CVL KRA File Transfer API Documentation

Version 1.0
06 Dec 2024

Contents

1. INTRODUCTION

1.1 Purpose of the Document

CVL KRA offers an API method feature known as "File Transfer," designed for securely transferring files from client locations to our predefined path/folder via the JSON protocol and JWT Token Authentication.

1.2 API Overview

This guide provides specifications of the JSON-based REST API to validate requests against KYC Status, and to provide response with KYC Data.

This guide assumes that you are familiar with and experienced with the HTTP request using JSON payload and JWT Token Authentication.

1.3 Technical Pre-requisite required from client's end.

Make sure only TLS 1.2 protocol is enabled to connect API

Payload Encryption mechanism

IP Address: The IP needs to be whitelisted at our end to ensure security.
The request payload needs to be formatted as Json Data and then encrypted as below.

Encryption specification:

The Json payload shall be encrypted with AES IV mechanism.
Both API user and CVL will share the AES key for encryption-Decryption.

2. API DETAILS

2.1 Authentication API

2.1.1 Description

This API returns the Auth Token for the entity to perform transactions,
Client needs to request for JWT (JSON Web Token) by passing parameter "api_key" and "tokenvalidtime"(Optional) in the Header and Encrypted Username, Passcode and Password in the Body.

2.1.2 API Endpoint

UAT: https://krapancheck.cvlindia.com/V3/api/GetToken
LIVE: https://api.kracvl.com/int/api/GetToken

2.1.3 Input Headers

content-type, **user-agent** and **Tokenvalidtime** will come in request headers along with **apikey**.

2.1.4 Input Parameters

Name	Type (max. char. Limit)	Special characters allowed	Description	Mandatory (Y-Yes, N-No, O-Optional & Conditional)	Provided by CVL (Y/N)
content-type	String	/	Default value is: application/json	Y	N
user-agent	String		Default value is: CustomUsrAgnt	Y	Y
api_key	String		The API Key provided by CVL	Y	Y
tokenvalidtime	String		Token valid time can be specified by the user in Minutes format, The default validity duration of the auth token value will be for 1 day if	O	N

			not specified by the user.		
username	String (100)		The username provided by CVL	Y	Y
poscode	String (20)		The poscode provided by CVL	Y	Y
password	String (100)	~,!,@,#,$,%,^,&,*,(,),-,_,{,},?	The password provided by CVL	Y	Y

Request Packets

```
{
  "username":"UserName",
  "poscode":"PosCode",
  "password":"Password"
}
```

2.1.5 Output Parameters

Name	Type	Description
token	String (512)	This Token will be used by the API User for future requests
Validity	String (14)	The validity of the Token in yyyyMMddHHmmss format, the default validity will be 1 day.
Success	String (1)	The status of Token generation, 1 = Success else Failure
error_code	String (20)	Error while generating the token
error_message	String (512)	Description of error while generating the token

Response Packet: (Success)

```
{
        "success":"1",
        "token":"Generated Token string",
        "validity": " yyyyMMddHHmmss",
        "error_code": "",
        "error_message": ""
}
```

Response Packet: (Failure)

{
"success":"0",
"token":"",
"validity":"",
"error_code":"Error Code",
"error_message":"Error Message"
}

2.2 File Transfer Through API

2.2.1 Description

This API will facilitate the transfer of files from client locations to our designated path/folder.

2.2.2 API Endpoint

UAT: https://krapancheck.cvlindia.com/v3/api/FileTransfer
LIVE: https://api.kracvl.com/int/api/FileTransfer

2.2.3 Input Headers

content-type and user-agent will come in request headers along with token.

2.2.4 Input Parameters

Name	Type (max. char. Limit)	Special characters allowed	Description	Mandatory (Y-Yes, N-No, O-Optional & C-Conditional)	Provided by CVL (Y/N)
content-type	String	/	Default value is: application/json	Y	N
user-agent	String		Default value is: CustomUsrAgnt	Y	Y
Token	String	: and -	The Token generated with the GetToken method	Y	Y
pan	String		PAN	Y	N
poscode	String		The poscode provided by CVL	Y	Y
pdf_filename	String		File name with extension	O	N
pdf_base64	String		File bites converted in Base 64	O	N
pdf_password_exempt	Character		If PDF is password protected pass 'N'	O	N
pdf_password	String		If pdf_password_exempt is 'N',pdf_password must be provided.	O	N
xml_filename	String		File name with extension	O	N
xml_base64	String		File bites converted in Base 64	O	N
image_filename	String		File name with extension	O	N
image_base64	String		File bites converted in Base 64	O	N
zip_filename	String		File name with extension	O	N
zip_base64	String		File bites converted in Base 64	O	N
zip_password_exempt	Character		If ZIP is password protected pass 'N'	O	N
zip_password	String		If zip_password_exempt is 'N',zip_password must be provided.	O	N

Request Header:
Token: Generated Token string

Sample Request (To be encrypted by the client):


```
{
  "pan": "ENTER VALID PAN",
  "poscode": "ENTER VALID POSCODE",
  "pdf_filename": "sample_pdf.pdf",
  "pdf_base64": "ENTER VALID BASE64",
  "pdf_password_exempt": "N",
  "pdf_password": "ENTER PASSWORD",
  "xml_filename": "sample_xml.xml",
  "xml_base64": "ENTER VALID BASE64",
  "image_filename": "sample_image.jpg",
  "image_base64": "ENTER VALID BASE64",
  "zip_filename": "sample_zip.zip",
  "zip_base64": "ENTER VALID BASE64",
  "zip_password_exempt": "",
  "zip_password": ""
}
```


Encrypted Payload:

BixrwiNjoGV8VBzjvago-w:HwxmZM1PXxX4npASREDD_ygUd1OsfEeWuJADdB4sKfltMZC380bqqmL_GzBECJ4MHAxksF9P0T2pjLly9dl4WGzdPzb3ClydV5MGY9ucFpSgkRBceY9C_4mLlOQheZBFtxqrxCAFXpol9IOre7cT0h-hTQrO6NfW4BthHHdL0wBj6bC8Hjp-vh044okc45qMIMjrCorXxbJbnxoARJxupUsdXVd8Zj_9m8D25j6dnIXE5RpkrcryXjbSb9w4qBZ7g_1C3ElTZQv9fOkIUmjaEBnsCFZDt8ZmvLZzJB3WeqQ-W_xISFurN25XUjKm1Xq4aZ1QAnX5DaPUQEN7Xj-g2D2EVHIPH1F-qSkp_ctS7hByMyxImmH_as49fgWuTaKZkl1yM7u9YTbz-jt8yggCWc5O0WzU4QtgJDDgfmkj704VEe43S5FmWlvfd92Ud7xYQYpx6jF9zfdDXYG6Xy0lUYq5CmONQcdFtAFEB21eo3cTRn7kYqFd3Zv2GXWakMMKA1rqcF43w62r6EfWNKP595pQ7HmVAH1m_4FPGBEZLrkMUh7u3HqXJ0i2PkdpZHEJXN9REsZCvXARx5wJR3LqRIXMBHELbkznPEtgAh01vwJvFwEPBsLHNaDUfQw-1EpV14vAxi30tJcxkb50VtLwdc694MyUynFjnb-V_YSDaY

Upon requesting the above scenario, the API will respond with below displayed.

2.2.5 Output Parameters

Sample Response:

{

"resdtls":"pdf:SUCCESS,xml:SUCCESS,img:SUCCESS;zip:SUCCESS",

"error_code":"",

"error_message":""

}

Sample Encrypted Response:

sfDwJeZ5FYGT7xaFuDtzug:kO6eEi9F7JLBBwHIKkOEg72VYH1_JzptayrhdVgChaI9X5Rvb3T7JI7O
0SOFWtRGZyJKtwDKgz-DSFzOqtyFtTHoFTkbgPXWDDsF-fxAN3dZceBZ-
Z8FSTRgI2J5x5w1598olSxBqY0yQyIFsYVXtQ

3. ENCRYPTION-DECRYPTION PROCESS

Overview

Using AES (Advanced Encryption Standard) encryption and decryption using the CBC (Cipher Block Chaining) mode with PKCS5Padding.

Private Key = Randomly generated string, provided by CVL.

Steps for Encryption:

1. Generate a random Initialization Vector (IV) for AES encryption.
2. Encrypt the plaintext string using AES/CBC/PKCS5 Padding mode encryption with the specified Private Key and IV.
3. Send IV as a part of encrypted Data by creating a string with IV and encrypted data separated by a colon ":".

Steps for Decryption:
1) Split response string by colon ":"

2) Get the IV- from the first string and the second string is encrypted response.

2) Decrypt encrypted data using IV and Client's private key.

4. API Error code Reference

Code	Description
WEBERR-001	Invalid User ID / PosCode / Password / Access Privilege Not Set
WEBERR-002	Invalid XML
WEBERR-005	UNKNOWN ERROR
WEBERR-006	Invalid Record Count
WEBERR-007	Invalid Batch Size
WEBERR-008	Invalid / Expired Batch No.
WEBERR-009	Invalid RTA Code (APP_RTA_CODE) provided
WEBERR-010	Invalid Intermediary Code (APP_POS_CODE) provided
WEBERR-011	Invalid KRA Code (APP_KRA_CODE) provided
WEBERR-012	Invalid DOB (APP_DOB_INCORP) PROVIDED
WEBERR-013	You are not allowed to fetch the details
WEBERR-014	Invalid OTHKRA Code (APP_OTHKRA_CODE) provided
WEBERR-015	Access Limit exceeded
WEBERR-016	Invalid header values
WEBERR-017	The Content-Type in the header not specified
WEBERR-018	Unsupported media type
WEBERR-019	Token not provided
WEBERR-020	Invalid Token
WEBERR-021	Encryption parameter not specified
WEBERR-022	Encrypted request data not provided
WEBERR-023	Invalid encrypted data
WEBERR-024	Invalid request details
WEBERR-025	API Key not provided
WEBERR-026	Invalid API Key
WEBERR-027	Invalid PAN format
WEBERR-028	Duplicate request
WEBERR-029	Invalid IP address
WEBERR-030	At least one of pdf_Base64 or xml_Base64 or image_Base64 or zip_Base64 must be provided.
WEBERR-031	If base64 is provided, filename with proper extension must also be provided.
WEBERR-032	please provide the password.
WEBERR-033	File is too large. Maximum size is 10 MB.
WEBERR-034	File extension does not match the Base64 decoded data or BASE64./ Invalid file format.

5. C# Code for Encrypting and Decrypting Data

```
public string EncryptString(string aesKey, string data, out string IV)
{
    var plainText = data.Trim();
    byte[] Key = Base64UrlEncoder.DecodeBytes(aesKey);
    byte[] encrypted;

    using (Aes aesAlg = Aes.Create())
    {
        aesAlg.Key = Key;
        IV = Base64UrlEncoder.Encode(aesAlg.IV);
        ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);

        using (MemoryStream msEncrypt = new MemoryStream())
        {
            using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor,
CryptoStreamMode.Write))
            {
                using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
                {
                    swEncrypt.Write(plainText);
                }

                encrypted = msEncrypt.ToArray();
                return Base64UrlEncoder.Encode(encrypted);
            }
        }
    }
}

public string DecryptString(string aesKey, string encryptedText, string IV)
{
    string plaintext = null;

    try
```

```
    {
        var cipherText = Base64UrlEncoder.DecodeBytes(encryptedText.Trim());
        byte[] Key = Base64UrlEncoder.DecodeBytes(aesKey);
        var iv = Base64UrlEncoder.DecodeBytes(IV.Trim());

        using (Aes aesAlg = Aes.Create())
        {
            aesAlg.Key = Key;
            aesAlg.IV = iv;
            ICryptoTransform decryptor = aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);

            using (MemoryStream msDecrypt = new MemoryStream(cipherText))
            {
                using (CryptoStream csDecrypt = new CryptoStream(msDecrypt, decryptor,
CryptoStreamMode.Read))
                {
                    using (StreamReader srDecrypt = new StreamReader(csDecrypt))
                    {
                        plaintext = srDecrypt.ReadToEnd();
                    }
                }
            }
        }

    }
    catch (Exception ex)
    {
        plaintext = "-1";

    }
    return plaintext;
}
```

6. Java Code for Encrypting and Decrypting Data

For Encryption

```
import java.security.SecureRandom;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class EncryptionUtils {

     public static void main(String args[]) {
        try {
            String aesKey = "AES Key here";
            byte[] ivForEncryption = generateIV();
            String ivForPayload =
Base64.getUrlEncoder().encodeToString(ivForEncryption);

            String data = "{'username':UserName here,'poscode':'Pos Code
here','password':'Password here'}";
            String encryptedData = encryptString(aesKey, data,
ivForEncryption);
            String encryptedRequestData = ivForPayload + ":" + encryptedData;

            System.out.println("Encrypted Request Data: " +
encryptedRequestData);
        } catch (Exception e) {

            e.printStackTrace();

        }

    }


    public static byte[] generateIV() {

        byte[] ivForEncryption = null;
```

```
        try {
            SecureRandom secureRandom = new SecureRandom();
            byte[] iv = new byte[16];
            secureRandom.nextBytes(iv);
            ivForEncryption = iv;
        } catch (Exception e) {
            e.printStackTrace();
        }

        return ivForEncryption;
    }


    public static String encryptString(String aesKey, String data, byte[] iv)
{
        String encryptedText = null;

        try {
            byte[] keyBytes = Base64.getDecoder().decode(aesKey);
            SecretKeySpec secretKey = new SecretKeySpec(keyBytes, "AES");

            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey, new
IvParameterSpec(iv));

            byte[] encryptedBytes = cipher.doFinal(data.getBytes("UTF-8"));
            encryptedText =
Base64.getUrlEncoder().encodeToString(encryptedBytes);
        } catch (Exception e) {
            e.printStackTrace();
        }


        return encryptedText;

    }

}
```

For Decryption

```
import java.security.SecureRandom;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class DecryptionUtils {

     public static void main(String args[]) {
        try {
            String aesKey = "AES Key here";
            String EncryptedResponseData =
"GsMbupa1dlWU22o0qQmIVH:v3vMlZTq_5wtU5gSd1_8OnW-
ra3UKI_mrWjjxuXzx9LaRqmiGYhv89Ap1N1ObsL7j4ZQKGZsgObU9i5R1Uve_jPy_JbYBMjFDPXQQ_
dbHv7DSttuGid-ctjX5aR_xL1IkUN3cgqPVkX6MG6pn_jovDM60wpfaDcx0a10jg2AthoCuIL33u6-
Ap8lcxwN1UnSWNwNgcP4-
D5GIbhU6JQPAq2EcW9XoNyWKp9cLIYiUZBMocx6vHIJJ9a4N52toOOa_PJMnzXjDcLSV2eW5qc8PtB
G4S4ybxJrQSUgPaU52q99EfI4PA_yLMtrJwKCFcIE2tkBbgcpYdAnt6nNGo-
2_OYOwJNYrycnySz0mVMFDCTlCr7rvc6Q_BOsjaS27GA4kt_3LJxtjOJA70HBxHvDkdloV0i5sBu8S
CSsvIMQ4c0k3YAD0R7-FSKFB2fYK8IVGPnvtNiultEmdxNCCvKUBpn7LvkiGvAXjRe8IVGUOLk";
            String[] splittedStrings = EncryptedResponseData.split(":");
            String iv = splittedStrings[0];
            String encryptedText = splittedStrings[1];
            String DecryptedResponseData = decryptString(aesKey,
encryptedText, iv);
            System.out.println("Decrypted Response Data: " +
DecryptedResponseData);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    public static String decryptString(String aesKey, String encryptedText,
String iv) {
        String decryptedText = null;
        try {
            byte[] keyBytes = Base64.getDecoder().decode(aesKey);
            byte[] ivBytes = Base64.getUrlDecoder().decode(iv);
```

```
            SecretKeySpec secretKey = new SecretKeySpec(keyBytes, "AES");
            Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
            cipher.init(Cipher.DECRYPT_MODE, secretKey, new
IvParameterSpec(ivBytes));
            byte[] encryptedBytes =
Base64.getUrlDecoder().decode(encryptedText);
            byte[] decryptedBytes = cipher.doFinal(encryptedBytes);
            decryptedText = new String(decryptedBytes, "UTF-8");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return decryptedText;
    }
}
```

7. NodeJS Code for Encrypting and Decrypting Data

For Encryption

```
const crypto = require('crypto');

function generateIV() {
  return crypto.randomBytes(16);
}
function encryptString(aesKey, data, iv) {
  const keyBytes = Buffer.from(aesKey, 'base64');
  const cipher = crypto.createCipheriv('aes-256-cbc', keyBytes, iv);
  let encryptedData = cipher.update(data, 'utf-8', 'base64');
  encryptedData += cipher.final('base64');
  return encryptedData;
}
function main() {
  try {
    const aesKey = "AES KEY here";
    const ivForEncryption = generateIV();
    const ivForPayload = ivForEncryption.toString('base64');

    const data = "{'username':'User Name','poscode':'Pos Code','password':'Password'}";
    const encryptedData = encryptString(aesKey, data, ivForEncryption);
    const encryptedRequestData = `${ivForPayload}:${encryptedData}`;

    console.log("Encrypted Request Data: " + encryptedRequestData);
  } catch (e) {
    console.error(e);
  }
}

main();
```

For Decryption

```
const crypto = require('crypto');
function decryptString(aesKey, encryptedText, iv) {
  let decryptedText = null;
  try {
    const keyBytes = Buffer.from(aesKey, 'base64');
    const ivBytes = Buffer.from(iv, 'base64');
    const decipher = crypto.createDecipheriv('aes-256-cbc', keyBytes, ivBytes);
    let decryptedData = decipher.update(encryptedText, 'base64', 'utf-8');
    decryptedData += decipher.final('utf-8');
    decryptedText = decryptedData;
  } catch (e) {
    console.error(e);
  }
  return decryptedText;
}
function main() {
  try {
    const aesKey = "AES KEY here";
    const EncryptedResponseData = "ENCRYPTED DATA here";
    const splittedStrings = EncryptedResponseData.split(":");
    const iv = splittedStrings[0];
    const encryptedText = splittedStrings[1];
    const DecryptedResponseData = decryptString(aesKey, encryptedText, iv);
    console.log("Decrypted Response Data: " + DecryptedResponseData);
  } catch (e) {
    console.error(e);  }}main();
```

8. PYTHON Code for Encrypting and Decrypting Data

For Encryption

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import base64
import os

def base64UrlEncode(data):
    return base64.urlsafe_b64encode(data).decode().rstrip("=")

def base64UrlDecode(data):
    padded_data = data + "=" * (4 - len(data) % 4)
    return base64.urlsafe_b64decode(padded_data)

def encrypt_string(aes_key, data):
    iv = os.urandom(16)
    key = base64UrlDecode(aes_key.strip())
    cipher = AES.new(key, AES.MODE_CBC, iv)
    padded_data = pad(data.encode('utf-8'), AES.block_size)
    encrypted_text = cipher.encrypt(padded_data)
    iv_encoded = base64UrlEncode(iv)
    encrypted_text_encoded = base64UrlEncode(encrypted_text)
    return iv_encoded + ":" + encrypted_text_encoded
```

For Decryption

```
def decrypt_string(aes_key, encrypted_text, iv):
    try:
        key = base64UrlDecode(aes_key.strip())
        iv_bytes = base64UrlDecode(iv.strip())
        cipher_text = base64UrlDecode(encrypted_text.strip())
        cipher = AES.new(key, AES.MODE_CBC, iv_bytes)
        decrypted_data = cipher.decrypt(cipher_text)
        plaintext = unpad(decrypted_data, AES.block_size).decode('utf-8',
errors='ignore')
    except Exception as ex:
        plaintext = "-1"
    return plaintext



aes_key = "AES KEY HERE"
data = "ENCRYPTED DATA HERE"
encrypted_message = encrypt_string(aes_key, data)
print("Encrypted Message:", encrypted_message)

i = encrypted_message.split(":")
iv = i[0]
encrypted_text = i[1]
decrypted_text = decrypt_string(aes_key, encrypted_text, iv)
print("Decrypted Text:", decrypted_text)
```