMur": "50",
                                "vatAmt": "10",
                                "totalPrice": "60"
                        },
                        {
                                "itemNo": "3",
                                "taxCode": "TC01",
                                "nature": "GOODS",
                                "productCodeMra": "pdtCode",
                                "productCodeOwn": "pdtOwn",
                                "itemDesc": "2",
                                "quantity": "3",
                                "unitPrice": "20",
                                "discount": "0",
                                "discountedValue": "12",
                                "amtWoVatCur": "50",
                                "amtWoVatMur": "50",
                                "vatAmt": "10",
                                "totalPrice": "60"
                        },
                        {
                                "itemNo": "4",
                                "taxCode": "TC01",
                                "nature": "GOODS",
                                "productCodeMra": "pdtCode",
                                "productCodeOwn": "pdtOwn",
                                "itemDesc": "2",
                                "quantity": "3",
                                "unitPrice": "20",
                                "discount": "0",
                                "discountedValue": "12.0",
                                "amtWoVatCur": "50",
                                "amtWoVatMur": "50",
                                "vatAmt": "0",
                                "totalPrice": "60"
                        },
                        {
                                "itemNo": "5",
                                "taxCode": "TC01",
                                "nature": "GOODS",
                                "productCodeMra": "pdtCode",
                                "productCodeOwn": "pdtOwn",
                                "itemDesc": "2",
                                "quantity": "3",
                                "unitPrice": "20",
                                "discount": "0",
                                "discountedValue": "12.6",
                                "amtWoVatCur": "50",
                                "amtWoVatMur": "50",
                                "vatAmt": "0",
                                "totalPrice": "60"
                        }
                ],
                "salesTransactions": "CASH"
        }
    ]
```

The above JSON corresponds to a list with one invoice.

8.1.7.1. Previous invoice/note hash - Hashing algorithm

Hashing is a process of converting data (usually of variable length) into a fixed-length string of characters, which is typically a hexadecimal or binary representation. Hash function is commonly used for data integrity verification. Follow below steps for hashing the field << **Previous invoice/note hash** >> (previousNoteHash).

The value in **previousNoteHash** is in hexadecimal consisting of a concatenation of below fields values.

Field description	Field name
Date time previous invoice was issued	dateTime
Previous invoice amount	totalAmtPaid
BRN	brn
Previous invoice number	invoiceIdentifier

For example when generating an invoice, if previous invoice contains below values:

Field Name	Field Value
dateTime	20231019 14:54:51
totalAmtPaid	1000
brn	I2365XXXX
invoiceIdentifier	AINV101

The concatenated value should be as below:

Concatenated value
20231019 14:54:511000I2365XXXXAINV101

The previousNoteHash should contain below value after hashing above concatenated value:

Field previousNoteHash value
A78C2C5C5C3E33F1B4808D437F84BB303E0832D07A49912466EAF7137DF31EDC

8.1.7.2. Steps for hashing the values to be present in previousNoteHash

	Steps
1	Concatenate fields as described above
2	Create a MessageDigest object with the "SHA-256" algorithm
3	Compute the hash which will result in an array of bytes
4	Convert the byte array to a hexadecimal representation

8.1.8. Response Payload (SUCCESS)

The attributes of a response after a successful invoice transmission are

Attributes		Description
responseId		Unique response ID generated by MRA
responseDateTime		Date time response was sent back to user
requestId		Request ID generated by user when sending request
status		Status [**SUCCESS**|**ERROR**|**HAS_ERROR**] of request
environment		Status of EBS [**TEST**|**LIVE**]
infoMessages		List of warning messages with warning code and description
errorMessages		List of error messages with error code and description
fiscalisedInvoices		Details related to fiscalised invoice
	invoiceIdentifier	Invoice Identifier of transaction
	irn	A unique identification provided by MRA in case of success status
	qrCode	The QR Code provided by MRA for the transaction in case of success status. The generated QR code is a string representing the Base64-encoded QR Code PNG image data and it is not encrypted. (Refer to section 8.1.13).
	status	The status of invoice transaction [**SUCCESS**|**ERROR**]
	warningMessages	List of warning messages with warning code and description
	errorMessages	List of error messages with error code and description

8.1.9. Response Payload (ERRORS)

The attributes of a response after a failed invoice transmission are

Attributes		Description
responseId		Unique response ID generated by MRA
responseDateTime		Date time response was sent back to consuming client
requestId		Request ID generated by consuming client when sending request
status		Status [**SUCCESS**|**ERROR**|**HAS_ERROR**] of request sent
environment		Environment from where response was sent [**TEST**|**LIVE**]
infoMessages		List of warning messages with warning code and description
errorMessages		List of error messages with error code and description
fiscalisedInvoices		Details related to fiscalised invoice
	invoiceIdentifier	Invoice number of transaction
	irn	Blank in case of error status
	qrCode	Blank in case of error status
	status	The status of invoice transaction [**SUCCESS**|**ERROR**]
	warningMessages	List of warning messages with warning code and description
	errorMessages	List of error messages with error code and description

8.1.10. Sample JSON Response (Success)

```
{
  "responseId": "479259408032023133718505",
  "responseDateTime": "20230308 13:37:18",
  "requestId": "RequestId1330",
  "status": "SUCCESS",
  "environment": "TEST",
  "infoMessages": null,
  "errorMessages": null,
  "fiscalisedInvoices": [
    {
      "invoiceIdentifier": "test1",
      "irn": "8b5108b0-97fa-36bd-835f-20eab5dcfef2",
      "qrCode":
"iVBORw0KGgoAAAANSUhEUgAAAV4AAAFeAQAAAADlUEq3AAACB0lEQVR4Xu2aMZKDMAxFlaFImSNwFI4WjsZR
OAIlRWa9+l+G9WZnM2kiN/83GOmZ5o9kORMr7+thz5EXEtxKcCvBrQS3EtxKcCvBrSq8WWj0yKWYTevNU2b3p
SbugvPhKx/LuF0dLrPZbR9KcXhkQnAXOPyqKZuQuhB2c7EQ3A+uqcXfYiG4P1xQTdTuARaa4G4wH0gNX76r+A
FU/u91glNgC40bDiAwx6ImBOfDPzqr6Vq+nnOCW30cDgfp10Bmve1ocRik94gIzoZpHCa0qCbWFxycfYk2+Lu
sBCfBtAnHzSMuNYZhoMzTkRKcDrOsIuJCNRmsdK10MCQ4FQ6bZmxBi/Om570OmjwFKwX3gGvK5+cIvOx1gnNg
MJ7yu8xxqXlUKxEJSHAqvPG4icUxPxMu6HXcLjgbLrRp9gktYMzP7H6YnyMiOBsOhqkBU4HBQTC00v7YLTgDr
imLCW22OHewPRaU4Hy4djY4OEXWD6AxpjjBPeCa4qjGYaDWFzzldwRnw6e4KxYxFRzfQUxwKswDyDWe9XX8yy
Ii8QHBuXAYt3hns+MkqmPzxiun4B7w6VekzpkNivuO4H4wWlzUFxbTevwgILgjjNfZIBjHyIuyEvxJmA9PgYF
xvN24cSubnuAeMC1jNRUMZku8wkr+jCm4A/yeBLcS3EpwK8GtBLcS3Epwqw/C3196CtS+PdRKAAAAAElFTkSu
QmCC",
      "status": "SUCCESS",
      "warningMessages": null,
      "errorMessages": null
    }
  ]
}
```

8.1.11. Sample JSON Response (Error)

```
{
  "responseId": "47925950803202313385581",
  "responseDateTime": "20230308 13:38:55",
  "requestId": "RequestId1330",
  "status": "ERROR",
  "environment": "TEST",
  "infoMessages": null,
  "errorMessages": null,
  "fiscalisedInvoices": [
    {
      "invoiceIdentifier": "test1",
      "irn": null,
      "qrCode": null,
      "status": "ERROR",
      "warningMessages": null,
      "errorMessages": [
        {
          "code": "LV_ERR002",
          "description": "TAN of registered user does not match with TAN of seller in invoice details"
        }
      ]
    }
  ]
}
```

8.1.12. Steps to generate JSON for invoice submission

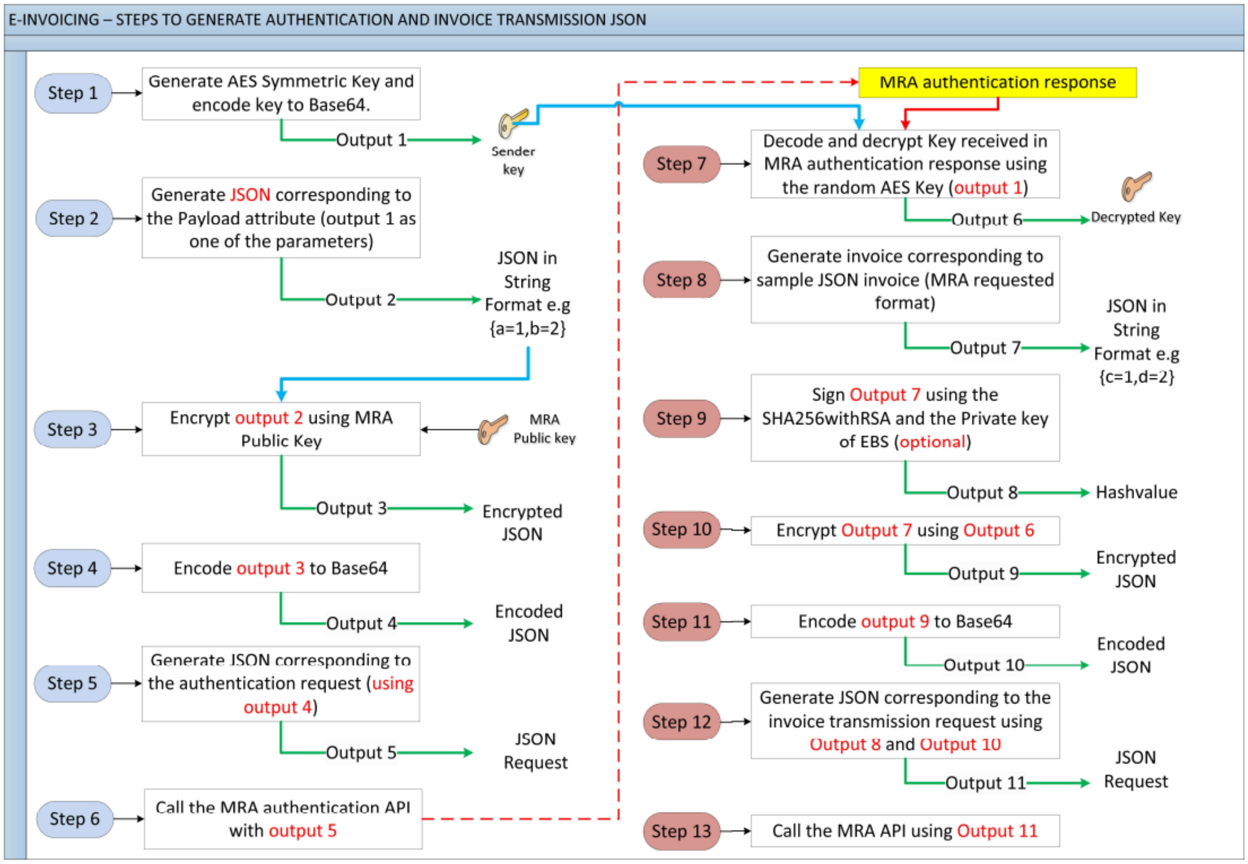

E-INVOICING – STEPS TO GENERATE AUTHENTICATION AND INVOICE TRANSMISSION JSON
Step 1
Generate AES Symmetric Key and encode key to Base64.
Output 1
Sender key
Step 2
Generate JSON corresponding to the Payload attribute (output 1 as one of the parameters)
Output 2
JSON in String Format e.g {a=1,b=2}
Step 3
Encrypt output 2 using MRA Public Key
MRA Public key
Output 3
Encrypted JSON
Step 4
Encode output 3 to Base64
Output 4
Encoded JSON
Step 5
Generate JSON corresponding to the authentication request (using output 4)
Output 5
JSON Request
Step 6
Call the MRA authentication API with output 5
MRA authentication response
Step 7
Decode and decrypt Key received in MRA authentication response using the random AES Key (output 1)
Output 6
Decrypted Key
Step 8
Generate invoice corresponding to sample JSON invoice (MRA requested format)
Output 7
JSON in String Format e.g {c=1,d=2}
Step 9
Sign Output 7 using the SHA256withRSA and the Private key of EBS (optional)
Output 8
Hashvalue
Step 10
Encrypt Output 7 using Output 6
Output 9
Encrypted JSON
Step 11
Encode output 9 to Base64
Output 10
Encoded JSON
Step 12
Generate JSON corresponding to the invoice transmission request using Output 8 and Output 10
Output 11
JSON Request
Step 13
Call the MRA API using Output 11

Figure 10: Flowchart to generate authentication and invoice transmission JSON

	Steps
1	Decode and decrypt Key received from MRA using the random AES Key that was generated in the authentication step
2	Generate invoice corresponding to sample JSON invoice (MRA requested format)
3	Sign JSON in step 2 using the SHA256withRSA and the Private key of EBS downloaded during the registration process **(optional)**
4	Encrypt JSON string from step 2 using decrypted Key from step 1 (The encrypted key that was received from MRA in the authentication response parameters). Note that the decrypted key should be decoded first before using same for encryption (refer to code snippet)
5	Encode encrypted JSON from step 4 to Base 64
6	Generate JSON payload corresponding to the invoice transmission request
7	Call the transmission API with **username**, **EBS MRA ID, and token** in the request header

8.1.13. Steps to use the MRA generated QR Code

Below steps to use the MRA generated QR Code and print same in receipts

1. **Decode Base64 String**: Decode the Base64-encoded string to obtain the binary PNG image data
2. **Create QR Code Image**: Use the binary PNG image data to create an image
3. **Print QR Code Image**: Render image or save it as a file in a format that's suitable for printing. The specific method depends on application and the tools or libraries that software developers are using.
4. **In case of any errors (connectivity issues) whereby MRA did not respond with a success message:** Follow step 8.1.14

8.1.14. Receipt Not Fiscalised

In the event of any connectivity issues with the MRA e-Invoicing System resulting in the invoice not being fiscalised, the text 'Not Yet Fiscalised' should be displayed in the same position where the QR Code image was intended to be displayed.

9. List of errors

Code	Reason	HTTP Status Code	Description/Action
ERR0001	Authentication Error	400	Unauthorized request
ERR0020	Authentication Error	400	Missing header parameters **Action requested:** Please submit request with correct header parameters.
ERR0021	Invalid request header/body format	400	Missing body parameters **Action requested:** Please submit request with correct header parameters.
ERR0022	Invalid request header/body format	400	Below reasons could entail this error code • Request DateTime should not be blank • Request Id should not be blank • Encrypted invoice data should not be blank • Request Id should not exceed 50 characters • Request DateTime should not exceed 17 characters • token should not be blank • username should not be blank • ebsMraId should not be blank • token should not exceed 255 characters • username should not exceed 100 characters • ebsMraId should not exceed 50 characters **Action requested:** Please submit request with correct format
ERR0050	Authorisation Error	401	Token sent is not valid
ERR0100	EBS validation failed	400	Below reasons could entail this error code • EBS status is not active • EBS test status is **PASS_CONFIRMED**. EBS should be reset to be able to proceed with submission of invoice
ERR0200	Decryption failed	400	Could not decrypt **encryptedInvoice**
ERR0300	Failed Signature	400	Could not validate signature
ERR0400	Invoice data mapping error	400	Invalid JSON format for invoice data Note the maximum number of items should not exceed 2000 **Action requested:** Please build the invoice data as per the correct schema and resubmit the request.
ERR0500	Invalid TAN	200	TAN of registered user does not match with TAN of seller in invoice details. **Action requested:** Please ensure that the correct TAN has been set in the request JSON and resubmit the request.
ERR0600	Invoice Data Validation Error	200	The decrypted invoice data string could not be validated as per the e-Invoice JSON Schema provided by MRA. **Action requested:** Please build the invoice data as per the correct schema and resubmit the request. ****Please refer to The Data Structure of an e-Invoice in the Guidelines Section for the JSON schema****
ERR0023	Internal Server Error	500	An internal server has occurred. **Action requested:** Please resubmit the request

10.Annex 1

10.1. Sample JSON for Invoice Transmission response

- Error Code ERR0020

```
{
  "status":  "ERROR",
  "errorMessages": [
    {
      "code":  "ERR0020",
       "description":  "Missing header parameters"
    }
  ]
}
```

- Error Code ERR0021

```
{
      "status":  "ERROR",
      "errorMessages":  [
            {
                  "code":  "ERR0021",
                  "description":  "Missing body parameters"
    }
  ]
}
```

- Error Code ERR0023

```
{
      "status":  "ERROR",
      "errorMessages":  [
            {
                  "code":  "ERR0023",
                  "description":  "An internal server error has occurred"
    }
  ]
}
```

- Error Code ERR0050

```
{
      "responseId":  "LT1679933969377330205O115",
      "responseDateTime":  "20230327 20:19:29",
      "requestId":  "1",
      "status":  "ERROR",
      "environment":  "TEST",
      "infoMessages":  null,
      "errorMessages":  [
            {
                  "code":  "ERR0050",
                  "description":  "Token is not valid"
    }
  ],
      "fiscalisedInvoices":  null
}
```

- Error Code ERR0100

```
{
    "responseId": "LT1679934875305612793341 4",
    "responseDateTime": "20230327 20:34:35",
    "requestId": "1",
    "status": "ERROR",
    "environment": "TEST",
    "infoMessages": null,
    "errorMessages": [
        {
            "code": "ERR0100",
            "description": "EBS status is not active"
        }
    ],
    "fiscalisedInvoices": null
}
```

- Error Code ERR0200

```
{
    "responseId": "LT1679935109795490850 8853",
    "responseDateTime": "20230327 20:38:29",
    "requestId": "1",
    "status": "ERROR",
    "environment": "TEST",
    "infoMessages": null,
    "errorMessages": [
        {
            "code": "ERR0200",
            "description": "Could not decrypt encryptedInvoice"
        }
    ],
    "fiscalisedInvoices": null
}
```

- Error Code ERR0300

```
{
    "responseId": "LT1679935109795290850853",
    "responseDateTime": "20230327 20:38:29",
    "requestId": "1",
    "status": "ERROR",
    "environment": "TEST",
    "infoMessages": null,
    "errorMessages": [
        {
            "code": " ERR0300",
            "description": " Could not validate signature: Invalid signature"
        }
    ],
    "fiscalisedInvoices": null
}
```

- Error Code ERR0400

```
{
    "responseId": "LT1679935748603263617101",
    "responseDateTime": "20230327 20:49:08",
    "requestId": "1",
    "status": "ERROR",
    "environment": "TEST",
    "infoMessages": null,
    "errorMessages": [
        {
            "code": "ERR0400",
            "description": "Invalid JSON format for decrypted encryptedInvoice"
        }
    ],
    "fiscalisedInvoices": null
}
```

- Error Code ERR0500

```
{
    "responseId": "LT16799358226919389842667",
    "responseDateTime": "20230327 20:50:22",
    "requestId": "1",
    "status": "ERROR",
    "environment": "TEST",
    "infoMessages": null,
    "errorMessages": null,
    "fiscalisedInvoices": [
        {
            "invoiceIdentifier": "test1",
            "irn": "",
            "qrCode": "",
            "status": "ERROR",
            "warningMessages": null,
            "errorMessages": [
                {
                    "code": "ERR0500",
                    "description": "TAN of registered user does not match with TAN of seller in decrypted 
encryptedInvoice"
                }
            ]
        }
    ]
}
```

- Error Code ERR0600

```
{
      "responseId":  "LT16799360192407258577721",
      "responseDateTime":  "20230327 20:53:39",
      "requestId":  "1",
      "status":  "ERROR",
      "environment":  "TEST",
      "infoMessages":  null,
      "errorMessages":  null,
      "fiscalisedInvoices":  [
            {
                  "invoiceIdentifier":  "test1",
                  "irn":  "",
                  "qrCode":  "",
                  "status":  "ERROR",
                  "warningMessages":  null,
                  "errorMessages":  [
                        {
                              "code":  "ERR0600",
                              "description":  "Type of person should contain only values: VATR/NVTR"

                        },
                        {
                              "code":  "ERR0600",
                              "description":  "Item Number 1: Tax Code of items should contain only values
: TC01/TC02/TC03/TC04/TC05"
                        },
                        {
                              "code":  "ERR0600",
                              "description":  "Item Number 5: Tax Code of items should contain only values
: TC01/TC02/TC03/TC04/TC05"
                        }
      ]
    }
  ]
}
```

11. Code snippets for encryption and decryption

11.1. In C#

Sample code to generate AES Key

```
Aes aesAlgorithm = Aes.Create();
aesAlgorithm.Mode = CipherMode.ECB;
aesAlgorithm.Padding = PaddingMode.PKCS7;
aesAlgorithm.KeySize = 256;
aesAlgorithm.GenerateKey();
```

Sample code to encode key to base 64 string

```
string aesKey = Convert.ToBase64String(aesAlgorithm.Key);
```

Sample code to encrypt using MRA Public Key

```
// Get the MRA public key which was initially downloaded
var cert = new X509Certificate2();
cert.Import(File.ReadAllBytes(@"C:\Users\Downloads\MRAPublicKey.crt"));
var publicKey = (RSACryptoServiceProvider)cert.PublicKey.Key;

// Convert authentication paylod (JSON) and encrypt payload using MRA public key
var bytes = publicKey.Encrypt(Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(payload)), false);
```

Sample code to decrypt Key from MRA

```
//Decrypt key received from MRA using the random AES Key generated in Step for authentication
ICryptoTransform decryptor = aesAlgorithm.CreateDecryptor();

//Decryption will be done in a memory stream through a CryptoStream object
using (MemoryStream msDecrypt = new MemoryStream(Convert.FromBase64String(mraEncryptedKey)))
using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
using (StreamReader srDecrypt = new StreamReader(csDecrypt))
{
     // Read the decrypted bytes from the decrypting stream
     // and place them in a string.
     return srDecrypt.ReadToEnd();
}
```

Sample code to decrypt invoice using MRA Key

```
ICryptoTransform decryptor = aesAlgorithm.CreateDecryptor();

//Decryption will be done in a memory stream through a CryptoStream object
using (MemoryStream msDecrypt = new MemoryStream(Convert.FromBase64String(encryptedInvoice)))
using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
using (StreamReader srDecrypt = new StreamReader(csDecrypt))
{
    // Read the decrypted bytes from the decrypting stream
    // and place them in a string.
    return srDecrypt.ReadToEnd();
}
```

Sample code to encrypt invoice using MRA Key

```
    using (var aesAlgorithm = Aes.Create())
    {
        aesAlgorithm.Key = Convert.FromBase64String(mrakey);
        aesAlgorithm.Mode = CipherMode.ECB;
        aesAlgorithm.Padding = PaddingMode.PKCS7;

        var encryptor = aesAlgorithm.CreateEncryptor();
        byte[] encryptedData;

        //Encryption will be done in a memory stream through a CryptoStream object
        using (var ms = new MemoryStream())
        {
            using (var cs = new CryptoStream(ms, encryptor, CryptoStreamMode.Write))
            {
                using (var sw = new StreamWriter(cs))
                {
                    sw.Write(JsonConvert.SerializeObject(obj));
                }

                encryptedData = ms.ToArray();
            }
        }
        return Convert.ToBase64String(encryptedData);
    }
```

Sample code to authenticate an EBS and submit an Invoice

```
 private const string CONST_USERNAME = "";
 private const string CONST_PASSWORD = "";
 private const string CONST_EBSMRAID = "";
 private const string AREA_CODE = "";
 private const string CONST_AUTH_URL = "https://vfisc.mra.mu/einvoice-token-service/token-api/generate-token";
 private const string CONST_INVC_URL = "https://vfisc.mra.mu/realtime/invoice/transmit";

private static void Main(string[] args)
{
    // Step 1: Generate random AES Key and encode key to base 64 string
    // This key will be used to decrypt the Key received from MRA
    Aes aesAlgorithm = generateRandomAesKey();
    string aesKey = Convert.ToBase64String(aesAlgorithm.Key);
    Console.WriteLine("1: " + aesKey);
    Console.WriteLine("2: " + aesKey.ToString());

    // Step 2: Generate authentication payload with attributes (username, password, encryptKey, refreshToken)
    AuthenticationPayload authPayload = generateAuthenticationPayload(aesKey);

    // Step 2: Generate JSON corresponding to the authPayload
    // Step 3: Encrypt JSON string from step 2 using MRA Public Key
    // Step 4: Encode encrypted JSON from step 3 to Base 64
    string encryptedPayload = encryptAuthPayload(authPayload);

    // Step 5: Generate JSON corresponding to the authentication request with a unique request ID
    AuthPayloadRequest authenticationRequest = generateAuthenticationRequest(encryptedPayload);

    // Step 6: Call the authentication API with username and EBS MRA ID in the request header
    // and JSON from Step 5 in the request body
    RestResponse responseAuth = authenticateEBSWithMRA(authenticationRequest);

    // check StatusCode in response
    if (responseAuth.StatusCode == HttpStatusCode.OK)
    {
        var mraResponse = new AuthenticationResponseMra();
        mraResponse = JsonConvert.DeserializeObject<AuthenticationResponseMra>(responseAuth.Content);

        // Response details from MRA
        Console.WriteLine("MRA response status: " + mraResponse.status);
        Console.WriteLine("MRA response requestId: " + mraResponse.requestId);
        Console.WriteLine("MRA response responseId: " + mraResponse.responseId);
        Console.WriteLine("MRA response token: " + mraResponse.token);
        Console.WriteLine("MRA response expiryDate: " + mraResponse.expiryDate);
        Console.WriteLine("MRA encrypted key: " + mraResponse.key);

        // Step 7: Decrypt key received from MRA using the random AES Key generated in Step 1
        // The key provided by MRA should be used for encrypting invoices
        string mraKey = decryptKeyReceivedFromMRA(aesAlgorithm, mraResponse.key);
        byte[] decryptedMRAKey = DecryptDataWithAesKey(Convert.FromBase64String(mraResponse.key),
Convert.FromBase64String(aesKey));

        // Step 8: Generate sample invoice (list with 1 invoice)
        List<MRAInvoice> sampleInvoice = generateSampleInvoice();
        Console.WriteLine("sampleInvoice: " + JsonConvert.SerializeObject(sampleInvoice));

        // Step 9: Encrypt invoice from step 8 using decrypted Key from step 7
        string encryptedInvoice = encryptInvoice(aesAlgorithm, sampleInvoice, mraKey);
        // Generate payload for invoice submission to MRA
        MRAInvoiceRequest mraInvoiceRequest = generateInvoiceSubmissionPayload(encryptedInvoice);

        // Call the transmission API with username, EBS MRA ID, and token in the request header for invoice
transmission
        RestResponse transRestResponse = submitInvoiceToMRA(mraResponse.token, mraInvoiceRequest);

        // check StatusCode in response
        if (transRestResponse.StatusCode == HttpStatusCode.OK)
        {
            TransmissionResponseMra transmissionResponse = new TransmissionResponseMra();
            transmissionResponse =
JsonConvert.DeserializeObject<TransmissionResponseMra>(transRestResponse.Content);

            // Response details from MRA
            Console.WriteLine("responseId: " + transmissionResponse.responseId);
            Console.WriteLine("responseDateTime: " + transmissionResponse.responseDateTime);
            Console.WriteLine("requestId: " + transmissionResponse.requestId);
            Console.WriteLine("status: " + transmissionResponse.status);
            // Global errors / info messages
            Console.WriteLine("errorMessages: " + transmissionResponse.errorMessages);
            Console.WriteLine("infoMessages: " + transmissionResponse.infoMessages);
            // fiscalised invoice
            Console.WriteLine("fiscalisedInvoices: " + transmissionResponse.fiscalisedInvoices[0]);
            if (transmissionResponse.fiscalisedInvoices != null)
            {
                Console.WriteLine("fiscalisedInvoices invoiceIdentifier:" +
transmissionResponse.fiscalisedInvoices[0].invoiceIdentifier);
                Console.WriteLine("fiscalisedInvoices irn: " + transmissionResponse.fiscalisedInvoices[0].irn);
                Console.WriteLine("fiscalisedInvoices qrCode: " +
transmissionResponse.fiscalisedInvoices[0].qrCode);
                Console.WriteLine("fiscalisedInvoices errorMessages: " +
transmissionResponse.fiscalisedInvoices[0].errorMessages);
                if (transmissionResponse.fiscalisedInvoices[0].errorMessages != null)
                {
                    // for each error messages, loop to display list of errors
                    foreach (var err in transmissionResponse.fiscalisedInvoices[0].errorMessages)
                    {
                        Console.WriteLine(err.code);
                        Console.WriteLine(err.description);
                    }
                }
            }
        }
    }
    else
    {
        // ToDo cater for errors here
    }
    Console.WriteLine("finsh");
}

/**
 * Function to generate ramdom AES Key
 */
private static Aes generateRandomAesKey()
{
    var aesAlgorithm = Aes.Create();
    aesAlgorithm.Mode = CipherMode.ECB;
    aesAlgorithm.Padding = PaddingMode.PKCS7;
    aesAlgorithm.KeySize = 256;
    aesAlgorithm.GenerateKey();
    return aesAlgorithm;
}

/**
 * generate Authentication Payload with attributes (username, password, encryptKey, refreshToken)
 */
private static AuthenticationPayload generateAuthenticationPayload(String aloKey)
{
    return new AuthenticationPayload
    {
        encryptKey = aloKey,
        username = CONST_USERNAME,
        password = CONST_PASSWORD,
        refreshToken = true
    };
}

/**
 * Generate the authentication request
 */
private static AuthPayloadRequest generateAuthenticationRequest(String encryptedAuthPayload)
{
    return new AuthPayloadRequest
    {
        requestId = Guid.NewGuid().ToString("N"),
        payLoad = encryptedAuthPayload
    };
}

/**
 * EncryptAuthPayloadKey - encrypt the authentication payload
 * 1. Get the MRA public key
 * 2. Convert the authentication payload to JSON
 * 3. Encrypt JSON string from step 2 using MRA Public Key
 * 4. Encode encrypted JSON from step 3 to Base 64
 */
private static string encryptAuthPayload(AuthenticationPayload payload)
{
    // Get the MRA public key which was initially downloaded
    var cert = new X509Certificate2();
    cert.Import(File.ReadAllBytes(@"C:\Users\MRAPublicKey.crt"));
    var publicKey = (RSACryptoServiceProvider)cert.PublicKey.Key;

    // Convert authentication paylod to JSON and encrypt payload using MRA public key
    var bytes = publicKey.Encrypt(Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(payload)), false);
    // Encode encrypted JSON from step 3 to Base 64
    return Convert.ToBase64String(bytes);
}

private static RestResponse authenticateEBSWithMRA(AuthPayloadRequest authPayloadReq)
{
    ServicePointManager.SecurityProtocol |= SecurityProtocolType.Tls11 | SecurityProtocolType.Tls12;
    RestClient restClientAuth = new RestClient(CONST_AUTH_URL);
    var requestAuth = new RestRequest(CONST_AUTH_URL, Method.Post);
    requestAuth.AddHeader("ebsMraId", CONST_EBSMRAID);
    requestAuth.AddHeader("Content-Type", "application/json");
    requestAuth.AddHeader("username", CONST_USERNAME);
    requestAuth.AddParameter("application/json", JsonConvert.SerializeObject(authPayloadReq),
ParameterType.RequestBody);
    return (RestResponse)restClientAuth.Execute(requestAuth);
}


/**
 * Decrypt - Decrypt the encrypted Key received from MRA using the AES Key that was sent in the authentication
payload
 */
private static string decryptKeyReceivedFromMRA(Aes aesAlgorithm, string mraEncryptedKey)
{
    ICryptoTransform decryptor = aesAlgorithm.CreateDecryptor();
    //Decrypt key received from MRA using the random AES Key generated in Step for authentication

    //Decryption will be done in a memory stream through a CryptoStream object
    using (MemoryStream msDecrypt = new MemoryStream(Convert.FromBase64String(mraEncryptedKey)))
    using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
    using (StreamReader srDecrypt = new StreamReader(csDecrypt))
    {
         // Read the decrypted bytes from the decrypting stream
         // and place them in a string.
         return srDecrypt.ReadToEnd();
    }
}
```

```
private static byte[] DecryptDataWithAesKey(byte[] encryptedData, byte[] aesKey)
{
    using (AesCryptoServiceProvider aesProvider = new AesCryptoServiceProvider())
    {
        aesProvider.Mode = CipherMode.ECB;
        aesProvider.Padding = PaddingMode.PKCS7;
        aesProvider.KeySize = 256;
        aesProvider.Key = aesKey;

        ICryptoTransform decryptor = aesProvider.CreateDecryptor();
        using (MemoryStream msDecrypt = new MemoryStream(encryptedData))
        using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
        using (StreamReader srDecrypt = new StreamReader(csDecrypt))
        {
            return Convert.FromBase64String(srDecrypt.ReadToEnd());
        }
    }
}

private static string decryptInvoice(Aes aesAlgorithm, string encryptedInvoice)
{
    ICryptoTransform decryptor = aesAlgorithm.CreateDecryptor();

    //Decryption will be done in a memory stream through a CryptoStream object
    using (MemoryStream msDecrypt = new MemoryStream(Convert.FromBase64String(encryptedInvoice)))
    using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
    using (StreamReader srDecrypt = new StreamReader(csDecrypt))
    {
        // Read the decrypted bytes from the decrypting stream
        // and place them in a string.
        return srDecrypt.ReadToEnd();
    }
}

private static List<MRAInvoice> generateSampleInvoice()
{
    // generate one item
    var itemList = new List<ItemList>();
    itemList.Add(new ItemList
    {
        itemNo = "",
        taxCode = "TC01",
        nature = "GOODS",
        productCodeMra = "",
        productCodeOwn = "",
        itemDesc = "",
        quantity = "",
        unitPrice = "",
        discount = "",
        discountedValue = "",
        amtWoVatCur = "",
        amtWoVatMur = "",
        vatAmt = "",
        totalPrice = ""
    });

    var invoiceList = new List<MRAInvoice>();
    // generate one invoice with one item
    var sampleInvoice = new MRAInvoice
    {
        invoiceCounter = "1",
        transactionType = "B2C",
        personType = "VATR",
        invoiceTypeDesc = "STD",
        currency = "MUR",
        invoiceIdentifier = "test1",
        invoiceRefIdentifier = "",
        previousNoteHash = "prevNote",
        reasonStated = "",
        totalVatAmount = "3400",
        totalAmtWoVatCur = "3000",
        totalAmtWoVatMur = "6400",
        totalAmtPaid = "10",
        dateTimeInvoiceIssued = "20221012 10:40:30",
        seller = new SellerTax
        {
            name = "",
            tradeName = "",
            tan = "",
            brn = "",
            businessAddr = "",
            businessPhoneNo = "",
            ebsCounterNo = ""
        },
        buyer = new BuyerTax
        {
            businessAddr = "",
            brn = "",
            buyerType = "",
            name = "",
            nic = "",
            tan = ""
        },
        itemList = itemList,
        salesTransactions = "CASH",
    };
    invoiceList.Add(sampleInvoice);
    return invoiceList;
}
private static MRAInvoiceRequest generateInvoiceSubmissionPayload(String encryptedInvoice)
{
    var resultInvoice = new MRAInvoiceRequest
    {
        requestId = "1000001",
        requestDateTime = "20230901 23:00:01",
        encryptedInvoice = encryptedInvoice
    };
    return resultInvoice;
}

/**
 * EncryptInvoice - Encrypt invoice with key received in the authentication endpoint
 */
private static string encryptInvoice(Aes aes, object obj, string key)
{
    using (var aesAlgorithm = Aes.Create())
    {
        aesAlgorithm.Key = Convert.FromBase64String(key);
        aesAlgorithm.Padding = aes.Padding;
        aesAlgorithm.Mode = aes.Mode;

        var encryptor = aesAlgorithm.CreateEncryptor();
        byte[] encryptedData;

        //Encryption will be done in a memory stream through a CryptoStream object
        using (var ms = new MemoryStream())
        {
            using (var cs = new CryptoStream(ms, encryptor, CryptoStreamMode.Write))
            {
                using (var sw = new StreamWriter(cs))
                {
                    sw.Write(JsonConvert.SerializeObject(obj));
                }
                encryptedData = ms.ToArray();
            }
        }

        var testmsg = decryptInvoice(aesAlgorithm, Convert.ToBase64String(encryptedData));
        Console.WriteLine("DecryptInvoice: " + testmsg);
        return Convert.ToBase64String(encryptedData);
    }
}

private static byte[] encryptInvoice1(string jsonString, byte[] aesKey)
{
    byte[] encryptedData;
    if (string.IsNullOrWhiteSpace(jsonString) || aesKey == null || aesKey.Length == 0)
        return null;
    using (var aesProvider = Aes.Create())
    {
        aesProvider.Key = aesKey;
        aesProvider.Padding = PaddingMode.PKCS7;
        aesProvider.Mode = CipherMode.ECB;
        aesProvider.KeySize = 256;

        var encryptor = aesProvider.CreateEncryptor();
        //Encryption will be done in a memory stream through a CryptoStream object
        using (var ms = new MemoryStream())
        {
            using (var cs = new CryptoStream(ms, encryptor, CryptoStreamMode.Write))
            {
                using (var sw = new StreamWriter(cs))
                {
                    sw.Write(jsonString);
                }
                encryptedData = ms.ToArray();
            }
        }

        var testmsg = decryptInvoice(aesProvider, Convert.ToBase64String(encryptedData));
        Console.WriteLine("DecryptInvoice: " + testmsg);
        return encryptedData;
    }
}

private static RestResponse submitInvoiceToMRA(String token, MRAInvoiceRequest mraInvoiceRequest)
{
    ServicePointManager.SecurityProtocol |= SecurityProtocolType.Tls11 | SecurityProtocolType.Tls12;
    RestClient restClientInv = new RestClient(CONST_INVC_URL);
    var requestInv = new RestRequest(CONST_INVC_URL, Method.Post);
    requestInv.AddHeader("Content-Type", "application/json");
    requestInv.AddHeader("username", CONST_USERNAME);
    requestInv.AddHeader("ebsMraId", CONST_EBSMRAID);
    // Area code received during the registration process
    requestInv.AddHeader("areaCode", AREA_CODE);
    requestInv.AddHeader("token", token);
    requestInv.AddParameter("application/json", JsonConvert.SerializeObject(mraInvoiceRequest),
ParameterType.RequestBody);
    return (RestResponse)restClientInv.Execute(requestInv);
}
public class AuthPayloadRequest
{
    [JsonProperty("requestId")] public string requestId { get; set; }
    [JsonProperty("payload")] public string payLoad { get; set; }
}
public class AuthenticationPayload
{
    [JsonProperty("username")] public string username { get; set; }
    [JsonProperty("password")] public string password { get; set; }
    [JsonProperty("encryptKey")] public string encryptKey { get; set; }
    [JsonProperty("refreshToken")] public bool refreshToken { get; set; }
}
public class SellerTax
{
    [JsonProperty("businessAddr")] public string businessAddr { get; set; }
    [JsonProperty("brn")] public string brn { get; set; }
    [JsonProperty("businessPhoneNo")] public string businessPhoneNo { get; set; }
    [JsonProperty("ebsCounterNo")] public string ebsCounterNo { get; set; }
    [JsonProperty("name")] public string name { get; set; }
    [JsonProperty("tan")] public string tan { get; set; }
    [JsonProperty("tradeName")] public string tradeName { get; set; }
}
public class BuyerTax
{
    [JsonProperty("businessAddr")] public string businessAddr { get; set; }
    [JsonProperty("brn")] public string brn { get; set; }
    [JsonProperty("buyerType")] public string buyerType { get; set; }
    [JsonProperty("name")] public string name { get; set; }
    [JsonProperty("nic")] public string nic { get; set; }
    [JsonProperty("tan")] public string tan { get; set; }
}
public class MRAInvoice
{
    [JsonProperty("invoiceCounter")] public string invoiceCounter { get; set; }
    [JsonProperty("transactionType")] public string transactionType { get; set; }
    [JsonProperty("personType")] public string personType { get; set; }
    [JsonProperty("invoiceTypeDesc")] public string invoiceTypeDesc { get; set; }
    [JsonProperty("currency")] public string currency { get; set; }
    [JsonProperty("invoiceIdentifier")] public string invoiceIdentifier { get; set; }
    [JsonProperty("invoiceRefIdentifier")] public string invoiceRefIdentifier { get; set; }
    [JsonProperty("previousNoteHash")] public string previousNoteHash { get; set; }
    [JsonProperty("reasonStated")] public string reasonStated { get; set; }
    [JsonProperty("totalVatAmount")] public string totalVatAmount { get; set; }
    [JsonProperty("totalAmtWoVatCur")] public string totalAmtWoVatCur { get; set; }
    [JsonProperty("totalAmtWoVatMur")] public string totalAmtWoVatMur { get; set; }
    [JsonProperty("totalAmtPaid")] public string totalAmtPaid { get; set; }
    [JsonProperty("dateTimeInvoiceIssued")] public string dateTimeInvoiceIssued { get; set; }
    [JsonProperty("itemList")] public List<ItemList> itemList { get; set; }
    [JsonProperty("salesTransactions")] public string salesTransactions { get; set; }
    [JsonProperty("buyer")] public BuyerTax buyer { get; set; }
    [JsonProperty("seller")] public SellerTax seller { get; set; }
}


public class ItemList
{
    [JsonProperty("itemNo")] public string itemNo { get; set; }
    [JsonProperty("taxCode")] public string taxCode { get; set; }
    [JsonProperty("nature")] public string nature { get; set; }
    [JsonProperty("productCodeMra")] public string productCodeMra { get; set; }
    [JsonProperty("productCodeOwn")] public string productCodeOwn { get; set; }
    [JsonProperty("itemDesc")] public string itemDesc { get; set; }
    [JsonProperty("quantity")] public string quantity { get; set; }
    [JsonProperty("unitPrice")] public string unitPrice { get; set; }
    [JsonProperty("discount")] public string discount { get; set; }
    [JsonProperty("discountedValue")] public string discountedValue { get; set; }
    [JsonProperty("amtWoVatCur")] public string amtWoVatCur { get; set; }
    [JsonProperty("amtWoVatMur")] public string amtWoVatMur { get; set; }
    [JsonProperty("vatAmt")] public string vatAmt { get; set; }
    [JsonProperty("totalPrice")] public string totalPrice { get; set; }
}
public class AuthenticationResponseMra
{
    [JsonProperty("responseId")] public string responseId { get; set; }
    [JsonProperty("requestId")] public string requestId { get; set; }
    [JsonProperty("status")] public string status { get; set; }
    [JsonProperty("token")] public string token { get; set; }
    [JsonProperty("key")] public string key { get; set; }
    [JsonProperty("expiryDate")] public string expiryDate { get; set; }

}

public class MRAInvoiceRequest
{
    public string encryptedInvoice { get; set; }
    public string requestDateTime { get; set; }
    public string requestId { get; set; }
}

public class TransmissionResponseMra
{
    [JsonProperty("responseId")] public string responseId { get; set; }
    [JsonProperty("responseDateTime")] public string responseDateTime { get; set; }
    [JsonProperty("requestId")] public string requestId { get; set; }
    [JsonProperty("status")] public string status { get; set; }
    [JsonProperty("environment")] public string environment { get; set; }
    [JsonProperty("infoMessages")] public List<Messages> infoMessages { get; set; }
    [JsonProperty("errorMessages")] public List<Messages> errorMessages { get; set; }
    [JsonProperty("fiscalisedInvoices")] public List<FiscalisedInvoices> fiscalisedInvoices { get; set; }
}

public class FiscalisedInvoices
{
    [JsonProperty("invoiceIdentifier")] public string invoiceIdentifier { get; set; }
    [JsonProperty("irn")] public string irn { get; set; }
    [JsonProperty("qrCode")] public string qrCode { get; set; }
    [JsonProperty("status")] public string status { get; set; }
    [JsonProperty("warningMessages")] public List<Messages> warningMessages { get; set; }
    [JsonProperty("errorMessages")] public List<Messages> errorMessages { get; set; }
    [JsonProperty("infoMessages")] public List<Messages> infoMessages { get; set; }
}

public class Messages
{
    [JsonProperty("code")] public string code { get; set; }
    [JsonProperty("description")] public string description { get; set; }
}
```

11.2. In Java

Sample code to generate random AES Key

```
SecureRandom rand = new SecureRandom();
KeyGenerator generator = KeyGenerator.getInstance("AES");
generator.init(256, rand);
SecretKey secretKey = generator.generateKey();
```

Sample code to encode key to base 64 string

```
byte[] rawData = secretKey.getEncoded();
String encodedKey = Base64.getEncoder().encodeToString(rawData);
```

Sample code to encrypt using MRA Public Key

```
PublicKey pkey = null;
try(FileInputStream fis = new FileInputStream(“path of MRA Public Key”)) {
    CertificateFactory cf = CertificateFactory.getInstance("X.509");
    X509Certificate cert = (X509Certificate) cf.generateCertificate(fis);
    pkey = cert.getPublicKey();
   }
} catch (Exception e) {
  // TODO
}

Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.ENCRYPT_MODE, pkey);
String encodedString = Base64.getEncoder().encodeToString((cipher.doFinal(msg.getBytes("UTF-8"))));
```

Sample code to decrypt Key from MRA

```
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
cipher.init(Cipher.DECRYPT_MODE, secretKey);
String decryptedKey = new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
```

Sample code to encrypt invoice using MRA Key

```
byte[] decodedKey = Base64.getDecoder().decode(decryptedKey);
SecretKey secretKey = new SecretKeySpec(decodedKey, 0, decodedKey.length, "AES");

Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, secretKey);
String encryptedString = Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
```

11.3. In PHP

Sample code to generate random AES Key

```
// Generate a random key
$encryptionKey = openssl_random_pseudo_bytes(32);
$aeskey = base64_encode($encryptionKey);
echo "\n aeskey: " .$aeskey;
```

Sample code to encrypt using MRA Public Key

```
openssl_public_encrypt(json_encode($payload), $encrypted_string, $pub_key,
OPENSSL_PKCS1_PADDING);
```

Sample code to decrypt Key from MRA

```
$mraKey = 'encrypted key received from MRA’;
// In PHP for decrypt the $mraKey should not be decoded.
// aeskey should be decoded
// Algorithm should be AES-256-ECB
$decryptedKey = openssl_decrypt($mraKey, 'AES-256-ECB', base64_decode($aeskey));
echo "\n decryptedKey: " .$decryptedKey;
```

Sample code to encrypt invoice using MRA Key

```
// json_encode converts JSON to string
$invoiceData = json_encode($arInvoice);
echo "\n invoiceData: " .$invoiceData;

// Algorithm should be AES-256-ECB
$encryptedInvoice = openssl_encrypt($invoiceData, 'AES-256-ECB', base64_decode($decryptedKey),
OPENSSL_RAW_DATA);

// Encrypted invoice should be encoded
$payload = base64_encode($encryptedInvoice);
echo "\n payload: " .$payload;
```

Sample code to authenticate an EBS and submit an invoice (PHP version 8.2)

```
<!DOCTYPE HTML>
<html>
<head>
<title>PHP Example</title>
</head>
<body>

<?php

$ebsMraId = '';
$ebsMraUsername = '';
$ebsMraPassword = '';
$areaCode = '';
```

```
$publicKey = "";

// Generate a random AES key
$aesKey = openssl_random_pseudo_bytes(32); // 32 bytes for AES-256
// Convert the AES key to Base64 string
$aesKeyBase64 = base64_encode($aesKey);
echo 'AES KEY = ' .$aesKeyBase64. "<br>";

$payload = array(
    'encryptKey' => $aesKeyBase64,
    'username' => $ebsMraUsername,
    'password' => $ebsMraPassword,
    'refreshToken' => true
);
echo '<br>JSON authentication Payload = ' .json_encode($payload);

// Import the certificate
$certPath = 'C:/Users/MRAPublicKey.crt';
$certContent = file_get_contents($certPath);
$cert = openssl_x509_read($certContent);

// Extract the public key from the certificate
$pubKeyDetails = openssl_pkey_get_details(openssl_pkey_get_public($cert));
$publicKey = $pubKeyDetails['key'];

// Encrypt payload using MRA public key
$encryptedData = '';
openssl_public_encrypt(json_encode($payload), $encryptedData, $publicKey);

// Encode encrypted data to Base64
$base64EncodedData = base64_encode($encryptedData);
echo '<br><br>Encrypted payload = ' .$base64EncodedData. "<br>";

$postData = array (
    'requestId' => mt_rand(),
    'payload' => $base64EncodedData
);

$requestHeadersAuth = [
    'Content-Type: application/json',
    'ebsMraId: '.$ebsMraId,
    'username: '.$ebsMraUsername
];


$chAuth = curl_init();
curl_setopt($chAuth, CURLOPT_URL,"https://vfisc.mra.mu/einvoice-token-service/token-
api/generate-token");
curl_setopt($chAuth, CURLOPT_POST, 1);
curl_setopt($chAuth, CURLOPT_POSTFIELDS,json_encode($postData));  //Post Fields
curl_setopt($chAuth, CURLOPT_RETURNTRANSFER, true);
curl_setopt($chAuth, CURLOPT_HTTPHEADER, $requestHeadersAuth);
curl_setopt($chAuth, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($chAuth, CURLOPT_SSL_VERIFYPEER, 0);

$responseDataAuth = curl_exec($chAuth);
echo "<br> responseDataAuth: = " .$responseDataAuth;

// Decode the JSON response into a PHP associative array
$responseArray = json_decode($responseDataAuth, true);
```

```
foreach ($responseArray as $key => $value) {
    echo $key . "= " . $value . "<br><br>";
}

$token = $responseArray['token'];
echo "<br><br> token = " .$token;

//MRA key received from generate token
$mraKey = $responseArray['key'];
echo "<br><br> mraKey = " .$mraKey;

$requestId = mt_rand();
echo "<br><br> request id = " .$requestId;

$arInvoice = [
    'invoiceCounter' => $requestId,
    'transactionType' => 'B2C',
    'personType' => 'VATR',
    'invoiceTypeDesc' => 'STD',
    'currency' => 'MUR',
    'invoiceIdentifier' => 'SI-HO-231027094929',
    'invoiceRefIdentifier' => '',
    'previousNoteHash' => 'prevNote',
    'reasonStated' => '',
    'totalVatAmount' => '15.0',
    'totalAmtWoVatCur' => '200.00',
    'totalAmtWoVatMur' => '200.00',
    'invoiceTotal' => '215.00',
    'discountTotalAmount' => '0.0',
    'totalAmtPaid' => '215.0',
    'dateTimeInvoiceIssued' => date('Ymd H:i:s'),
    "salesTransactions" => "CASH",
    'seller' => [
        'name' => '',
        'tradeName' => '',
        'tan' => '',
        'brn' => '',
        'businessAddr' => '3Port Louis',
        'businessPhoneNo' => '',
        'ebsCounterNo' => 'a1'
    ],
    'buyer' => [
        'name' => '',
        'tan' => '',
        'brn' => '',
        'businessAddr' => 'Quatre Bornes',
        'buyerType' => 'VATR',
        'nic' => ''
    ],
    'itemList' => [
        [
            'itemNo' => '1',
            'taxCode' => 'TC01',
            'nature' => 'GOODS',
            'productCodeMra' => '',
            'productCodeOwn' => 'ITEMCODE01',
            'itemDesc' => 'ITEM NAME 01',
            'quantity' => '1',
            'unitPrice' => '110',
            'discount' => '0',
            'discountedValue' => '10',
            'amtWoVatCur' => '100',
            'amtWoVatMur' => '100',
            'vatAmt' => '15',
            'totalPrice' => '115'
        ]
    ],
];

$invoiceArray = array($arInvoice);
$jsonencode =    json_encode($invoiceArray);

echo "<br><br>  invoice json = " .$jsonencode;

//algorithm should be AES-256-ECB
$decryptedKey = openssl_decrypt($mraKey, 'AES-256-ECB', base64_decode($aesKeyBase64));
echo "<br><br>  decryptedKey: = " .$decryptedKey;

// algorithm should be AES-256-ECB
$encryptedInvoice = openssl_encrypt($jsonencode, 'AES-256-ECB', base64_decode($decryptedKey),
OPENSSL_RAW_DATA);

// encrypted invoice should be encoded
$payloadInv = base64_encode($encryptedInvoice);
echo "<br><br> payload: = " .$payloadInv;

$requestHeadersInv = [
    'Content-Type: application/json',
    'ebsMraId: '.$ebsMraId,
    'username: '.$ebsMraUsername,
    'areaCode: ' .$areaCode,
    'token: '.$token
];

$postDataInv = [
    'requestId' => $requestId,
    'requestDateTime' => date('Ymd H:i:s'),
    'signedHash' => '',
    'encryptedInvoice' => $payloadInv
];

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL,"https://vfisc.mra.mu/realtime/invoice/transmit");
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS,json_encode($postDataInv));  //Post Fields
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, $requestHeadersInv);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);

$responseData = curl_exec($ch);
echo "<br><br>  responseData: = " .$responseData;

?>

</body>
</html>
```

11.4. VB.net

Sample code to authenticate an EBS and submit an invoice

```
Private Const CONST_USERNAME As String = ""
Private Const CONST_PASSWORD As String = ""
Private Const CONST_EBSMRAID As String = ""
Private Const CONST_AREACODE As String = ""
Private Const CONST_AUTH_URL As String = "https://vfisc.mra.mu/einvoice-token-service/token-api/generate-token"
Private Const CONST_INVC_URL As String = "https://vfisc.mra.mu/realtime/invoice/transmit"

Sub Main(args As String())
    ' Step 1: Generate random AES Key and encode key to base 64 string
    ' This key will be used to decrypt the Key received from MRA
    Dim aesAlgorithm As Aes = generateRandomAesKey()
    Dim aesKey As String = Convert.ToBase64String(aesAlgorithm.Key)
    Console.WriteLine("1: " & aesKey)
    Console.WriteLine("2: " & aesKey.ToString())

    ' Step 2: Generate authentication payload with attributes (username, password, encryptKey, refreshToken)
    Dim authPayload As AuthenticationPayload = generateAuthenticationPayload(aesKey)

    ' Step 2: Generate JSON corresponding to the authPayload
    ' Step 3: Encrypt JSON string from step 2 using MRA Public Key
    ' Step 4: Encode encrypted JSON from step 3 to Base 64
    Dim encryptedPayload As String = encryptAuthPayload(authPayload)

    ' Step 5: Generate JSON corresponding to the authentication request with a unique request ID
    Dim authenticationRequest As AuthPayloadRequest = generateAuthenticationRequest(encryptedPayload)

    ' Step 6: Call the authentication API with username and EBS MRA ID in the request header
    ' and JSON from Step 5 in the request body
    Dim responseAuth As RestResponse = authenticateEBSWithMRA(authenticationRequest)

    ' check StatusCode in response
    If responseAuth.StatusCode = HttpStatusCode.OK Then
        Dim mraResponse As New AuthenticationResponseMra()
        mraResponse = JsonConvert.DeserializeObject(Of AuthenticationResponseMra)(responseAuth.Content)

        ' Response details from MRA
        Console.WriteLine("MRA response status: " & mraResponse.status)
        Console.WriteLine("MRA response requestId: " & mraResponse.requestId)
        Console.WriteLine("MRA response responseId: " & mraResponse.responseId)
        Console.WriteLine("MRA response token: " & mraResponse.token)
        Console.WriteLine("MRA response expiryDate: " & mraResponse.expiryDate)
        Console.WriteLine("MRA encrypted key: " & mraResponse.key)

        ' Step 7: Decrypt key received from MRA using the random AES Key generated in Step 1
        ' The key provided by MRA should be used for encrypting invoices
        Dim mraKey As String = decryptKeyReceivedFromMRA(aesAlgorithm, mraResponse.key)
        Dim decryptedMRAKey As Byte() = DecryptDataWithAesKey(Convert.FromBase64String(mraResponse.key),
Convert.FromBase64String(aesKey))

        ' Step 8: Generate sample invoice (list with 1 invoice)
        Dim sampleInvoice As List(Of MRAInvoice) = generateSampleInvoice()
        Console.WriteLine("sampleInvoice: " & JsonConvert.SerializeObject(sampleInvoice))

        ' Step 9: Encrypt invoice from step 8 using decrypted Key from step 7
        Dim encryptedInvoice As String = encryptInvoice(aesAlgorithm, sampleInvoice, mraKey)
        ' Generate payload for invoice submission to MRA
        Dim mraInvoiceRequest As MRAInvoiceRequest = generateInvoiceSubmissionPayload(encryptedInvoice)

        ' Call the transmission API with username, EBS MRA ID, and token in the request header for invoice
transmission
        Dim transRestResponse As RestResponse = submitInvoiceToMRA(mraResponse.token, mraInvoiceRequest)

        ' check StatusCode in response
        If transRestResponse.StatusCode = HttpStatusCode.OK Then
            Dim transmissionResponse As New TransmissionResponseMra()
            transmissionResponse = JsonConvert.DeserializeObject(Of
TransmissionResponseMra)(transRestResponse.Content)

            ' Response details from MRA
            Console.WriteLine("responseId: " & transmissionResponse.responseId)
            Console.WriteLine("responseDateTime: " & transmissionResponse.responseDateTime)
            Console.WriteLine("requestId: " & transmissionResponse.requestId)
            Console.WriteLine("status: " & transmissionResponse.status)
            ' Global errors / info messages
            Console.WriteLine("errorMessages: " & transmissionResponse.errorMessages)
            Console.WriteLine("infoMessages: " & transmissionResponse.infoMessages)
        Else
            'todo cater for errors here
        End If

    Else
        ' ToDo cater for errors here
    End If
    Console.WriteLine("finsh")
End Sub

''' <summary>
''' Function to generate ramdom AES Key
''' </summary>
Private Function generateRandomAesKey() As Aes
    Dim aesAlgorithm = Aes.Create()
    aesAlgorithm.Mode = CipherMode.ECB
    aesAlgorithm.Padding = PaddingMode.PKCS7
    aesAlgorithm.KeySize = 256
    aesAlgorithm.GenerateKey()
    Return aesAlgorithm
End Function

''' <summary>
''' Generate Authentication Payload with attributes (username, password, encryptKey, refreshToken)
''' </summary>
Private Function generateAuthenticationPayload(aloKey As String) As AuthenticationPayload
    Return New AuthenticationPayload With {
        .encryptKey = aloKey,
        .username = CONST_USERNAME,
        .password = CONST_PASSWORD,
        .refreshToken = True
    }
End Function

''' <summary>
''' Generate the authentication request
''' </summary>
Private Function generateAuthenticationRequest(encryptedAuthPayload As String) As AuthPayloadRequest
    Return New AuthPayloadRequest With {
        .requestId = Guid.NewGuid().ToString("N"),
        .payLoad = encryptedAuthPayload
    }
End Function

''' <summary>
''' EncryptAuthPayloadKey - encrypt the authentication payload
''' 1. Get the MRA public key
''' 2. Convert the authentication payload to JSON
''' 3. Encrypt JSON string from step 2 using MRA Public Key
''' 4. Encode encrypted JSON from step 3 to Base 64
''' </summary>
Private Function encryptAuthPayload(payload As AuthenticationPayload) As String
    ' Get the MRA public key which was initially downloaded
    Dim cert = New X509Certificate2()
    cert.Import(File.ReadAllBytes("C:\Users\MRAPublicKey.crt"))
    Dim publicKey = CType(cert.PublicKey.Key, RSACryptoServiceProvider)

    ' Convert authentication paylod to JSON and encrypt payload using MRA public key
    Dim bytes = publicKey.Encrypt(Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(payload)), False)
    ' Encode encrypted JSON from step 3 to Base 64
    Return Convert.ToBase64String(bytes)
End Function

Private Function authenticateEBSWithMRA(authPayloadReq As AuthPayloadRequest) As RestResponse
    ServicePointManager.SecurityProtocol = ServicePointManager.SecurityProtocol Or SecurityProtocolType.Tls11 Or
SecurityProtocolType.Tls12
    Dim restClientAuth As New RestClient(CONST_AUTH_URL)
    Dim requestAuth = New RestRequest(CONST_AUTH_URL, Method.Post)
    requestAuth.AddHeader("ebsMraId", CONST_EBSMRAID)
    requestAuth.AddHeader("Content-Type", "application/json")
    requestAuth.AddHeader("username", CONST_USERNAME)
    requestAuth.AddParameter("application/json", JsonConvert.SerializeObject(authPayloadReq),
ParameterType.RequestBody)
    Return CType(restClientAuth.Execute(requestAuth), RestResponse)
End Function
```

```
''' <summary>
''' Decrypt - Decrypt the encrypted Key received from MRA using the AES Key that was sent in the authentication
payload
''' </summary>
Private Function decryptKeyReceivedFromMRA(aesAlgorithm As Aes, mraEncryptedKey As String) As String
    Dim decryptor As ICryptoTransform = aesAlgorithm.CreateDecryptor()
    'Decrypt key received from MRA using the random AES Key generated in Step for authentication

    'Decryption will be done in a memory stream through a CryptoStream object
    Using msDecrypt As New MemoryStream(Convert.FromBase64String(mraEncryptedKey)),
         csDecrypt As New CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read),
         srDecrypt As New StreamReader(csDecrypt)
        ' Read the decrypted bytes from the decrypting stream
        ' and place them in a string.
        Return srDecrypt.ReadToEnd()
    End Using
End Function

Private Function DecryptDataWithAesKey(encryptedData As Byte(), aesKey As Byte()) As Byte()
    Using aesProvider As New AesCryptoServiceProvider()
        aesProvider.Mode = CipherMode.ECB
        aesProvider.Padding = PaddingMode.PKCS7
        aesProvider.KeySize = 256
        aesProvider.Key = aesKey

        Dim decryptor = aesProvider.CreateDecryptor()

        'Decryption will be done in a memory stream through a CryptoStream object
        Using msDecrypt As New MemoryStream(encryptedData),
             csDecrypt As New CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read),
             srDecrypt As New StreamReader(csDecrypt)
            Return Convert.FromBase64String(srDecrypt.ReadToEnd())
        End Using
    End Using
End Function

Private Function decryptInvoice(aesAlgorithm As Aes, encryptedInvoice As String) As String
    Dim decryptor As ICryptoTransform = aesAlgorithm.CreateDecryptor()

    'Decryption will be done in a memory stream through a CryptoStream object
    Using msDecrypt As New MemoryStream(Convert.FromBase64String(encryptedInvoice)),
         csDecrypt As New CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read),
         srDecrypt As New StreamReader(csDecrypt)
        ' Read the decrypted bytes from the decrypting stream
        ' and place them in a string.
        Return srDecrypt.ReadToEnd()
    End Using
End Function

Private Function generateSampleInvoice() As List(Of MRAInvoice)
    ' generate one item
    Dim itemList = New List(Of ItemList) From {
        New ItemList With {
            .itemNo = "",
            .taxCode = "TC01",
            .nature = "GOODS",
            .productCodeMra = "",
            .productCodeOwn = "",
            .itemDesc = "",
            .quantity = "",
            .unitPrice = "",
            .discount = "",
            .discountedValue = "",
            .amtWoVatCur = "",
            .amtWoVatMur = "",
            .vatAmt = "",
            .totalPrice = ""
        }
    }

    ' generate one invoice with one item
    Dim sampleInvoice = New MRAInvoice With {
        .invoiceCounter = "1",
        .transactionType = "B2C",
        .personType = "VATR",
        .invoiceTypeDesc = "STD",
        .currency = "MUR",
        .invoiceIdentifier = "test1",
        .invoiceRefIdentifier = "",
        .previousNoteHash = "prevNote",
        .reasonStated = "",
        .totalVatAmount = "3400",
        .totalAmtWoVatCur = "3000",
        .totalAmtWoVatMur = "6400",
        .totalAmtPaid = "10",
        .dateTimeInvoiceIssued = "20221012 10:40:30",
        .seller = New SellerTax With {
            .name = "",
            .tradeName = "",
            .tan = "",
            .brn = "",
            .businessAddr = "",
            .businessPhoneNo = "",
            .ebsCounterNo = ""
        },
        .buyer = New BuyerTax With {
            .businessAddr = "",
            .brn = "",
            .buyerType = "",
            .name = "",
            .nic = "",
            .tan = ""
        },
        .itemList = itemList,
        .salesTransactions = "CASH"
    }

    Return New List(Of MRAInvoice) From {
        sampleInvoice
    }
End Function

Private Function generateInvoiceSubmissionPayload(encryptedInvoice As String) As MRAInvoiceRequest
    Return New MRAInvoiceRequest With {
        .requestId = "1000001",
        .requestDateTime = "20230901 23:00:01",
        .encryptedInvoice = encryptedInvoice
    }
End Function

''' <summary>
''' EncryptInvoice - Encrypt invoice with key received in the authentication endpoint
''' </summary>
Private Function encryptInvoice(aes As Aes, obj As Object, key As String) As String
    Using aesAlgorithm = Aes.Create()
        aesAlgorithm.Key = Convert.FromBase64String(key)
        aesAlgorithm.Padding = aes.Padding
        aesAlgorithm.Mode = aes.Mode

        Dim encryptor = aesAlgorithm.CreateEncryptor()
        Dim encryptedData As Byte()

        'Encryption will be done in a memory stream through a CryptoStream object
        Using ms = New MemoryStream()
            Using cs = New CryptoStream(ms, encryptor, CryptoStreamMode.Write)
                Using sw = New StreamWriter(cs)
                    sw.Write(JsonConvert.SerializeObject(obj))
                End Using

                encryptedData = ms.ToArray()
            End Using
        End Using

        Dim testmsg = decryptInvoice(aesAlgorithm, Convert.ToBase64String(encryptedData))
        Console.WriteLine("DecryptInvoice: " & testmsg)
        Return Convert.ToBase64String(encryptedData)
    End Using
End Function

Private Function encryptInvoice1(jsonString As String, aesKey As Byte()) As Byte()
    Dim encryptedData As Byte()
    If String.IsNullOrWhiteSpace(jsonString) OrElse aesKey Is Nothing OrElse aesKey.Length = 0 Then
        Return Nothing
    End If
    Using aesProvider = Aes.Create()
        aesProvider.Key = aesKey
        aesProvider.Padding = PaddingMode.PKCS7
        aesProvider.Mode = CipherMode.ECB
        aesProvider.KeySize = 256

        Dim encryptor = aesProvider.CreateEncryptor()

        'Encryption will be done in a memory stream through a CryptoStream object
        Using ms = New MemoryStream()
            Using cs = New CryptoStream(ms, encryptor, CryptoStreamMode.Write)
                Using sw = New StreamWriter(cs)
                    sw.Write(jsonString)
                End Using
                encryptedData = ms.ToArray()
            End Using
        End Using

        Dim testmsg = decryptInvoice(aesProvider, Convert.ToBase64String(encryptedData))
        Console.WriteLine("DecryptInvoice: " & testmsg)
        Return encryptedData
    End Using
End Function

Private Function submitInvoiceToMRA(token As String, mraInvoiceRequest As MRAInvoiceRequest) As RestResponse
    ServicePointManager.SecurityProtocol = ServicePointManager.SecurityProtocol Or SecurityProtocolType.Tls11 Or
SecurityProtocolType.Tls12
    Dim restClientInv As New RestClient(CONST_INVC_URL)
    Dim requestInv = New RestRequest(CONST_INVC_URL, Method.Post)
    requestInv.AddHeader("Content-Type", "application/json")
    requestInv.AddHeader("username", CONST_USERNAME)
    requestInv.AddHeader("ebsMraId", CONST_EBSMRAID)
    ' Area code received during the registration process
    requestInv.AddHeader("areaCode", CONST_AREACODE)
    requestInv.AddHeader("token", token)
    requestInv.AddParameter("application/json", JsonConvert.SerializeObject(mraInvoiceRequest),
ParameterType.RequestBody)
    Return CType(restClientInv.Execute(requestInv), RestResponse)
End Function

Public Class AuthPayloadRequest
    <JsonProperty("requestId")> Public Property requestId As String
    <JsonProperty("payload")> Public Property payLoad As String
End Class

Public Class AuthenticationPayload
    <JsonProperty("username")> Public Property username As String
    <JsonProperty("password")> Public Property password As String
    <JsonProperty("encryptKey")> Public Property encryptKey As String
    <JsonProperty("refreshToken")> Public Property refreshToken As Boolean
End Class

Public Class SellerTax
    <JsonProperty("businessAddr")> Public Property businessAddr As String
    <JsonProperty("brn")> Public Property brn As String
    <JsonProperty("businessPhoneNo")> Public Property businessPhoneNo As String
    <JsonProperty("ebsCounterNo")> Public Property ebsCounterNo As String
    <JsonProperty("name")> Public Property name As String
    <JsonProperty("tan")> Public Property tan As String
    <JsonProperty("tradeName")> Public Property tradeName As String
End Class

Public Class BuyerTax
    <JsonProperty("businessAddr")> Public Property businessAddr As String
    <JsonProperty("brn")> Public Property brn As String
    <JsonProperty("buyerType")> Public Property buyerType As String
    <JsonProperty("name")> Public Property name As String
    <JsonProperty("nic")> Public Property nic As String
    <JsonProperty("tan")> Public Property tan As String
End Class

Public Class MRAInvoice
    <JsonProperty("invoiceCounter")> Public Property invoiceCounter As String
    <JsonProperty("transactionType")> Public Property transactionType As String
    <JsonProperty("personType")> Public Property personType As String
    <JsonProperty("invoiceTypeDesc")> Public Property invoiceTypeDesc As String
    <JsonProperty("currency")> Public Property currency As String
    <JsonProperty("invoiceIdentifier")> Public Property invoiceIdentifier As String
    <JsonProperty("invoiceRefIdentifier")> Public Property invoiceRefIdentifier As String
    <JsonProperty("previousNoteHash")> Public Property previousNoteHash As String
    <JsonProperty("reasonStated")> Public Property reasonStated As String
    <JsonProperty("totalVatAmount")> Public Property totalVatAmount As String
    <JsonProperty("totalAmtWoVatCur")> Public Property totalAmtWoVatCur As String
    <JsonProperty("totalAmtWoVatMur")> Public Property totalAmtWoVatMur As String
    <JsonProperty("totalAmtPaid")> Public Property totalAmtPaid As String
    <JsonProperty("dateTimeInvoiceIssued")> Public Property dateTimeInvoiceIssued As String
    <JsonProperty("itemList")> Public Property itemList As List(Of ItemList)
    <JsonProperty("salesTransactions")> Public Property salesTransactions As String
    <JsonProperty("buyer")> Public Property buyer As BuyerTax
    <JsonProperty("seller")> Public Property seller As SellerTax
End Class

Public Class ItemList
    <JsonProperty("itemNo")> Public Property itemNo As String
    <JsonProperty("taxCode")> Public Property taxCode As String
    <JsonProperty("nature")> Public Property nature As String
    <JsonProperty("productCodeMra")> Public Property productCodeMra As String
    <JsonProperty("productCodeOwn")> Public Property productCodeOwn As String
    <JsonProperty("itemDesc")> Public Property itemDesc As String
    <JsonProperty("quantity")> Public Property quantity As String
    <JsonProperty("unitPrice")> Public Property unitPrice As String
    <JsonProperty("discount")> Public Property discount As String
    <JsonProperty("discountedValue")> Public Property discountedValue As String
    <JsonProperty("amtWoVatCur")> Public Property amtWoVatCur As String
    <JsonProperty("amtWoVatMur")> Public Property amtWoVatMur As String
    <JsonProperty("vatAmt")> Public Property vatAmt As String
    <JsonProperty("totalPrice")> Public Property totalPrice As String
End Class

Public Class MRAInvoiceRequest
    <JsonProperty("requestId")> Public Property requestId As String
    <JsonProperty("requestDateTime")> Public Property requestDateTime As String
    <JsonProperty("encryptedInvoice")> Public Property encryptedInvoice As String
End Class

Public Class AuthenticationResponseMra
    <JsonProperty("status")> Public Property status As String
    <JsonProperty("requestId")> Public Property requestId As String
    <JsonProperty("responseId")> Public Property responseId As String
    <JsonProperty("token")> Public Property token As String
    <JsonProperty("expiryDate")> Public Property expiryDate As String
    <JsonProperty("key")> Public Property key As String
End Class

Public Class TransmissionResponseMra
    <JsonProperty("responseId")> Public Property responseId As String
    <JsonProperty("responseDateTime")> Public Property responseDateTime As String
    <JsonProperty("requestId")> Public Property requestId As String
    <JsonProperty("status")> Public Property status As String
    <JsonProperty("errorMessages")> Public Property errorMessages As String
    <JsonProperty("infoMessages")> Public Property infoMessages As String
    <JsonProperty("fiscalisedInvoices")> Public Property fiscalisedInvoices As List(Of FiscalisedInvoices)
End Class

Public Class FiscalisedInvoices
    <JsonProperty("invoiceIdentifier")> Public Property invoiceIdentifier As String
    <JsonProperty("irn")> Public Property irn As String
    <JsonProperty("qrCode")> Public Property qrCode As String
    <JsonProperty("errorMessages")> Public Property errorMessages As List(Of ErrorMessages)
End Class

Public Class ErrorMessages
    <JsonProperty("code")> Public Property code As String
    <JsonProperty("description")> Public Property description As String
End Class
```