E-MITRA HACKATHON INTEGRATION DOCUMENT

Version 1.5

e-Mitra

Version Number	Author	Date
1.0	Sanjay Atreya	Feb 23, 2017
1.1	Parag Kachhawa/Sanjay Atreya	March 19, 2017
1.2	Parag Kachhawa/Sanjay Atreya	Aug 05, 2017
1.3	Akshay Sharma/Sanjay Atreya	Nov 21, 2017
1.4	Akshay Sharma/Sanjay Atreya	Mar 12, 2018
1.5	Akshay Sharma/Sanjay Atreya	Jul 17, 2018

Table of Contents

Glossary of Terms

Acronyms

Acronym	
AES	Advanced Encryption Standard
B2C	Business to Consumer
G2C	Government to Consumer
JSON	JavaScript Object Notation
MD5	Message-Digest algorithm 5
RISL	RajComp Info Service Limited
SSO	Single Sign On
UAT	User Acceptance Testing
URL	Uniform Resource Locator

Introduction

e-Mitra is an ambitious e-governance initiative of Government of Rajasthan (GoR) which is being implemented in all 33 Districts of the state using Public-Private Partnership (PPP) model for convenience and transparency to citizens in availing various services of the Government and Private Sectors under a single roof at their door steps using an e-platform. The services are delivered via counters known as CSC (Common Service Center) kiosks in Rural Areas and e-Mitra kiosks in urban areas and also online via eMitra portal. Hence, these counters provide services related to various departments in an integrated and easily accessible manner to people residing in rural as well as urban areas without any need for running around in government offices. The project has been operational since 2005. Initially it was functioning through a Client Server based Application Software developed by Department of IT&C. In 2010, the old Client Server Application was migrated to Web-based on-line e-Mitra application across all the 33 districts. Recently, a new generic module has been added to e-Mitra portal which allows end to end application and delivery of "Digitally Signed Certificates" such as Bonafide, Caste, Income, Solvency etc.

FetchDetails_Services:

Introduction :

Billing Data Fetch Details:

REQUEST

Request Method = POST
Request Parameter Name = **encData**

1. **Airtel Mobile Bill Postpaid**
   SrvId = 1214
   Sample Search Key: 9352423664
   Sample Data Decrypted = {"SRVID":"1214","searchKey":"9352423664","SSOID":"SSOTESTKIOSK"}

   Sample Encrypted Value =
   //wzeTciCLKFnevjkR+Y+9Av818ZKvBZLA4rU1FTCpA/2aoltEJROkqp79VPthK/hpYH7I9dZQAu0pNqjKPmn9IMUoazesKWqTNkvxnYv3I=

   URL :: http://103.203.139.158/webServicesRepository/getFetchDetailsEncryptedBySso?

   encData=//wzeTciCLKFnevjkR+Y+9Av818ZKvBZLA4rU1FTCpA/2aoltEJROkqp79VPthK/hpYH7I9dZQAu0pNqjKPmn9IMUoazesKWqTNkvxnYv3I=

2. **Idea Mobile Postpaid Bill**
   SrvId = 1220
   Sample Search Key : 8440042182
   Sample Data Decrypted =
   {"SRVID":"1220","searchKey":"8440042182","SSOID":"SSOTESTKIOSK"}

   Sample Encrypted Value =
   05jMkbnJAHEBPte1IWEhFxa8ODS01Axt7OQmAClhvthWKULyJWHRNY6X7/bqD53regvYLnGTf9AXc0qIvU9FBMPnlMViuAy7pth5bXa5jfs=

3. **MTS Mobile_Telephone_DataCard Postpaid Bill**
   SrvId = 1216
   Sample Search Key : 8432926694
   Sample Data Decrypted =
   {"SRVID":"1216","searchKey":"8432926694","SSOID":"SSOTESTKIOSK"}

   Sample Encrypted Value =
   tQYR/DVbqTax0kti0aKhcemqmxXQyqYcYbdr0hs+XxvwAEG4LNeGkFBUD2aUrSXTzfJaAJeCAKBu6rjaLUIusFBIz6uKhPSkUGtQMfhRb3o=

4. **Vodafone Mobile Postpaid Bill**
   SrvId = 1219
   Sample Search Key : 9982345598
   Sample Data Decrypted =
   {"SRVID":"1219","searchKey":"9982345598","SSOID":"SSOTESTKIOSK"}

Sample Encrypted Value =
ssmvVkFSulF8AWAPSqWsHmJQdUOAo73V0xvP0iTsxln3bLW8c656YNNVYlQTRTV2I1qtrbOyA7YynGoZA73PamtHzD2rFOv6c4wgTKd49P0=

5. **BSNL LANDLINE Bill**
   Srvld = 1221
   Sample Search key :: 1412441845, 1412441805
   Sample Data Decrypted =
   {"SRVID":"1221","searchKey":"1412441845","SSOID":"SSOTESTKIOSK"}

   Sample Encrypted Value =
   zFyiIE6vM8YFXzZ12/znA2LQ1AkhLlKg97u2E4MrRO6cmSDFFvoWUbJWmwliHG3iOrhf6Jve9sd1gIHCnDGY0R4JUYvokE3nBSbgXdH7sNk=

Encryption Type = AES
Encryption Key = E-m!tr@2016

Parameter Details				
#	Parameter Name	Description	Sample Value	Type
1	SSOID	SSO Id of Kiosk user	'SSOTESTKIOSK'	varchar
2	SRVID	Service id - this should be sent to eMitra for back to back transactions at eMitra	1214	varchar
3	SearchKey	Search key	9352423664	varchar

Examples

Sample request format

```
<form action="<POSTING_URL>" method="POST">
   <input type="hidden" name="encData" value="< encrypted data>" />
   <button type="submit">Submit</button>
</form>
```

Sample Response:

Sample Decrypted Response =

"Modified = {

""FetchDetails"": {

""TransactionDetails"": {

""ServiceName"" : ""Airtel Mobile"",

""officeID"":""DDD1"",

""BillAmount"": ""941.00"",

""ConsumerName"": ""Chetan Kumar Yadav"",
""consumerKeysValues"": ""9352423664"",
""partPaymentAllow"": ""1"",
""partPaymentType"": ""Both"",
""lookUpId"": ""6163298""
},
""BillDateils"": [
{
""LableName"": ""Consumer Name"",
""LableValue"": ""Chetan Kumar Yadav""
},
{
""LableName"": ""Account Number"",
""LableValue"": ""1116231291""
},
{
""LableName"": ""Mobile Number"",
""LableValue"": ""9352423664""
},
{
""LableName"": ""Amount Before Due Date"",
""LableValue"": ""931.00""
},
{
""LableName"": ""Amount After Due Date"",
""LableValue"": ""941.00""
},
{
""LableName"": ""Due Date"",
""LableValue"": ""NA""
},

```
    {
      ""LableName"": ""Bill Date"",
      ""LableValue"": ""NA""
    },
    {
      ""LableName"": ""Bill Cycle"",
      ""LableValue"": ""NA""
    },
    {
      ""LableName"": ""Bill Number"",
      ""LableValue"": ""NA""
    }
   ]
  }
}"
```


Sample Encrypted Response =

w2tZwKucTyOLM+cq+H1aRTUCQmP2FfowDEpHgs0Y/uvX8ffXHfZMPBZDxbYK4t5J0L857eNJnqOxC5f+yMA3
wnDvUAAtzMYnjhRFY/PPAai3V1pegD6ptNwZI21nxeWT/BYl7sfKgb2sAlyjimjNkzRmIUS/sSht2medbIy38wep
Lzs9txn6B4yFPVDtvj7fxn2CIgNZ15EvxPie1fcI6KqaFQPmqMhZzTDgsavJOhag3TJM0hK52iowW9nLNjmzoa0a
dCATFdQlsYEDWG3BlYPkuHLXTKcIPYtrhzHNKMCMGf1CRsQQ2eUqi4/pCCwgiBu7nRzSaJibz8zLlK+qcYNf2d
meIJnD3kTolYvkMjA1DCwoyKy9sc8rSfk8ntt/nhh8xrH6YqrQXI9Ar+WJFF1CFWyuOl8MhVGBC1SXgnp+BYwdd
zY0s5oCBlqzHs7saeCZZ6vxbgJvMESXeUCdPQ4BSLA4wk14spjl7q8Jo1ErRlPFlU+pfkCrBsspECujCFGfQ/iqfoV
UZMZmk0duHx6lkCAhSdDFuwlCoHE1ndaauYFeUFYIchRx6ZQAA2jdJ+ZWSb8xtSS/WSvOydt6ncqfgkCLIDhm
3rOYWbIViz4UaRl0tlbcCGyOBSiubR71GC2B/J0GERIQsCR+1VujQdqowWP+l02RyhM6SCsojpEX5BGKB2wVnsy
l8CU109VZClbu6UWySkljAS6w49dKT+XJR/z3qFpMWLxR3xDHReRzIojtd1fvK5UNprfle5j6NCCcGD8EMgcwG
DU3dMlGtzrjIruakbftmZblqroDy3ACYN2E1UOVeAdvIONsFMBViy1oIgY8IkZlJT8yPrAEBGK3Pnw1h8o0uUWNG
B83rHfMNETBH+5XYO5riE4CTWEPTYQPLniT1wqP9QWkknC0HreAsdxglT4BQIx9hzgZHW7qPwl5CIoak2afgP
WGU8L2Vk7ZUK7tF+AFLkcwtCW9KWsMySWCXAGAofmVXeh0M9tFGFHBRpxL4iEBLsWkqjVdUTwi7lt0SBINVg
Z4G0Oq93rqBQ1Y6ik/MFgtqslVo7RzxF+qeWFVq0LxBsDxecR8pxcK/TKsmdToc6OyL85h970INSG9stEeRysWp
KRaMQbnakhNfkJAPqr6QjA5G6oO

Payment Gateway Linking:

MERCHANT CODE = HACKATHON5.0
CHECKSUM_KEY = z/OjWt0XHtMfJDe

Transaction URL:

http://uat.rpp.rajasthan.gov.in/payments/v1/init

Merchant needs to *POST* the payment request on the above URL.

Request Format:

In this request, merchant post the transaction details on the payment gateway's Transaction URL without using any encryption

Sample Html form request from Merchant to Payment Gateway

```
<form action="<Transaction URL listed above>" method="POST">
   <input type="hidden" name="MERCHANTCODE" value="<MERCHANTCODE>" />
   <input type="hidden" name="PRN" value="<PRN>" />
   <input type="hidden" name="REQTIMESTAMP" value="<REQTIMESTAMP>" />
   <input type="hidden" name="PURPOSE" value="<PURPOSE>" />
   <input type="hidden" name="AMOUNT" value="<AMOUNT>" />
   <input type="hidden" name="SUCCESSURL" value="<SUCCESSURL>" />
   <input type="hidden" name="FAILUREURL" value="<FAILUREURL>" />
   <input type="hidden" name="CANCELURL" value="<CANCELURL>" />
   <input type="hidden" name="USERNAME" value="<USERNAME>" />
   <input type="hidden" name="USERMOBILE" value="<USERMOBILE>" />
   <input type="hidden" name="USEREMAIL" value="<USEREMAIL>" />
   <input type="hidden" name="UDF1" value="<UDF1>" />
   <input type="hidden" name="UDF2" value="<UDF2>" />
   <input type="hidden" name="UDF3" value="<UDF3>" />
   <input type="hidden" name="CHECKSUM" value="<CHECKSUM>" />
   <button type="submit">Submit</button>
</form>
```

Response Format

In this payment gateway post the payment details along with transaction details on the Merchant's SUCCESS/FAILURE/CANCEL URL without using any encryption. Merchant need to read parameters from "FORM" object of Request.

Sample Html form request from RPP to merchant

```
<form action="<MERCHANT RESPONSE URL provided in the request>" method="POST">
   <input type="hidden" name="MERCHANTCODE" value="<MERCHANTCODE>" />
   <input type="hidden" name="REQTIMESTAMP" value="<REQTIMESTAMP>" />
   <input type="hidden" name="PRN" value="<PRN>" />
   <input type="hidden" name="AMOUNT" value="<AMOUNT>" />
   <input type="hidden" name="RPPTXNID" value="<RPPTXNID>" />
   <input type="hidden" name="RPPTIMESTAMP" value="<RPPTIMESTAMP>" />
   <input type="hidden" name="PAYMENTAMOUNT" value="<PAYMENTAMOUNT>" />
```

```
        <input type="hidden" name="STATUS" value="<STATUS>" />
        <input type="hidden" name="PAYMENTMODE" value="<PAYMENTMODE>" />
        <input type="hidden" name="PAYMENTMODEBID" value="<PAYMENTMODEBID>" />
        <input type="hidden" name="PAYMENTMODETIMESTAMP" value="<PAYMENTMODETIMESTAMP>" />
        <input type="hidden" name="RESPONSECODE" value="<RESPONSECODE>" />
        <input type="hidden" name="RESPONSEMESSAGE" value="<RESPONSEMESSAGE>" />
        <input type="hidden" name="UDF1" value="<UDF1>" />
        <input type="hidden" name="UDF2" value="<UDF2>" />
        <input type="hidden" name="UDF3" value="<UDF3>" />
        <input type="hidden" name="CHECKSUM" value="<CHECKSUM>" />
        <button type="submit">Submit</button>
</form>
```

Request Parameters

Parameter Name	Parameter Type	Parameter Constraints	Comments/Description
MERCHANTCODE	Alphanumeric	Required Max length:32	This is the Payee Id/ Merchant Code Ex. HACKATHON2017
PRN	Alphanumeric	Required Max length:20	This is the Payment Reference Number - Merchant's Transaction ID which should be unique for every transaction. Ex. 123456
REQTIMESTAMP	Alphanumeric	Required Max length:17	Timestamp of the transaction at merchant while initiating transaction. Format: YYYYMMddHHmmssttt Example: 20160623132359958
PURPOSE	Alphanumeric	Required Max length:50	This is the purpose of the transaction that's need be display to the customer.
AMOUNT	Numeric	Required Max length:12	This is the base amount. i.e. Merchant Transaction Amount and it should be in decimal only Example: 100.50
SUCCESSURL	Alphanumeric	Required Max length:255	This parameter is the URL where PG post final response in case of successful transaction.
FAILUREURL	Alphanumeric	Required Max length:255	This parameter is the URL where PG post final response in case of failure transaction.
CANCELURL	Alphanumeric	Required Max length:255	This parameter is the URL where PG post final response in case of transaction cancellation.
USERNAME	Alpha	Required Max length: 50	Name of Customer (visible to user doing trans)

USERMOBILE	Numeric	Required Max length: 12	Customer's mobile number. (visible to user doing trans)
USEREMAIL	Alphanumeric	Required Max length: 50	Customer's email id. (visible to user doing trans)
UDF1	Alphanumeric	Optional Max length:255	User Define Parameter, an additional field that merchant can use to send any additional information about the transaction.
UDF2	Alphanumeric	Optional Max length:255	Same as "UDF1"
UDF3	Alphanumeric	Optional Max length:255	Same as "UDF1"
CHECKSUM	Alphanumeric	Required Max length:64	This is the string (Checksum) generated on merchant's server for data integrity checking. Ex. E059B24D93F38A62FDEE6BF2CAD48D9D5C49004713050F7783E6154FF3106311

RESPONSE PARAMETER:

Parameter Name	Parameter Type	Parameter Constraints	Comments/Description
MERCHANTCODE	Alphanumeric	Max length:32	This has the same value that was sent in the payment request.
REQTIMESTAMP	Alphanumeric	Length:17	Same as sent in request
PRN	Alphanumeric	Max length:20	This has the same value that was sent by you in the payment request.
AMOUNT	Numeric	Max length:12	This is the base amount sent in request(100.00)
RPPTXNID	Alphanumeric	Max length:20	Unique transaction reference number generated by PG for each transaction
RPPTIMESTAMP	Alphanumeric	Length:17	
PAYMENTAMOUNT	Numeric	Max length:12	This is the amount processed by the PG (100.00)
STATUS	Alpha		This is the status of the payment. This will have the following values:

			• SUCCESS • FAILED • PENDING
PAYMENTMODE	Alpha	Max length:255	This is the name of the selected payment mode by customer. Ex. Paytm/ SBBJ
PAYMENTMODEBID	Alphanumeric	Max length:20	Unique transaction id of payment mode selected
PAYMENTMODETIMESTAMP	Alphanumeric	Max length:17	Transaction timestamp of payment mode selected
RESPONSECODE	Numeric	Max length:4	This is the response code for the transaction status. All codes refer to a transaction failure or success with each code representing a different reason for failure.
RESPONSEMESSAGE	Alphanumeric	Max length:255	This is the human readable text message associated with the response code.
UDF1	Alphanumeric	Max length:255	Same as sent in request
UDF2	Alphanumeric	Max length:255	Same as sent in request
UDF3	Alphanumeric	Max length:255	Same as sent in request
CHECKSUM	Alphanumeric	Max length:64	This is the encrypted string (Checksum) generated on RPP server for data integrity checking.

Checksum calculation:

A. Checksum in Request

For checksum generation, first you need to generate a pipe separated string of certain request parameters.

Now encrypt the above string with the checksum method/ hash function.

*Supports only **MD5***

Formula for checksum generation

MD5 (MERCHANTCODE|PRN|AMOUNT|CHECKSUMKEY)

Example:

If MERCHANTCODE = *HACKATHON2017*, PRN = *123456*, AMOUNT = *1.00, CHECKSUMKEY* = *2tkEPvKwyU* than checksum would be calculated as below

md5(HACKATHON2017|123456|1.00|2tkEPvKwyU)

B. Checksum in response

Checksum is generated at PG end for data integrity checking at merchant's website/ application

For checksum generation, first PG generate a pipe separated string of certain response parameters

Now PG encrypt the above string with the checksum method/ hash function used in request checksum

Formula for checksum generation

MD5 (MERCHANTCODE|PRN|RPPTXNID|PAYMENT_AMOUNT |CHECKSUMKEY)

Example:

If MERCHANTCODE = HACKATHON2017, PRN = *123456*, RPPTXNID = *111111*, PAYMENTAMOUNT = *1.00, CHECKSUMKEY* = *2tkEPvKwyU* than checksum would be calculated as below

md5 (HACKATHON2017|*123456*|111111|1.00|*2tkEPvKwyU*)

It is highly recommended to validate the response checksum at merchant's end

Verification URL:

http://uat.rpp.rajasthan.gov.in/payments/v1/services/txnStatus

Verification Request:

To check the transaction status, merchant post only the 4 parameters named *"MERCHANTCODE", "PRN", "AMOUNT" and "APINAME"*.

Sample Html form request from RPP to merchant

```
<form action="< Verification URL >" method="POST">
    <input type="hidden" name="MERCHANTCODE" value="<MERCHANTCODE>" />
    <input type="hidden" name="PRN" value="<PRN>" />
    <input type="hidden" name="AMOUNT" value="<AMOUNT>" />
    <input type="hidden" name="APINAME" value="TXNSTATUS" />
</form>
```

Sample Request

POST /payments/v1/services/txnStatus HTTP/1.1
Host: PG_HOST_NAME
Content-Type: application/x-www-form-urlencoded

MERCHANTCODE=<*MerchantCode*>&**PRN**=<*PRN*>&**AMOUNT**=<*Amount*>&**APINAME**=*TXNSTATUS*

Verification Response

JSON Response

```
{
  "MERCHANTCODE": "<MERCHANTCODE>",
  "REQTIMESTAMP": "<REQTIMESTAMP>",
  "PRN": "<PRN>",
  "RPPTXNID": "<RPPTXNID>",
  "AMOUNT": "<AMOUNT>",
  "RPPTIMESTAMP": "<RPPTIMESTAMP>",
  "STATUS": "<STATUS>",
  "RESPONSECODE": "<RESPONSECODE>",
  "RESPONSEMESSAGE": "<RESPONSEMESSAGE>",
  "PAYMENTMODE": "<PAYMENTMODE>",
  "PAYMENTMODEBID": "<PAYMENTMODEBID>",
  "PAYMENTMODETIMESTAMP": "<PAYMENTMODETIMESTAMP>",
  "PAYMENTAMOUNT": "<PAYMENTAMOUNT>",
  "UDF1": "<UDF1>",
  "UDF2": "<UDF2>",
  "UDF3": "<UDF3>"
  "CHECKSUM": "<2e3071744e43c01f6c5c0d9d33d59308>"
}
```

CHECKSUM Calculation example code in Java:

```
public String md5(final String toBeEncryptString) throws Exception {
        if (toBeEncryptString == null) {
                throw new IllegalArgumentException("To be encrypt string must not be null");
        }
        try {
                MessageDigest md = MessageDigest.getInstance("MD5");
                md.update(toBeEncryptString.getBytes());
                return DatatypeConverter.printHexBinary(md.digest());
        } catch (Exception ex) {
                // Handle Exception
                throw new Exception("Checksum could not be generated due to an exception", ex);
        }
}
```

Annexure A: URL of Sites

Web Service Name	Web Service URL
HackathonEmitraFetch BillingData (Encrypted)	http://103.203.139.158/webServicesRepository/getFetchDetailsEncryptedBySso?
Parameter Name	encData=//wzeTciCLKFnevjkR+Y+9Av818ZKvBZLA4rU1FTCpA/2aoltEJROkqp79VPthK/hpYH7I9dZQAu0pNqjKPmn9IMUoazesKWqTNkvxnYv3I=

Annexure B: Encryption & Decryption Methods/Details

Encryption Method – AES encryption will be used with 128 bit key.
Encryption Key – Will be shared with Parameters

Environment	Encryption Method	Key
UAT/Staging/ Production	AES encryption will be used with 128 bit key.	Will be shared with Parameters

Annexure C: Various Parameter Values

Encryption Key	**EmitraNew@2016**
MID	
Service ID	

Annexure D: Encryption/ Decryption

- Encryption is the process of converting plain text data into a form, called a cipher text (encrypted text), that cannot be easily understood by unauthorized people.
- Decryption is the process of converting encrypted data back into its original form, so it can be understood.
- It is mandatory to implement encryption and exchange the information between Sites only in encrypted mode.
- e-MITRA follows Advanced Encryption Standard (AES) algorithm for Encryption and Decryption
- A symmetric key will be used for this Encryption and Decryption, which will be provided by e-MITRA.

Sample Encryption / Decryption code in Java.

AES Encryption Example code:

```
package encryption.impl;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;

import encryption.dec.Encryption;

public  class AESEncryption  extends EncryptionKey implements Encryption
{

	private static SecretKeySpec     secretKey;
	private static IvParameterSpec ivParameterSpec ;


	@Override
	public String encrypt(String encryptionSTR) {

		String encode = "";
		try {
			if (encryptionSTR != null) {

				//getKey(key);


				Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
				cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivParameterSpec);
				final String encryptedString =
Base64.encodeBase64String(cipher.doFinal(encryptionSTR.getBytes("UTF-8")));

				encode = encryptedString;
			}
		} catch (NoSuchPaddingException npe) {
			npe.printStackTrace();
		} catch (NoSuchAlgorithmException nae) {
			nae.printStackTrace();
		} catch (InvalidKeyException ike) {
			ike.printStackTrace();
		} catch (BadPaddingException bpe) {
			bpe.printStackTrace();
		} catch (IllegalBlockSizeException ibe) {
```

```
                ibe.printStackTrace();
        } catch (Exception e) {
                e.printStackTrace();
        }
        return encode;



}

@Override
public String decrypt(String decryptionSTR) {
        // TODO Auto-generated method stub



        try {
                if (decryptionSTR != null) {


                        //getKey(key);
                        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
                        cipher.init(Cipher.DECRYPT_MODE, secretKey, ivParameterSpec);


                        byte[] original = cipher.doFinal(Base64.decodeBase64(decryptionSTR));

                        return new String(original);

                }
        } catch (NoSuchPaddingException npe) {
                npe.printStackTrace();
        } catch (NoSuchAlgorithmException nae) {
                nae.printStackTrace();
        } catch (InvalidKeyException ike) {
                ike.printStackTrace();
        } catch (BadPaddingException bpe) {
                bpe.printStackTrace();
        } catch (IllegalBlockSizeException ibe) {
                ibe.printStackTrace();
        } catch (Exception e) {
                e.printStackTrace();
        }
        return null;

}

@Override
void getKey(String myKey) {

        byte[]   key;
        MessageDigest sha = null;
        try {
                key = myKey.getBytes("UTF-8");
                // System.out.println(key.length);
```

```
                        sha = MessageDigest.getInstance("SHA-256");
                        key = sha.digest(key);
                        key = Arrays.copyOf(key, 16); // use only first 128 bit
                        secretKey = new SecretKeySpec(key, "AES");
                        ivParameterSpec=new IvParameterSpec(key);
                        this.key=myKey;


                } catch (NoSuchAlgorithmException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (UnsupportedEncodingException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

        @Override
        public void setKey(String key) {
                getKey(key);
        }

}
```

Encryption Java Class:

```
package encryption.main;

//import java.net.URLDecoder;
//import java.net.URLEncoder;

import encryption.dec.Encryption;
import encryption.impl.encryptionHelper;

public class testClient {

        public static void main(String[] args) {
                Encryption AESencryption;

      try {
         AESencryption = encryptionFactory.getEncryptionByName("AES");
         String input ="<?xml version='1.0' encoding='utf-8'?><EmitraData PRN='16000000957'
AMT='1.25'/>";


        AESencryption.setKey("E-m!tr@2016");
        /* Parameters for posting refrence number  */


        System.out.println("AESencryption string :
"+AESencryption.encrypt("{\"SRVID\":\"1221\",\"searchKey\":1452627155\",\"SSOID\":\"SSOTESTKIOSK\"}"));
```

```
        /*for kiosk transaction verification*/
        System.out.println("AESDEcryption string :
"+AESencryption.decrypt("w2tZwKucTyOLM+cq+H1aRXckVuADGrE2GJb35btosuI3GNY3vViM/43tiBvinkRNy
o49DlVm3hr9FWe062eaKH+OE0UEMmnFL+1ci0f7P+VS0356eFk1WFwYoayFv+mG7Pmih4kCa7FekDZZFwFvd
A6VqEvq/SCcz7QgvWNr0/8udMJwUKeHIDHvbSvGdu/zeN2FTaipB43iuyUOAizQsH1s1lpx/aGXcmr4u6fPiyCF
aC540x6clNuYNoVktCPSPTNQ76KSg3CcqlUWyRJ3C/8+e6y3Z1ImJvXZ/Z7y4QWZZWJRZKrGnYta/6DvMmUJ
FXto14D1VSdt+IsWfiLpV2CCIa3mF9LeoT5pdF0zxOA++kNGD2w1QamVNfqWoolF/Fm25aCzImlXbJbHRFJlVN
ljCl9905NWoRo4c42Urg0l90pA9rPXy1H23z7TaT6Omx7szWz0ZguS+R/4jdrR9fPKJOFel1GFNxz2xsQuUgFqtz
0CKXL1vHHBXgSnaaj/ehvzvnl42T9TVpg93q0LG1ei/lXR6VK29UPmau3O6p0NB6kcg2lpvcS3N+WS+H3i30IYR
BkRhAQJsMkDA2tuitJolBBC8p1xSkp7OpFrs1Y077MUppJag8JpEPwG4jpk0CqK9iU8V+Shw0mrzsq4mqKRs8Jq
Fjd9MocKqkPfpkwmUTqPg8Ug3eaSSErVIBn4/BRsO5pOvEb21ySKpLG3YYji+8rP2GF1u43CUyfpQ8v7Zu5DuD
47FxTAcOoqGgvFvNbpIuFC53PH57htbxTRU4ruGplUKwpOAcvNXZrwe2blb7rKw3fNTBhkvOn+JfrkhQq9Dq6f
n3rUna5+2iWMp5XH0eH3kZZbIQq2kVqRjWN2f5TST+hAIkruvrHgf2C5a5c8xVDfR4C5JlngAEmsXp0zMdfQwt7
dcSWwQ+S0rxXn1sIJZ+YBN/m7k+k08nFEtaK+6l/j0W0sP6F245FePUkkx/vZOWRFr/TlquY7XvQBZbLmFsfszk
YDkWMnF0F5"));

      } catch (Exception e) {
         e.printStackTrace();
      }
         }
}
```

Example code for service call:

```
//package encryption.main;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.security.cert.X509Certificate;

import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

//import encryption.dec.Encryption;

public class TestWebService {


          private static final String POST_URL
="https://apitest.sewadwaar.rajasthan.gov.in/app/live/webServicesRepository/getFetchDetailsEncrypt
edBySso/WithEncryption?client_id=96f2ea05-8742-401d-aa42-ec269f8a71c0";

          private static final String POST_PARAMS
="encData=ELNJOXDH9zon/9qKclYzTLTLd3BHPSvI7tglugHzp7M8AkCLfGohln8L3h4nn1k+pPv93oX/FwCvQI
+QYNILvPYhfXhl2ZmiG9nj9GZGcAU=";
```

```
static StringBuffer data = new StringBuffer();

public static void main(String[] args) throws Exception {

        data.append(POST_PARAMS);
        // data.append(GET_PARAMS);
        sendPOST();
}

private static void sendPOST() throws IOException {

         byPassSSL();
        /*
         * MultivaluedMap formData = new MultivaluedMapImpl();
         * formData.add("P1", "1500005"); // formData.add("P2", "2218"); //
         * formData.add("P3", "1"); // String challan="1500002"; Client client =
         * Client.create(); WebResource resource = client.resource(
         * "http://jnvuemitra.azurewebsites.net/api/Products/GetProduct/");
         * ClientResponse responseY =
         * resource.post(ClientResponse.class,formData); String response =
         * responseY.getEntity(String.class);
         * System.out.println("Response in xml: "+response); String
         * TEST_XML_STRING =response;
         */
        // JSONObject xmlJSONObj = XML.toJSONObject(TEST_XML_STRING);
        // String jsonPrettyPrintString = xmlJSONObj.toString(4);
        URL obj = new URL(POST_URL);

        HttpsURLConnection con = (HttpsURLConnection) obj.openConnection();
        con.setRequestMethod("POST");

        // For POST only - START
        con.setDoOutput(true);
        OutputStream os = con.getOutputStream();
        os.write(data.toString().getBytes());
        os.flush();
        os.close();
        // For POST only - END

        int responseCode = con.getResponseCode();
        System.out.println("POST Response Code :: " + responseCode);

        if (responseCode == HttpURLConnection.HTTP_OK) { // success
                BufferedReader in = new BufferedReader(new InputStreamReader(
                                con.getInputStream()));
                String inputLine;
                StringBuffer response = new StringBuffer();

                while ((inputLine = in.readLine()) != null) {
                        response.append(inputLine);
                }
                in.close();

                // print result
```

```
                System.out.println("response of Emitra " + response.toString());
            } else {
                System.out.println("POST request not worked");
            }
        }

        public static void byPassSSL() {
            HttpURLConnection connection = null;
            StringBuffer responseReader = new StringBuffer();
            try {
                TrustManager[] trustAllCerts = new TrustManager[] { new X509TrustManager() {
                    public java.security.cert.X509Certificate[] getAcceptedIssuers() {
                        return null;
                    }

                    public void checkClientTrusted(X509Certificate[] certs,
                            String authType) {
                    }

                    public void checkServerTrusted(X509Certificate[] certs,
                            String authType) {
                    }
                } };
                SSLContext sc = SSLContext.getInstance("SSL");
                sc.init(null, trustAllCerts, new java.security.SecureRandom());
                HttpsURLConnection
                        .setDefaultSSLSocketFactory(sc.getSocketFactory());
                // Create all-trusting host name verifier
                HostnameVerifier allHostsValid = new HostnameVerifier() {
                    public boolean verify(String hostname, SSLSession session) {
                        return true;
                    }
                };
                // Install the all-trusting host verifier
                HttpsURLConnection.setDefaultHostnameVerifier(allHostsValid);
                /*
                 * end of the fix
                 */
                // logger.info("Response  from oxigen ---"+responseReader);
            } catch (Exception e) {
                // logger.error("Error in oxigen call---"+e.getMessage());
                e.printStackTrace();
            } finally {
                // connection.disconnect();
            }
        }

}
```

Sample Encryption / Decryption code in Dot NET.

```
//Rextester.Program.Main is the entry point for your code. Don't change it.
//Compiler version 4.0.30319.17929 for Microsoft (R) .NET Framework 4.5

using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Text;

namespace Rextester
{
   public class Program
   {
      public static void Main(string[] args)
      {

         Console.WriteLine( Rextester.Program.Encrypt("Test","EmitraNew@2016"));

          Console.WriteLine( Rextester.Program.Decrypt("pKXdPpYtjDbb9cpfthlWAA==","
EmitraNew@2016"));
      }



      public static string Encrypt(string textToEncrypt, string EncryptionPassword)
      {
         RijndaelManaged rijndaelCipher = new RijndaelManaged();
         rijndaelCipher.Mode = CipherMode.CBC;
         rijndaelCipher.Padding = PaddingMode.PKCS7;
         rijndaelCipher.KeySize = 256;
         rijndaelCipher.BlockSize = 128;
         byte[] pwdBytes = Encoding.UTF8.GetBytes(EncryptionPassword);
         pwdBytes = SHA256.Create().ComputeHash(pwdBytes);
         byte[] keyBytes = new byte[16];
         int len = pwdBytes.Length;
         if (len > keyBytes.Length)
         {
            len = keyBytes.Length;
         }
         Array.Copy(pwdBytes, keyBytes, len);
         rijndaelCipher.Key = keyBytes;
         rijndaelCipher.IV = keyBytes;
         ICryptoTransform transform = rijndaelCipher.CreateEncryptor();
         byte[] plainText = Encoding.UTF8.GetBytes(textToEncrypt);
         return Convert.ToBase64String(transform.TransformFinalBlock(plainText, 0,
plainText.Length));
      }

      public static string Decrypt(string textToDecrypt, string EncryptionPassword)
      {
         RijndaelManaged rijndaelCipher = new RijndaelManaged();
         rijndaelCipher.Mode = CipherMode.CBC;
         rijndaelCipher.Padding = PaddingMode.PKCS7;
         rijndaelCipher.KeySize = 256;
         rijndaelCipher.BlockSize = 128;
```

```
            byte[] encryptedData = Convert.FromBase64String(textToDecrypt);
            byte[] pwdBytes = Encoding.UTF8.GetBytes(EncryptionPassword);
            pwdBytes = SHA256.Create().ComputeHash(pwdBytes);
            byte[] keyBytes = new byte[16];
            int len = pwdBytes.Length;
            if (len > keyBytes.Length)
            {
               len = keyBytes.Length;
            }
            Array.Copy(pwdBytes, keyBytes, len);
            rijndaelCipher.Key = keyBytes;
            rijndaelCipher.IV = keyBytes;
            byte[] plainText = rijndaelCipher.CreateDecryptor().TransformFinalBlock(encryptedData, 0,
encryptedData.Length);
            return Encoding.UTF8.GetString(plainText);
         }
      }
}

public string CreateMD5(string input)
      {
         // Use input string to calculate MD5 hash
         using (System.Security.Cryptography.MD5 md5 = System.Security.Cryptography.MD5.Create())
         {
            byte[] inputBytes = System.Text.Encoding.ASCII.GetBytes(input);
            byte[] hashBytes = md5.ComputeHash(inputBytes);

            // Convert the byte array to hexadecimal string
            StringBuilder sb = new StringBuilder();
            for (int i = 0; i < hashBytes.Length; i++)
            {
               sb.Append(hashBytes[i].ToString("X2"));
            }
            return sb.ToString();
         }
      }

}
```