LG CNS-CNSPay(Non-PG)

암복호화 샘플 소스

Ver. 1.2

본 문서와 정보는 LG CNS 재산입니다. 또한, 모든 정보는 LG CNS 소유 정보이며, LG CNS 의 사전 동의 없이 본 문서의 어떤 정보도 열람, 복사, 유용, 또는 타인과 공유되어서는 안됩니다.

본 문서의 정보는 변경될 수 있으며 변경 시, 본 문서는 수정될 것입니다. 본 문서의 내용에 관한 어떠한 의견이라도 귀하의 프로젝트 또는 사업 담당자에게 전달하여 주십시오.

개정이력

버전	변경일	변경 내용	작성자	승인자
1.0	2022.04.13	최초작성	JSW	
1.1	2022.04.14	Import 소스 추가	JSW	
1.2	2022.04.18	Base64Coder 설명 추가	JSW	

목 차

1. Java

1.1 Hash 생성 샘플 소스

```java
import java.security.MessageDigest;

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

public String SHA256Salt(String str, String salt) { // salt : 제공된 MerchantKey
    String passACL = null;

    MessageDigest md = null;

    try {
        md = MessageDigest.getInstance("SHA-256");

        md.reset();
        md.update(salt.getBytes());
        byte[] raw = md.digest(str.getBytes());

        //Hardening against the attacker's attack
        md.reset();
        raw = md.digest(raw);

        byte[] encodedBytes = Base64.encodeBase64(encodeHex(raw).getBytes());
        passACL = new String(encodedBytes);

    } catch (Exception e) {
        LOGGER.error("[SHA256Salt] Exception : {}", e.getMessage());
        passACL = "";
    }

    return passACL;
}

public String encodeHex(byte[] b) {
    char[] c = Hex.encodeHex(b);
    return new String(c);
}
```

1.2 Encrypt 및 Decrypt 샘플 소스

```java
// 공통 소스
import java.security.InvalidKeyException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

// CNS 암복호화 Util (1.3 기타 내용 참고)
import com.lgcns.common.utils.Base64Coder;
import com.lgcns.common.utils.ByteUtils;
```

```
public class CipherCBC {
    private String cipherPass = "";
    private String cipherIv = "";

    private String sk = "";
    private Key key = null;
    private Cipher cipher = null;

    private String transformation = "AES/CBC/PKCS5Padding";

    public CipherCBC(String password, String iv) throws InvalidKeyException, NumberFormatException,
NoSuchAlgorithmException, InvalidKeySpecException, IllegalArgumentException, NoSuchPaddingException {
        cipherPass=password;
        cipherIv = iv;
        sk = ByteUtils.toHexString(this.cipherPass.getBytes());
        key = generateKey("AES", ByteUtils.toBytes(sk, 16));
        cipher = Cipher.getInstance(transformation);
    }

    public String encrypt(String encryptStr)throws Exception{

        cipher.init(1, key, new IvParameterSpec(cipherIv.getBytes()));

        byte[] plain = encryptStr.getBytes("UTF-8");
        byte[] encrypt = cipher.doFinal(plain);

        return Base64Coder.encodeLines(encrypt);
    }

    public String decrypt(String decryptStr) throws Exception{

        cipher.init(2, key, new IvParameterSpec(cipherIv.getBytes()));

        byte[] decryptByte =  Base64Coder.decode(decryptStr);
        byte[] decrypt = cipher.doFinal(decryptByte);
        String plainText = new String(decrypt,"UTF-8");

        return plainText;
    }

    public Key generateKey(String algorithm) throws NoSuchAlgorithmException{
        KeyGenerator keyGenerator = KeyGenerator.getInstance(algorithm);
        SecretKey keys = keyGenerator.generateKey();
        return keys;
    }

    public Key generateKey(String algorithm, byte[] keyData) throws NoSuchAlgorithmException{
        return new SecretKeySpec(keyData, algorithm);
    }

}

// Encrypt
// encKey : 제공된 MerchantKey(암호화 시 앞 16byte만 사용)
private String encryptAESCBCByKey(String encKey, String decStr) throws Exception {

    String encStr = "";

    String masterKey = encKey;
    String cbcIV = encKey.substring(0, 16);
```

```
            try {
                CipherCBC cipherCBC = new CipherCBC(masterKey, cbcIV);
                encStr = cipherCBC.encrypt(decStr);
            } catch (InvalidKeyException e) {
                LOGGER.error("[encryptAESCBCByKey] InvalidKeyException : {}", e.getMessage(), e);
            }
            return encStr;
          }

        // Decrypt
        // encKey : 제공된 MerchantKey(복호화 시 앞 16byte만 사용)
        public String decryptAESCBCByKey(String encKey, String encStr) throws Exception {

            String decStr = "";

            String masterKey = encKey;
            String cbcIV = encKey.substring(0, 16);

            try {
                CipherCBC cipherCBC = new CipherCBC(masterKey, cbcIV);
                decStr = cipherCBC.decrypt(encStr);
            } catch (InvalidKeyException e) {
                LOGGER.error("[decryptAESCBCByKey] InvalidKeyException : {}", e.getMessage(), e);
            }

            return decStr;
        }
```

1.3 기타

- CNS 암복호화 Util

  - Base64Coder
    RFC 1521에 정의된 내용을 기준으로 작성된 Base64Coder 클래스입니다. 아래의 사이트에서 제공되는 소스파일을 다운받으셔서 사용하시면 됩니다.

    Project home page: https://www.source-code.biz/base64coder/java/
    Source code: Base64Coder.java

  - ByteUtils

```
/**
 * <p>unsigned byte(바이트) 배열을 16진수 문자열로 바꾼다.</p>
 *
 * <pre>
 * ByteUtils.toHexString(null)                        = null
 * ByteUtils.toHexString([(byte)1, (byte)255])       = "01ff"
 * </pre>
 *
 * @param bytes unsigned byte's array
 * @return
 * @see HexUtils.toString(byte[])
 */
public static String toHexString(byte[] bytes) {
            if (bytes == null) {
                        return null;
            }

            StringBuffer result = new StringBuffer();
            for (byte b : bytes) {
                result.append(Integer.toString((b & 0xF0) >> 4, 16));
                result.append(Integer.toString(b & 0x0F, 16));
        }
        return result.toString();
}

/**
 * <p>8, 10, 16진수 문자열을 바이트 배열로 변환한다.</p>
 * <p>8, 10진수인 경우는 문자열의 3자리가, 16진수인 경우는 2자리가, 하나의 byte로 바뀐다.</p>
 *
 * <pre>
 * ByteUtils.toBytes(null)     = null
 * ByteUtils.toBytes("0E1F4E", 16) = [0x0e, 0xf4, 0x4e]
 * ByteUtils.toBytes("48414e", 16) = [0x48, 0x41, 0x4e]
 * </pre>
 *
 * @param digits 문자열
 * @param radix 진수(8, 10, 16만 가능)
 * @return
 * @throws NumberFormatException
 */
public static byte[] toBytes(String digits, int radix) throws IllegalArgumentException, NumberFormatException {
        if (digits == null) {
                return null;
        }
        if (radix != 16 && radix != 10 && radix != 8) {
                throw new IllegalArgumentException("For input radix: ₩"" + radix + "₩"");
        }
        int divLen = (radix == 16) ? 2 : 3;
int length = digits.length();
if (length % divLen == 1) {
        throw new IllegalArgumentException("For input string: ₩"" + digits + "₩"");
}
length = length / divLen;
byte[] bytes = new byte[length];
for (int i = 0; i < length; i++) {
        int index = i * divLen;
        bytes[i] = (byte)(Short.parseShort(digits.substring(index, index+divLen), radix));
}
return bytes;
}
```

2. .NET

2.1 Hash 생성 샘플 소스

```
// ASP.NET 기준
using System;
using System.Security.Cryptography;

static public String SHA256salt(String val, String salt){
    StringBuilder sb = new StringBuilder();
    String passACL = "";

    SHA256 sha = new SHA256Managed();

    Byte[] hash = sha.ComputeHash(Encoding.UTF8.GetBytes(salt + val));
    hash = sha.ComputeHash(hash);

    foreach (byte b in hash){
        sb.Append(b.ToString("x2"));
    }
    passACL = sb.ToString();

    byte[] raw = Encoding.UTF8.GetBytes(passACL);
    passACL = Convert.ToBase64String(raw);

    return passACL;
}
```

2.2 Encrypt 및 Decrypt 샘플 소스

```
// ASP.NET 기준
public class CipherCBC{
        public CipherCBC(){
        }

        private RijndaelManaged GenerateAES(String key){
            RijndaelManaged aes = new RijndaelManaged();
            //aes.KeySize = 256;
            //aes.BlockSize = 128;
            aes.Mode = CipherMode.CBC;
            aes.Padding = PaddingMode.PKCS7;
            aes.Key = Encoding.UTF8.GetBytes(key);
            aes.IV = Encoding.UTF8.GetBytes(key.Substring(0, 16));
            return aes;
        }

        //AES_256 encrypt
        // key : 제공된 MerchantKey(암호화 시 앞 16byte만 사용)
        public String ParameterEncrypt(String key, String plainText){
            String Output=string.Empty;
            using (RijndaelManaged aes = GenerateAES(key)){
                ICryptoTransform encrypt = aes.CreateEncryptor(aes.Key, aes.IV);
                byte[] xBuff = null;
                using (MemoryStream ms = new MemoryStream()){
                    using (CryptoStream cs = new CryptoStream(ms, encrypt, CryptoStreamMode.Write)){
                        byte[] xXml = Encoding.UTF8.GetBytes(plainText);
                        cs.Write(xXml, 0, xXml.Length);
            }

            xBuff = ms.ToArray();
        }
        Output = Convert.ToBase64String(xBuff);
    }
    return Output;
}

//AES_256 decrypt
// key : 제공된 MerchantKey(복호화 시 앞 16byte만 사용)
public String ParameterDecrypt(String key, String encryptText){
    String Output = string.Empty;
    using (RijndaelManaged aes = GenerateAES(key)){

        ICryptoTransform decrypt = aes.CreateDecryptor();
        byte[] xBuff = null;
        using (MemoryStream ms = new MemoryStream()){
            using (CryptoStream cs = new CryptoStream(ms, decrypt, CryptoStreamMode.Write)){
                byte[] xXml = Convert.FromBase64String(encryptText);
                cs.Write(xXml, 0, xXml.Length);
            }

            xBuff = ms.ToArray();
        }
        Output = Encoding.UTF8.GetString(xBuff);
    }
    return Output;
}
```

3. Python

3.1 Hash 생성 샘플 소스

```
import hashlib, base64

    def SHA256Salt(self, plainText, salt):

        first_hash = hashlib.sha256(salt+plainText).digest()
        hash_object = hashlib.sha256(first_hash)
        hex_dig = hash_object.hexdigest()
        encoded_text = base64.b64encode(hex_dig)
```

3.2 Encrypt 및 Decrypt 샘플 소스

```
import base64
from Crypto.Cipher import AES

BLOCK_SIZE = 16

# PKCS5 Padding
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
unpad = lambda s : s[0:-ord(s[-1])]

class CipherAesCbc :

    def __init__(self):
        { }

    def encrypt(self, key, plaintext):
        #{
            iv = key[:16]

            cipher = AES.new(key, AES.MODE_CBC, iv)

            #self.mlogger.info('plain text : ' + plaintext)
            ctext = pad(plaintext)
            ctext = cipher.encrypt( ctext )
            encoded_text = base64.encodestring(ctext)

            encoded_text = encoded_text.replace("₩n", "")
            #encoded_text = encoded_text.strip('₩n')

            #self.mlogger.info('cipher text : ' + encoded_text)

            return encoded_text
        #}

    def decrypt(self, key, cipherText) :
        #{
            try :
                iv = key[:16]

                cipher = AES.new(key, AES.MODE_CBC, iv)

                #self.mlogger.info('cipher text : ' + cipherText)
                decoded_text = base64.decodestring(cipherText)
            ptext = cipher.decrypt( decoded_text )
            ptext = unpad(ptext)

            #self.mlogger.info('plain text : ' + ptext)

            return ptext

        except :
            #self.mlogger.info('Error on decryption')
            ptext = ""

        return ptext
    #}

// key : 제공된 MerchantKey(암호화 시 앞 16byte만 사용)
def parameterEncrypt(self, key, plainText):
    #{
        if key == None or plainText == None or key == "" or plainText == "" or len(key) < 16 :
            return ""
        else :

            cipher = CipherAesCbc()
            #cipher.setLogger(self.mlogger)
            encryptText = cipher.encrypt(key, plainText)

            return encryptText
    #}

// key : 제공된 MerchantKey(복호화 시 앞 16byte만 사용)
def parameterDecrypt(self, key, cipherText):
    #{
        if key == None or cipherText == None or key == "" or cipherText == "" or len(key) < 16 :
            return ""
        else :
            cipher = CipherAesCbc()
            #cipher.setLogger(self.mlogger)
            decryptText = cipher.decrypt(key, cipherText)

            return decryptText
    #}
```

4. PHP

4.1 Hash 생성 샘플 소스

```
<?php
class kmpayFunc {

    public function parameterHash($key, $plainText) {
        try {
            $hashText = "";
            $mth = "sha256";
            if ($key == null || $plainText == null || $key == "" || $plainText == "" || strlen($key) < 16) {
                return "";
            } else {
                $hashText= base64_encode(hash($mth, hash($mth, $key.$plainText, true),
false));
            }
            return $hashText;
        } catch (Exception $ex) {
            return "_FAIL_";
        }
    }
}
$kmFunc = new kmpayFunc();
$payHash = $kmFunc->parameterHash("salt키", "text값");
echo $payHash . "₩n";
?>
```

4.2 Encrypt 및 Decrypt 샘플 소스

```
<?php

class kmpayFunc {
    // key : 제공된 MerchantKey(암호화 시 앞 16byte만 사용)
    public function parameterEncrypt($key, $plainText) {
        try {
            $encryptText = "";
            $iv = "";
            if ($key == null || $plainText == null || $key == "" || $plainText == "" || strlen($key) < 16) {
                return "";
            } else {
                $iv = substr($key, 0, 16);
                $encryptText = $this->AESCBCPKCS5($plainText, $key, $iv, "enc", "yes");
            }
            return $encryptText;
        } catch (Exception $ex) {
            $this->writeLog("parameterEncrypt() Exception Code ".$ex->getCode()." : ".$ex-
>getMessage()." in ".$ex->getFile()." on line ".$ex->getLine());
            return "_FAIL_";
        }
    }
    // key : 제공된 MerchantKey(복호화 시 앞 16byte만 사용)
    public function parameterDecrypt($key, $EncryptText) {
        try {
            $decryptText = "";
            $iv = "";
            if ($key == null || $EncryptText == null || $key == "" || $EncryptText == "" || strlen($key) <
16) {
                return "1";
            } else {
                $iv = substr($key, 0, 16);
                $decryptText = $this->AESCBCPKCS5($EncryptText, $key, $iv, "dec", "yes");
            }
            return $decryptText;
        } catch (Exception $ex) {
            $this->writeLog("parameterDecrypt() Exception Code ".$ex->getCode()." : ".$ex-
>getMessage()." in ".$ex->getFile()." on line ".$ex->getLine());
            return "_FAIL_";
        }
    }
    public function PKCS5Pad($text, $blocksize = 16) {
        try {
            $pad = $blocksize - (strlen($text) % $blocksize);
            return $text.str_repeat(chr($pad), $pad);
        } catch (Exception $ex) {
            $this->writeLog("PKCS5Pad() Exception Code ".$ex->getCode()." : ".$ex->getMessage()."
in ".$ex->getFile()." on line ".$ex->getLine());
            return "_FAIL_";
        }
    }
    public function PKCS5UnPad($text) {
        try {
            $pad = ord($text{strlen($text)-1});
            if ($pad > strlen($text)) return $text;
            if (!strspn($text, chr($pad), strlen($text) - $pad)) return $text;
            return substr($text, 0, -1 * $pad);
        } catch (Exception $ex) {
            $this->writeLog("PKCS5UnPad() Exception Code ".$ex->getCode()." : ".$ex-
>getMessage()." in ".$ex->getFile()." on line ".$ex->getLine());
            return "_FAIL_";
        }
    }
    public function encrypt($iv, $key, $str) {
        try {
            $td = mcrypt_module_open("rijndael-128", "", "cbc", "");
            @mcrypt_generic_init($td, $key, $iv);
            $encrypted = @mcrypt_generic($td, $this->PKCS5Pad($str));
            mcrypt_generic_deinit($td);
            mcrypt_module_close($td);
            return $encrypted;
        } catch (Exception $ex) {
            $this->writeLog("encrypt() Exception Code ".$ex->getCode()." : ".$ex->getMessage()." in
".$ex->getFile()." on line ".$ex->getLine());
            return "_FAIL_";
        }
    }
    public function decrypt($iv, $key, $code) {
        try {
            $td = mcrypt_module_open("rijndael-128", "", "cbc", "");
            @mcrypt_generic_init($td, $key, $iv);
            $decrypted = @mdecrypt_generic($td, $code);
            mcrypt_generic_deinit($td);
            mcrypt_module_close($td);
            return $this->PKCS5UnPad($decrypted);
        } catch (Exception $ex) {
            $this->writeLog("decrypt() Exception Code ".$ex->getCode()." : ".$ex->getMessage()." in
".$ex->getFile()." on line ".$ex->getLine());
            return "_FAIL_";
            }
    }
    public function AESCBCPKCS5($source_data, $key, $iv, $mode="enc", $base64="yes") {
            try {
                    if ($mode == "dec") {
                            if ($base64 == "yes") return $this->decrypt($iv, $key,
base64_decode($source_data));
                            else return $this->decrypt($iv, $key, $source_data);
                    }
                    else {
                            if ($base64 == "yes") return base64_encode($this->encrypt($iv, $key,
$source_data));
                            else return $this->encrypt($iv, $key, $source_data);
                    }
            } catch (Exception $ex) {
                    $this->writeLog("AESCBCPKCS5() Exception Code ".$ex->getCode()." : ".$ex-
>getMessage()." in ".$ex->getFile()." on line ".$ex->getLine());
                    return "_FAIL_";
            }
    }
    public function writeLog($strLogText) {
            $log_string = "";
            if (is_array($strLogText)) {
                    $log_string = "[".date("Y/m/d H:i:s")."] ₩r₩n";
                    foreach (array_keys($strLogText) as $key) {
                            $log_string = $log_string."                    [".$key."] =>
".$strLogText[$key]."₩r₩n";
                    }
            } else {
                    $log_string = "[".date("Y/m/d H:i:s")."] ".$strLogText."₩r₩n";
            }
            $log_filenm = $this->LogPath.date("Ymd")."_KMpay.log";
            $log_file = fopen($log_filenm, "a");
            if($log_file == false) return;
            flock($log_file, LOCK_EX);
            //fwrite($log_file, $log_string);
            fputs($log_file, $log_string);
            fflush($log_file);
            flock($log_file, LOCK_UN);
            fclose($log_file);
    }
}

$kmFunc = new kmpayFunc();
$merchantEncKey = "52b017585c98067b";
$paramStr = 'planText';
$encryptStr = $kmFunc->parameterEncrypt($merchantEncKey, $paramStr);
echo "paramStr: " . $paramStr . "₩n";
echo "encrypted: " . $encryptStr . "₩n";

$payReqResult = $kmFunc->parameterEncrypt($merchantEncKey, 'res:000');
$resultString = $kmFunc->parameterDecrypt($merchantEncKey, $payReqResult);
echo "payReqResult: " . $payReqResult . "₩n";
echo "decrypted: " . $resultString . "₩n";

?>
```