OTT PAY

OTTPAY API DOCUMENT

V1.1

Updated At 2017/ 8/ 22

OTT PAY

Version History

Version	Writer	participant	Date	Notes
V1.0				
V1.1	Steven Zhang			Add API URL and test caption

1. Communication Type

Between server and third party clients, they communicate by HTTP POST request, the data format is in JSON. In order to ensure data security, the data adopts two-stage structure, the first level is system parameter, and the second level is the business data. At the same time, the secondary data is encrypted which refer to appendix A. The data encoding format is utf-8. All of the monetary unit involved in the document are in Cent.

Production API : https://frontapi.ottpay.com:443/process

Because WechatPay and AliPay have not been provided the official sandbox test environment, we provide the production API URL to you directly to do online access, all of test payment actions will do the real payment. It will help you meet more comprehensive API returns the result. At the same time, because the payment is real, the merchant can directly check the payment record and payment status through the “merchant accounting checking system”, which can help you to improve the debugging efficiency.

2. Common Data Format

<table>
<tr><td>Requester</td><td colspan="4">Third party client</td></tr>
<tr><td>Responser</td><td colspan="4">Server</td></tr>
<tr><td>Request params</td><td colspan="4"><table>
<tr><th>No.</th><th>Field Name</th><th>Describtion</th><th>Notes</th></tr>
<tr><td>1</td><td>action</td><td>Action type</td><td></td></tr>
<tr><td>2</td><td>version</td><td>Version Number</td><td></td></tr>
<tr><td>4</td><td>merchant_id</td><td>Merchant ID</td><td>We will give you this through email once you become our payment merchant.</td></tr>
<tr><td>5</td><td>data</td><td>Encrypt data</td><td></td></tr>
<tr><td>6</td><td>md5</td><td>Data summary</td><td></td></tr>
</table></td></tr>
<tr><td>Response params</td><td colspan="4"><table>
<tr><th>No.</th><th>Field Name</th><th>Describtion</th><th>Notes</th></tr>
<tr><td>1</td><td>rsp_code</td><td>Response code</td><td></td></tr>
</table></td></tr>
</table>

	2	rsp_msg	Response message	
	4	data	Encrypt data	
	5	md5	Message-Digest	

3. APIs

1. Payment API

a) Active Pay (Payment by user scan QR code actively)

Params Name		Describtion	Required	Notes
action		Action name	Y	ACTIVEPAY
version		Version number	Y	Default 1.0
merchant_id		Merchant ID	Y	
data	order_id	Order ID	N	This ID could be generated by customizable rules in client's code, if not passed, it also could be generated in the background system by default.
	amount	Transaction Amount	Y	
	biz_type	Business Type	Y	ALIPAY, WECHATPAY
	call_back_url	Call back URL	Y	
md5		Message Digest	Y	

The results of returned：

Params Name		Describtion	Required	Notes
rsp_code		Response code	Y	Please refer to appendix C
rsp_msg		Response message	Y	
data	code_url	QR code URL	Y	
	order_id	Order ID	Y	

md5	Message Digest	Y	

Call back data:

Params Name	Describtion	Required	Notes
rsp_code	Response code	Y	Please refer to appendix C
rsp_msg	Response message	Y	
merchant_id	Merchant ID	Y	
finish_time	Payment time	Y	YyyyMMddHHmmss (Beijing)
order_id	Order ID	Y	
amount	Transaction amount	Y	
tip	Tip amount	Y	
md5	Message Digest	Y	

b) Passive Pay (Payment by POS scan user's auth code)

Params Name		Describtion	Required	Notes
action		Response code	Y	PASSIVEPAY
version		Response message	Y	Defaut 1.0
merchant_id		Merchant ID	Y	
data	order_id	Order ID	N	
	amount	Transaction amount	Y	
	auth_code	Auth code	Y	User will show you from their Apps.
	biz_type	Business type	Y	ALIPAY, WECHATPAY

md5	Message Digest	Y	

Result of returned：

Params Name		Describtion	Required	Notes
rsp_code		Response code	Y	Please refer to appendix C
rsp_msg		Response message	Y	
data	order_status	Order Status	Y	
	trade_time	Transact time	N	
	order_id	Order ID	Y	
	buyer_login_id	Buyer ID	Y	
	total_amount	Order total amount	Y	
	receive_amount	Merchant receive amount	Y	
	pay_amount	User pay amount	Y	
md5		Message digest	Y	

c) Query transaction

Params Name		Describtion	Required	Notes
action		Response code	Y	order_status_QUERY
version		Response message	Y	Default 1.0
merchant_id		Merchant ID	Y	
data	order_id	Order ID	Y	

md5	Message digest	Y	

Results of returned：

Params Name		Describtion	Required	Notes
rsp_code		Reponse code	Y	Please refer to appendix C
rsp_msg		Response message	Y	
data	order_status	Order status	Y	
	trade_time	Transaction time	N	
	order_id	Order ID	N	
	buyer_login_id	Buyer ID	N	
	total_amount	Order total amount	N	
	receive_amount	Merchant receive amount	N	
	pay_amount	User pay amount	N	
	refund_fee	Refund fee	N	Exist when you query a refund order
md5		Message digest	Y	

d) Refund

Params Name	Describtion	Required	Notes
action	Action name	Y	REFUND
version	Version number	Y	Default 1.0

merchant_id		Merchant ID	Y	
data	order_id	Refund order ID	Y	It will be a new order id for refund order.
	ori_order_id	Original order ID	Y	
	refund_amt	Refund amount	Y	
md5		Message digest	Y	

Results of returned：

Params Name		Describtion	Required	Notes
rsp_code		Response code	Y	Please refer to appendix C
rsp_msg		Response message	Y	
data	order_status	Order status	Y	
	trade_time	Transact time	N	
	order_id	Order ID	Y	
	refund_amt	Refund amount	Y	
md5		Message digest	Y	

e) Query refund

Params Name	Describtion	Required	Notes
action	Action name	Y	REFUND_order_status_QUERY
version	Version number	Y	Default 1.0
merchant_id	Merchant ID	Y	

data	order_id	Order ID	Y	
md5		Message digest	Y	

Results of returned：

Params Name		Describtion	Required	Notes
rsp_code		Response code	Y	Please refer to appendix C
rsp_msg		Response message	Y	
data	order_id	Order ID	Y	
	order_status	Order status	Y	
	refund_amt	Refund amount	Y	
md5		Message digest	Y	

2. Pre-auth payment API

a) Pre-auth order

Params Name		Describtion	Required	Notes
action		Action name	Y	PRE_AUTH
version		Version number	Y	Default 1.0
merchant_id		Merchant ID	Y	
data	order_id	Order id	Y	
	amount	Pre-auth amount	Y	
	auth_code	User auth code	Y	
	biz_type	Business type	Y	ALIPAY,

				WECHATPAY
md5		Message digest	Y	

Results of returned：

Params Name		Describtion	Required	Notes
rsp_code		Response code	Y	Please refer to appendix C
rsp_msg		Response message	Y	
data	order_id	Order ID	Y	
	order_status	Order status	Y	
	trade_time	Transaction time	N	
	buyer_login_id	Buyer ID	N	
	total_amount	Order total amount	N	
	receive_amount	Merchant receive amount	N	
	pay_amount	User pay amount	N	
md5		Message digest	Y	

b) Pre-auth payment finished（according the original and final transaction amount, then people use auth_code to finish transaction or refund）

Params Name	Describtion	Required	Notes
action	Action name	Y	AUTH_FINISH

version		Version number	Y	Default 1.0
merchant_id		Merchant ID	Y	
data	ori_order_id	Pre-auth original order ID	Y	
	amount	Final transaction amount	Y	
	auth_code	User's auth_code	N	If people need to pay additional amount, this field would be used
	biz_type	Business type	Y	ALIPAY, WECHATPAY
md5		Message digest	Y	

Results of returned：

Params Name		Describtion	Required	Notes
rsp_code		Reponse code	Y	Please refer to appendix C
rsp_msg		Response message	Y	
data	order_id	Order ID	Y	
	order_status	Order status	Y	
	trade_time	Transact time	N	
	pre_auth_amt	Pre-auth amount	N	
	final_amount	Final transaction amout	Y	

md5	Message digest	Y	

3. Exchange rate query

Params Name		Describtion	Required	Notes
action		Action name	Y	EX_RATE_QUERY
version		Version number	Y	Default 1.0
merchant_id		Merchant ID	Y	
data	fee_type	Currency type	Y	ISO-4217
md5		Message digest	Y	

Results of returned：

Params Name		Describtion	Required	Notes
rsp_code		Response code	Y	Please refer to appendix C
rsp_msg		Response message	Y	
data	fee_type	Currency type	Y	
	rate	Exchange rate	Y	If rate = 6.5，then you will get the number like 650000000, ten to the eighth
md5		Message digest	Y	

Appendix A·the way of data encryption

Encryption procedure：

AES Encryption mode is ECB，fill mode：PKCS5Padding

1. Taking dictionary sort for each subdomain key in the data domain, and then splice these key's value, use the original data string to do MD5, and then convert to uppercase, so far we can get the MD5 value as the http request parameter.

Example： orderId=1234567890， idCardName=Tom， idCardNum=123456789012345678，bankCardNum=62000000000010，mobile=13711111111，idCardType=1；

After sort and splice：
62000000000010Tom123456789012345678113711111111234567890

MD5: C12F9560769C2CB55E6954935B325916

2. Splice the MD5 value we got in first step and your Signkey, then do MD5 and convert to uppercase, this MD5 value will be the AES KEY. Then, use AES KEY to do the AES encryption in the JSON string data which is the subdomain of data domain. And then, use base64 to encode. So far, you get the value which will be the "data" parameter passed through http request.

Example：Merchant SignKey：6698851A525C9433

Do MD5 : C12F9560769C2CB55E6954935B3259166698851A525C9433

Get AESKEY: EF1712FC070C2C21

Subdomain JSON string：

{"orderId":"1234567890","idCardName":"Tom","idCardNum":"123456789012345678","bankCardNum":"62000000000010","mobile":"13711111111","idCardType":"1"}

Do AES and base64：

Y2+GoqBBPc6EJ9JzXRFoOziKd+DqWg5FiUZY1IYoVBVv0xfFznT9/qanMpiNEamEN2NM7J+hxoRn8VGSJImA2soeip9nYr+VJvTXe7D8j4aXiKyFipuPVCQLiCDg3jkJP+S2EezYMf7crqKY/YAni1CCeIwr2aJfFVT1vYhsWIm8t3eyPL//cY4kwombAKhE2gAdEFXz6gVvencz80aRWQ==

Decryption procedure：

1. Splice the MD5 value in the returned data and your SignKey, do MD5 and convert to uppercase, you will get the AES KEY. And then use BASE64 decode the data field and use AES to decrypt, you will see the data detail.

Appendix B·Response code

Response code	Response message
SUCCESS	success
PROCESSING	transaction processing
PWD_INCORRECT	Password is incorrect
FAIL	Failure.
TIMEOUT	timeout.
SYSTEM_ERROR	System error.
PARAM_INVALID	invalid parameter
PARAM_ERROR	parameter format error
ORDER_PAID	order paid
ORDER_USED	order number duplicated
ORDER_TYPE_ERROR	This is not a pre-authoried order.
DB_ERROR	database error
MER_NO_AUTH	no permission for the merchant
CODE_EXPIRE	QR code expired, please refresh and try again
AMOUNT_NOT_ENOUGH	insufficient balance
CARD_NOT_SUPPORT	unsupported card
CURRENCY_NOT_SUPPORT	unsupported currency
ORDER_CLOSED	order closed
ORDER_REVERSED	order cancelled
BANK_ERROR	banking system anomaly
AUTH_CODE_ERROR	authorization code parameter error
AUTH_CODE_INVALID	authorization code validation fail
REQ_AGAIN	please restart a transaction

REFUND_INVALID	refund amount greater than the amount paid
MERCHANT_INVALID	Merchant information does not match
REFUND_FEE_MISMATCH	refund amount check fail
REFUND_NOT_EXIST	refund order not exist
BODY_NOT_ENOUGH	insufficient message header
ORDER_NOT_EXIST	original order not exist
PRE_AUTH_FINISHED	original order has been finished
REVERSE_NOT_SUPPORT	original order does not support revocation.
STATU_FAIL	accepting institution abnormal state
SIGN_FAIL	verification signature fail
REFUND	transfer to refund
REFUND_SUCCESS	Refund success.
PARTIAL_REFUNDED	Partial Refunded
FULLY_REFUNDED	Fully Refunded.
REF_SUCCESS_CHANGE	refund success, user card frozen, need human intervention
NOTPAY	OrderStatus.notpay, unpaid
PAYERROR	payment failed
NO_CHANNEL	no channel supported
MER_NO_INFO	Failed to get merchant information.
MER_ACCOUNT_INSUFFICIENT	Merchant account balance is insufficient.Don't allow to refund.
OTHER	other case.
SERIALNO_EXIST	The account request serial number already exists, can not be submitted again.
REASON_TRADE_BEEN_FREEZEN	Corresponding trade is been frozen due to security issues.Please contact Alipay technical

	support
HAS_NO_PRIVILEGE	Has no privilege.Please contact Alipay technical support
BUYER_ERROR	The buyer does not exist.
BUYER_ENABLE_STATUS_FORBID	Buyer account status prohibits the refund
SELLER_ERROR	The seller does not exist.
TRADE_STATUS_ERROR	The trade status is illegal.
TRADE_HAS_FINISHED	The trade has been finished.
MERCHANT_BALANCE_NOT_ENOUGH	Merchant balance is not enough for refund.
TRADE_CANCEL_TIME_OUT	The cancellation request is beyond the opening hours.
UNKNOW	Indicates payment result is unknown);

Appendix C·Sample Code

ActivePayTest.java

```
package com.ottpay.front.test;


import com.google.gson.Gson;

import com.google.gson.JsonSyntaxException;

import org.apache.commons.beanutils.BeanUtils;

import org.apache.commons.codec.digest.DigestUtils;

import org.apache.commons.lang3.builder.ToStringBuilder;

import org.springframework.util.Base64Utils;


import java.util.*;


/**

 * com.ottpay.front.test

 * ott_dev

 *

*/

public class ActivePayTest {

    static String key = "E1BF9A83E9EADF2C";

    static Gson gson = new Gson();

    static String url = " https://frontapi.ottpay.com:443/process";


    public static void main(String[] args) {

        Map<String, String> bo = new HashMap<>();

        bo.put("order_id", "ACT" + System.currentTimeMillis());

        bo.put("call_back_url", "http://www.tianbaotravel.com");
```

```
        bo.put("biz_type", "WECHATPAY");
        bo.put("amount", "1");
        Map map = doSend(bo, "ACTIVEPAY");
}
/**
 * 将 Map 存储的对象，转换为 key=value&key=value&的字符,并将 key 按顺序排序
        Convert the Map object to such as Key=value&key=value& string, and order by key
 */
public static String sortStringByMap(Map<String, String> map) {
        ArrayList<String> list = new ArrayList<String>();
        String[] arrayToSort = map.keySet().toArray(new String[map.keySet().size()]);
        Arrays.sort(arrayToSort, String.CASE_INSENSITIVE_ORDER);
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < arrayToSort.length; i++) {
                sb.append(map.get(arrayToSort[i]));
        }
        return sb.toString();
}

private static String signByMD5(Map<String, String> objectAsMap) {
        String result = sortStringByMap(objectAsMap);
        String sign = DigestUtils.md5Hex(result).toUpperCase();
        return sign;
}

private static String signByMD5(Object obj) {
        Map<String, String> objectAsMap;
        try {
```

```
            objectAsMap = BeanUtils.describe(obj);
            objectAsMap.remove("class");
            return signByMD5(objectAsMap);
        } catch (Exception e) {
            throw new RuntimeException("bean convert map fail", e);
        }
    }

    /**
     * Encryption
     *
     * @param data json
     * @param key
     * @param md5
     * @return
     */
    private static String encrypted(String data, String key, String md5) {
        String aesKeyStr = DigestUtils.md5Hex(md5 + key).substring(8, 24).toUpperCase();
        byte[] encrypt = AES.encrypt(data.getBytes(), aesKeyStr.getBytes());
        return Base64Utils.encodeToString(encrypt);
    }

    /**
     * Decryption
     *
     * @param data
     * @param key
     * @param md5
```

```
 * @return
 */
public static String decipher(String data, String key, String md5) {

    byte[] orgData = Base64Utils.decodeFromString(data);
    String aesKeyStr = DigestUtils.md5Hex(md5 + key).substring(8, 24).toUpperCase();
    byte[] aesKey = aesKeyStr.getBytes();
    String decData = new String(AES.decrypt(orgData, aesKey));

    TreeMap<String, String> treeMap = null;
    try {
        treeMap = gson.fromJson(decData, TreeMap.class);
    } catch (JsonSyntaxException e) {

        return null;
    }
    StringBuilder stb = new StringBuilder();
    for (String tk : treeMap.keySet()) {
        stb.append(treeMap.get(tk));
    }

    String calmd5 = DigestUtils.md5Hex(stb.toString());
    if (calmd5.toUpperCase().equals(md5.toUpperCase())) {
        return decData;
    } else {
        throw new RuntimeException("check sign fail");
    }
}
```

```
    private static Map doSend(Map<String,String> bo, String type) {
        Map<String, String> vo = new HashMap<>();
        vo.put("action", type);
        vo.put("version", "1.0");
        vo.put("merchant_id", "ON00000010");
        String md5 = signByMD5(bo);
        String encrypted = encrypted(gson.toJson(bo), key, md5);
        vo.put("data", encrypted);
        vo.put("md5", md5);
        Map<String, String> appRespVO = gson.fromJson(HttpClientUtil.doPost(url, gson.toJson(vo)), Map.class);
        System.out.println(ToStringBuilder.reflectionToString(appRespVO));
        Map decipher = gson.fromJson(decipher(appRespVO.get("data"), key, appRespVO.get("md5")), Map.class);
        System.out.println("data:" + decipher.toString());
        return decipher;
    }
}
```

AES.java

```
/**
 * Copyright: Copyright (c)2017
 * Company: OTTPay
 */
package com.ottpay.front.test;
```

```
import com.icardpay.bussiness.front.util.Base64;
import com.icardpay.bussiness.front.util.CheckUtils;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.spec.SecretKeySpec;
import java.io.UnsupportedEncodingException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;



/**
 * AES Encryption Tool
 * @author:
 * @version:
 */
public class AES {
    /**
     * Encryption
     *
     * @param data
     *                  Need to be encrypted data
     * @param key
     *                  key
     * @return
     */
```

```
public static byte[] encrypt(byte[] data, byte[] key) {
        CheckUtils.notEmpty(data, "data");
        CheckUtils.notEmpty(key, "key");
        if(key.length!=16){
            throw new RuntimeException("Invalid AES key length (must be 16 bytes)");
        }
        try {
            SecretKeySpec secretKey = new SecretKeySpec(key, "AES");
            byte[] enCodeFormat = secretKey.getEncoded();
            SecretKeySpec seckey = new SecretKeySpec(enCodeFormat, "AES");
            Cipher cipher = Cipher.getInstance("AES");// create
            cipher.init(Cipher.ENCRYPT_MODE, seckey);// init
            byte[] result = cipher.doFinal(data);
            return result; // encryption
        } catch (Exception e){
            throw new RuntimeException("encrypt fail!", e);
        }
}


/**
  * decryption
  *
  * @param data
  *                  Need to be decrypted data
  * @param key
  *                  key
  * @return
  */
```

```
public static byte[] decrypt(byte[] data, byte[] key) {
    CheckUtils.notEmpty(data, "data");
    CheckUtils.notEmpty(key, "key");
    if(key.length!=16){
        throw new RuntimeException("Invalid AES key length (must be 16 bytes)");
    }
    try {
        SecretKeySpec secretKey = new SecretKeySpec(key, "AES");
        byte[] enCodeFormat = secretKey.getEncoded();
        SecretKeySpec seckey = new SecretKeySpec(enCodeFormat, "AES");
        Cipher cipher = Cipher.getInstance("AES");// create
        cipher.init(Cipher.DECRYPT_MODE, seckey);// init
        byte[] result = cipher.doFinal(data);
        return result; // decryption
    } catch (Exception e){
        throw new RuntimeException("decrypt fail!", e);
    }
}

public static String encryptToBase64(String data, String key){
    try {
        byte[] valueByte = encrypt(data.getBytes("UTF-8"), key.getBytes("UTF-8"));
        return new String(Base64.encode(valueByte));
    } catch (UnsupportedEncodingException e) {
        throw new RuntimeException("encrypt fail!", e);
    }

}
```

```
public static String decryptFromBase64(String data, String key){
    try {
        byte[] originalData = Base64.decode(data);
        byte[] valueByte = decrypt(originalData, key.getBytes("UTF-8"));
        return new String(valueByte, "UTF-8");
    } catch (UnsupportedEncodingException e) {
        throw new RuntimeException("decrypt fail!", e);
    }
}

public static String encryptWithKeyBase64(String data, String key){
    try {
        byte[] valueByte = encrypt(data.getBytes("UTF-8"), Base64.decode(key));
        return new String(Base64.encode(valueByte));
    } catch (UnsupportedEncodingException e) {
        throw new RuntimeException("encrypt fail!", e);
    }
}

public static String decryptWithKeyBase64(String data, String key){
    try {
        byte[] originalData = Base64.decode(data);
        byte[] valueByte = decrypt(originalData, Base64.decode(key));
        return new String(valueByte, "UTF-8");
    } catch (UnsupportedEncodingException e) {
        throw new RuntimeException("decrypt fail!", e);
    }
```

```
    }

    public static byte[] genarateRandomKey(){
        KeyGenerator keygen = null;
        try {
            keygen = KeyGenerator.getInstance("AES");
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException("genarateRandomKey fail!", e);
        }
        SecureRandom random = new SecureRandom();
        keygen.init(random);
        Key key = keygen.generateKey();
        return key.getEncoded();
    }

    public static String genarateRandomKeyWithBase64(){
        return new String(Base64.encode(genarateRandomKey()));
    }

}
```

HttpClientUtil.java

```
package com.ottpay.front.test;

import org.apache.http.HttpEntity;
```

```
import org.apache.http.HttpStatus;

import org.apache.http.NameValuePair;

import org.apache.http.client.config.RequestConfig;

import org.apache.http.client.entity.UrlEncodedFormEntity;

import org.apache.http.client.methods.CloseableHttpResponse;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.conn.ssl.SSLConnectionSocketFactory;

import org.apache.http.conn.ssl.SSLContexts;

import org.apache.http.entity.StringEntity;

import org.apache.http.impl.client.CloseableHttpClient;

import org.apache.http.impl.client.HttpClients;

import org.apache.http.impl.conn.PoolingHttpClientConnectionManager;

import org.apache.http.message.BasicNameValuePair;

import org.apache.http.util.EntityUtils;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Component;


import javax.net.ssl.SSLContext;

import java.io.IOException;

import java.nio.charset.Charset;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;


@Component
```

```
public class HttpClientUtil {


    private static PoolingHttpClientConnectionManager connManager = null;
    private static RequestConfig requestConfig;
    static final int CONNECTION_TIMEOUT = 30000;// connection timeout
    static final int SO_TIMEOUT = 60000;// data transfer timout
    static Long CONN_MANAGER_TIMEOUT = 500L;
    private static final Logger logger = LoggerFactory.getLogger(HttpClientUtil.class);
    private static RequestConfig.Builder configBuilder = RequestConfig.custom();
    static {
        // create connection pool
        connManager = new PoolingHttpClientConnectionManager();
        // set connection pool size
        connManager.setMaxTotal(100);
        connManager.setDefaultMaxPerRoute(connManager.getMaxTotal());



        // set connection timeout
        configBuilder.setConnectTimeout(CONNECTION_TIMEOUT);
        // set read data timeout
        configBuilder.setSocketTimeout(SO_TIMEOUT);
        // set connection request timeout
        configBuilder.setConnectionRequestTimeout(CONNECTION_TIMEOUT);
        // test before connect
        configBuilder.setStaleConnectionCheckEnabled(true);
        requestConfig = configBuilder.build();
    }
```

```
/**
 * Send POST request（HTTP）without data
 *
 * @param apiUrl
 * @return
 */
public static String doPost(String apiUrl) {
    return doPost(apiUrl, new HashMap<String, Object>());
}

/**
 * Send POST request（HTTP）with K-V format data
 *
 * @param apiUrl
 * @param params map
 * @return
 */
public static String doPost(String apiUrl, Map<String, Object> params) {
    CloseableHttpClient httpClient = HttpClients.createDefault();
    String httpStr = null;
    HttpPost httpPost = new HttpPost(apiUrl);
    CloseableHttpResponse response = null;

    try {
        httpPost.setConfig(requestConfig);
        List<NameValuePair> pairList = new ArrayList<NameValuePair>(params.size());
        for (Map.Entry<String, Object> entry : params.entrySet()) {
            NameValuePair pair = new BasicNameValuePair(entry.getKey(), entry
```

```
                                .getValue().toString());
                        pairList.add(pair);
                    }
                    httpPost.setEntity(new                    UrlEncodedFormEntity(pairList,
Charset.forName("UTF-8")));
                    response = httpClient.execute(httpPost);
                    System.out.println(response.toString());
                    HttpEntity entity = response.getEntity();
                    httpStr = EntityUtils.toString(entity, "UTF-8");
                } catch (IOException e) {
                     logger.error(e.getMessage(),e);
                } finally {
                    if (response != null) {
                        try {
                            EntityUtils.consume(response.getEntity());
                        } catch (IOException e) {
                            logger.error(e.getMessage(),e);
                        }
                    }
                }
                return httpStr;
            }

            /**
              * Send POST request with timout setting
              *
              * @param apiUrl string
              * @param params map
```

```
     * @param params second
     * @return
     */
    public static String doPost(String apiUrl, Map<String, Object> params,int timeout) {
            configBuilder.setSocketTimeout(timeout);
            return doPost(apiUrl,params);
    }

    /**
     * 发送 POST 请求（HTTP），JSON 形式
     *
     * @param apiUrl
     * @param json json 对象
     * @return
     */
    public static String doPost(String apiUrl, Object json) {
        CloseableHttpClient httpClient = HttpClients.createDefault();
        String httpStr = null;
        HttpPost httpPost = new HttpPost(apiUrl);
        CloseableHttpResponse response = null;

        try {
            httpPost.setConfig(requestConfig);
            StringEntity stringEntity = new StringEntity(json.toString(), "UTF-8");//解决中文乱
码问题
            stringEntity.setContentEncoding("UTF-8");
            stringEntity.setContentType("application/json");
            httpPost.setEntity(stringEntity);
```

```
            response = httpClient.execute(httpPost);
            HttpEntity entity = response.getEntity();
            System.out.println(response.getStatusLine().getStatusCode());
            httpStr = EntityUtils.toString(entity, "UTF-8");
        } catch (IOException e) {
             logger.error(e.getMessage(),e);
        } finally {
            if (response != null) {
                try {
                    EntityUtils.consume(response.getEntity());
                } catch (IOException e) {
                    logger.error(e.getMessage(),e);
                }
            }
        }
        return httpStr;
    }

    /**
     * 发送 SSL POST 请求（HTTPS），K-V 形式
     *
     * @param apiUrl API 接口 URL
     * @param params 参数 map
     * @return
     */
    public static String doPostSSL(String apiUrl, Map<String, Object> params) {
        CloseableHttpClient httpClient = HttpClients.custom().setSSLSocketFactory(createSSLConnSocketFactory()).setConnectionManager(connManager).setDefaultRequestConfig(requestConfig).build();
```

```
HttpPost httpPost = new HttpPost(apiUrl);
CloseableHttpResponse response = null;
String httpStr = null;

try {
    httpPost.setConfig(requestConfig);
    List<NameValuePair> pairList = new ArrayList<NameValuePair>(params.size());
    for (Map.Entry<String, Object> entry : params.entrySet()) {
        NameValuePair pair = new BasicNameValuePair(entry.getKey(), entry
                .getValue().toString());
        pairList.add(pair);
    }
    httpPost.setEntity(new UrlEncodedFormEntity(pairList,
Charset.forName("utf-8")));
    response = httpClient.execute(httpPost);
    int statusCode = response.getStatusLine().getStatusCode();
    if (statusCode != HttpStatus.SC_OK) {
        return null;
    }
    HttpEntity entity = response.getEntity();
    if (entity == null) {
        return null;
    }
    httpStr = EntityUtils.toString(entity, "utf-8");
} catch (Exception e) {
     logger.error(e.getMessage(),e);
} finally {
    if (response != null) {
```

```
                try {
                    EntityUtils.consume(response.getEntity());
                } catch (IOException e) {
                    logger.error(e.getMessage(),e);
                }
            }
        }
        return httpStr;
    }

    /**
     * 发送 SSL POST 请求（HTTPS），JSON 形式
     *
     * @param apiUrl API 接口 URL
     * @param json JSON 对象
     * @return
     */
    public static String doPostSSL(String apiUrl, Object json) {
        CloseableHttpClient httpClient = HttpClients.custom().setSSLSocketFactory(createSSLConnSocketFactory()).setConnectionManager(connManager).setDefaultRequestConfig(requestConfig).build();
        HttpPost httpPost = new HttpPost(apiUrl);
        CloseableHttpResponse response = null;
        String httpStr = null;

        try {
            httpPost.setConfig(requestConfig);
            StringEntity stringEntity = new StringEntity(json.toString(), "UTF-8");//解决中文乱码问题
```

```
            stringEntity.setContentEncoding("UTF-8");
            stringEntity.setContentType("application/json");
            httpPost.setEntity(stringEntity);
            response = httpClient.execute(httpPost);
            int statusCode = response.getStatusLine().getStatusCode();
            if (statusCode != HttpStatus.SC_OK) {
                return null;
            }
            HttpEntity entity = response.getEntity();
            if (entity == null) {
                return null;
            }
            httpStr = EntityUtils.toString(entity, "utf-8");
        } catch (Exception e) {
            logger.error(e.getMessage(),e);
        } finally {
            if (response != null) {
                try {
                    EntityUtils.consume(response.getEntity());
                } catch (IOException e) {
                    logger.error(e.getMessage(),e);
                }
            }
        }
        return httpStr;
    }

    /**
```

```
     * 创建 SSL 安全连接
     *
     * @return
     */
    private static SSLConnectionSocketFactory createSSLConnSocketFactory() {

        SSLContext sslContext = SSLContexts.createSystemDefault();
        SSLConnectionSocketFactory sslsf = new SSLConnectionSocketFactory(
                sslContext,
                SSLConnectionSocketFactory.STRICT_HOSTNAME_VERIFIER);
        return sslsf;
    }

    public static    String doGet(String apiUrl, Object json) {
        return doPost(apiUrl, json);
    }

    public static String doGet(String url)
    {
         CloseableHttpClient httpClient = HttpClients.createDefault();
            String httpStr = null;
            HttpGet httpGet = new HttpGet(url);
//          HttpPost httpPost = new HttpPost(apiUrl);
            CloseableHttpResponse response = null;

            try {
                httpGet.setConfig(requestConfig);
                //StringEntity stringEntity = new StringEntity(json.toString(), "UTF-8");//解决
```

中文乱码问题

```
                response = httpClient.execute(httpGet);
                HttpEntity entity = response.getEntity();
                System.out.println(response.getStatusLine().getStatusCode());
                httpStr = EntityUtils.toString(entity, "UTF-8");
            } catch (IOException e) {
                 logger.error(e.getMessage(),e);
            } finally {
                if (response != null) {
                    try {
                        EntityUtils.consume(response.getEntity());
                    } catch (IOException e) {
                        logger.error(e.getMessage(),e);
                    }
                }
            }
            return httpStr;
        }


}
```