# 1、1.1、准备环境

## 一、请求地址说明

#### ****1.完整的请求地址****：

上下文 + 能力服务（见业务接口属性）

示例：<https://cd.10010.com/zop> + /link/num/select/v1

得到地址：<https://cd.10010.com/zop/link/num/select/v1>

#### ****2.测试地址****

* 外网 <https://demo.mall.10010.com:8104/zop/>
* 内网1 （可选） <http://10.20.25.119:8104/zop/> （DCN地址：<http://132.38.2.50:8104/zop/> )
* 内网2（可选） <http://10.142.151.87:8104/zop/> （DCN地址：<http://132.38.0.114:8104/zop/> )

#### ****3.生产地址****

外网地址**(优先推荐)**

**2I2C业务：**<https://cd.10010.com/zop> **注：cd.10010.com域名经过阿里CDN全站加速；以/zop/link/开头的请求接口都支持该域名，部分老旧服务如不支撑请及时反馈给商城管理员。**

**码上购业务：** <https://msgo.10010.com/zop>

* 内网地址 <http://132.46.125.65:8301/zop/link/>...

## 二、ZOP接入申请

（1）**需求人员**做好接入权限申请

联通需求负责人搜集接入方（第三方）需求信息，提前联系总部负责人李静雅，建立好接入方公司信息，然后通过联通中台-线上移网订单接入菜单，首次接入请选择【线上移网订单新接入申请】子菜单，非首次接入请选择【线上移网订单已接入修改】子菜单，可申请ZOP接口接入权限，或者申请订单消息数据回传；提交完申请后等待联通总部管理员审核，审核通过后会通过邮件将鉴权接入参数（appCode，HmacMd5，AES等参数）发送给联通需求负责人和合作方负责人。

* 注：联通总部会对接入方ZOP接口规范性调用进行实时监控，如果存在疑似恶意调用，联通总部有权要求接入方配合提供接入方的测试环境的外网ip和生产环境的外网ip，进行安全策略限制；对于多次存在疑似恶意调用的接入方，联通总部有权回收ZOP接入权限。

（2）**开发人员**做好代码接入准备

* 请关注文档1.2-1.7部分

# 2、1.2、请求报文

#### 1.请求报文

* 根节点

| **节点** | **父节点** | **类型** | **长度** | **约束** | **注释** |
| --- | --- | --- | --- | --- | --- |
| appCode | - | string | F32 | 1 | 接入编码 |
| reqObj | - | - | \* | 1 | 请求体 |
| head | reqObj | - | \* | 1 | 头部 |
| body | reqObj | - | \* | 1 | 业务节点，具体业务节点见业务接口 |

* head节点

| **节点** | **父节点** | **类型** | **长度** | **约束** | **注释** |
| --- | --- | --- | --- | --- | --- |
| timestamp | head | String | V40 | 1 | 请求的当前时间，时间戳，格式:yyyy-MM-dd HH:mm:ss.SSS |
| uuid | head | string | V60 | 1 | 流水号,java实现uuid=String.valueOf(UUID.randomUUID()) |
| sign | head | string | V200 | 1 | 签名 |

#### 2.请求报文-不加密请求示例

|  |
| --- |
| {  "appCode": "88BEE3C956B24DA888BB3DE4D2EAEC91",  "reqObj": {  "head": {  "sign": "hIQyUYSHRp6S5ruiNaUxnA==",  "timestamp": "2017-10-26 20:16:48.126",  "uuid": "3d737228-a8b9-4e6e-b183-349706d4d977"  },  "body": {  "dataList": [  {  "appointmentDate": "2017-11-11",  "cityCode": "110",  "comeFrom": "alipay",  "price": "1048000",  "productName": "1048元50M包一年联通宽带产品",  "provinceCode": "11",  "thirdNo": "123456789",  "userAddress": "北京市西城区西环广场T2座12层c8室",  "userCardId": "362423190008293018",  "userName": "张三",  "userPhone": "18612031035"  }  ]  }  }  } |

* 以某接口为例，注意body节点属于业务节点

#### 3.请求报文-节点加密参数示例

|  |
| --- |
| {  "appCode": "9941C35FE3404367AE78DD8A458264A5",  "reqObj": "14zdYfiO9GZjN77L3x1Twgg066l7t9TmZmj14p88vkVhWkf5ct1nLsvkGtkRMziwhpzQgSghCBvs\n+vVLb7Q4LZRdcnTFaRYs8LPYgTZQ2aWYzioIPGrBxg9WeAAAUrtCIIeC3utEHzETTs5s8JPMZPC7\nqci4+u99h3Ko7gv/AKLH7pL5ok6w6nlnluu4Ece4O/NfWUAqY0rpV4E4tIPH3SvnjsZ7yU1Oal8O\n4alGpUg+xbP8WXGI4wkZr/HEC1DLUCPymQUInVtAjM/3FS9S5/6TqKn0V8EhbDPZZY/SEbdb5I8Q\nUlLvopleL4gltFRCrwQ/BUfulLY48KWw+Un/CV9fYr/7zk62g+TpOXc+cGDlloJtgBHIDqIsShw7\nYLBFifQsv1DGCbSfmXbW0Srikys8/tRtwD3PK8r4KntM45nGe2FSIcy+QPdegO8hQloOFajGoSsm\ny10kJcg4Ehig0HqwEcA+MinqjJMZtPXn8Oq3Vc6U+CeHSz/6ZBIQrSiwiWMprMyCyYClLnMt9MzM\nK286DuSKJrkFzVVyC//Cqcu1fSStDlT4pHl9szrz130123dCutDwasvReb3YsmbRzDBcLv+6Qpws\nyZsdpuBw60QFN+aLtU/UNHk6qKlalEsK"  } |

# 3、1.3、鉴权说明

#### 一、参数AES加密说明

一般业务参数都要进行AES加密

1.加密的节点是reqObj

2.加密原始数据的步骤：

首先，把AES密钥BASE64 decode (因为给的密钥进行了Base64 encode)

然后，拿到得到的值，进行对原数据（reqObj下的json数据）进行AES加密

最后，把加密的数据Base64 encode ，得到最后的加密数据

java示例：

|  |
| --- |
| public static final String ENCODE = "UTF-8";  public static final String CIPHER\_ALGORITHM = "AES";  private static final String KEY\_MAC = "HmacMD5";  public static final String CIPHER\_ALGORITHM\_INS = "AES/ECB/PKCS5Padding";  /\*\*  \* AES加密  \* @param keyStr 密钥  \* @param dataStr 原始数据  \*/  public static String encrypt(String keyStr,String dataStr) throws Exception {  BASE64Decoder decoder = new BASE64Decoder();  SecretKey secretKey = new SecretKeySpec(decoder.decodeBuffer(keyStr), CIPHER\_ALGORITHM);  //Cipher完成加密或解密工作类  Cipher cipher = Cipher.getInstance(CIPHER\_ALGORITHM\_INS);  //对Cipher初始化，解密模式  cipher.init(Cipher.ENCRYPT\_MODE, secretKey);  //加密data  byte[] cipherByte = cipher.doFinal(dataStr.getBytes(ENCODE));  BASE64Encoder base64Encoder = new BASE64Encoder();  return base64Encoder.encode(cipherByte);  } |

.net语言示例：

|  |
| --- |
| /// <summary&  /// AES 加密  /// </summary&  /// <param name="str"&明文(待加密),reqObj的json字符串</param&  /// <param name="bytes"&密文,Convert.FromBase64String(AES)的到的字节数组</param&  /// <returns&</returns&  public static string AesEncrypt(string str, byte[] bytes)  {  if (string.IsNullOrEmpty(str)) return null;  Byte[] toEncryptArray = Encoding.UTF8.GetBytes(str);  RijndaelManaged rm = new RijndaelManaged  {  Key = bytes,  Mode = CipherMode.ECB,  Padding = PaddingMode.PKCS7  };  ICryptoTransform cTransform = rm.CreateEncryptor();  Byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);  return Convert.ToBase64String(resultArray, 0, resultArray.Length);  } |

php示例：

|  |
| --- |
| /\*\*  \*AES加密  \*/  function encrypt($data, $key) {  $data = openssl\_encrypt($data, 'aes-128-ecb', base64\_decode($key), OPENSSL\_RAW\_DATA);  return base64\_encode($data);  } |

#### 二、HmacMD5验签说明

1.原数据组成规则： **appCode+appCode值+timestamp+timestamp值+uuid+uuid值+HmacMD5密钥值(平台侧提供的HmacMD5)**

举例： HmacMD5("appCode123456timestamp2017-11-11 00:00:00.000uuid2063978a-b375-4ce1-9eea-98153ea18afawnmHnq7GDz2xqx9AOYAiXgtSy24kOzLaj589myO134HLamdqSVyzs5HU9XlFA==")

2.HmacMD5步骤

首先，得到待签名的原数据X；如： String X = appCode+appCode值+timestamp+timestamp值+uuid+uuid值+HmacMD5密钥值(平台侧提供的HmacMD5)

然后，把平台侧提供的HmacMD5密钥进行BASE64 decode (因为给的密钥进行了Base64 encode),得到 byte[]格式的密钥 Y ,如： byte[] Y = BASE64.Decode (HmacMD5密钥)

其次，根据X和Y,进行HmacMD5签名,得到 摘要值 Z byte[] Z = 签名(X,Y)

* 最后，把Z 数据 进行Base64 encode ，得到最后的签名数据返回。 String result = BASE64.Encode(Z);

3.签名的数据 放在head节点下的sign节点。

4.X 转成byte[]时使用的编码格式为“UTF-8”。

java示例：

|  |
| --- |
| public static final String ENCODE = "UTF-8";  private static final String KEY\_MAC = "HmacMD5";  /\*\*  \* 签名  \* @param key 密钥  \* @param plaintext 原文  \*/  public static String sign(String key,String plaintext) throws Exception {  BASE64Decoder decoder = new BASE64Decoder();  SecretKey secretKey = new SecretKeySpec(decoder.decodeBuffer(key), KEY\_MAC);  Mac mac = Mac.getInstance(secretKey.getAlgorithm());  mac.init(secretKey);  byte[] ret = mac.doFinal(plaintext.getBytes(ENCODE));  BASE64Encoder base64Encoder = new BASE64Encoder();  return base64Encoder.encode(ret);  } |

.net语言示例：

|  |
| --- |
| /// <summary&  ///  /// </summary&  /// <param name="key"&平台侧提供的HmacMD5</param&  /// <param name="dataStr"&appCode+appCode值+timestamp+timestamp值+uuid+uuid值+HmacMD5密钥值</param&  /// <returns&</returns&  public string HmadMD5(string key, string dataStr)  {  HMACMD5 provider = new HMACMD5(Convert.FromBase64String(key));//传入秘钥  byte[] hashedPassword = provider.ComputeHash(Encoding.UTF8.GetBytes(dataStr));//传入加密对象  return Convert.ToBase64String(hashedPassword);  } |

php语言示例

|  |
| --- |
| /\*\*  \* 签名  \* @param key 密钥 平台侧提供的HmacMD5  \* @param dataStr appCode+appCode值+timestamp+timestamp值\*+uuid+uuid值+HmacMD5密钥值  \*/  function sign($key,$dataStr){  return base64\_encode(hash\_hmac("md5", $dataStr, base64\_decode($key), true));  } |

# 4、1.4、java sdk 使用指南

## SDK概述

为了提高开发者的效率，SDK提供了用户鉴权、参数加密、参数校验、接口访问等功能。

## 环境依赖

* 目前官方仅支持JAVA版本SDK
* JAVA SDK 需要依赖Jave SE/EE 1.8及以上

## SDK 下载

* [zop-sdk-2023-09-14.zip](https://card.10010.com/ko-order/miniofile/down?burket=zop-sdk&fileName=zop-sdk-2023-09-14.zip)
* [SDK 升级日志](https://www.showdoc.com.cn/liantongorder/8730519659799331)

注意事项

* Java版本的sdk包含两个包，一个jar可以直接导入工程中使用，第二个是源码包
* 开发者只能通过SDK访问该应用拥有的权限包下所有接口，对无访问权限的接口仍无法调用成功

## 使用示例

实例化ZopClient

|  |
| --- |
| ZopClient zopClient = new ZopHttpClient("appCode", "hmac", "aes"); |

* appCode, hmac, aes：鉴权参数，权限申请，审核通过会发邮件

自定义连接时间读写时间

|  |
| --- |
| ZopClient zopClient = new ZopHttpClient("appCode", "hmac", "aes", connectTimeout, readTimeout); |

* connectTimeout：自定义连接时间，默认3秒
* readTimeout：自定义读写时间，默认10秒

使用代理地址

|  |
| --- |
| ZopClient zopClient = new ZopHttpClient(BASE\_URL,"appCode", "hmac", "aes");  ZopClient zopClient = new ZopHttpClient(BASE\_URL,"appCode", "hmac", "aes", connectTimeout, readTimeout); |

* 可以自己代理的地址
* 默认地址：<http://cd.10010.com/zop>

开始使用 如：ZOP选号服务

|  |
| --- |
| ZopClient zopClient = new ZopHttpClient("appCode", "hmac", "aes");  KingNumSelectRequest request = new KingNumSelectRequest();  request.setProvinceCode("11");  request.setCityCode("110");  request.setQryType("02");  request.setSearchCategory("1");  request.setGoodsId("982112279603");  request.setAmounts("10");  KingNumSelectResponse response = zopClient.execute(request);  System.out.println("response=" + JSONObject.toJSONString(response)); |

> 更多调用接口可以参考具体接口文档中【SDK请求示例】

# 5、1.5、java okhttp调用示例

### java调用服务示例代码

（仅供参考，调用方可自行按接口规范实现http调用请求）

#### 示例1：（okhttp框架 - 以某选号接口为例）

|  |
| --- |
| public class NumSelectTest {  private static final String APP\_CODE = "3199F3CCA40945D188B4wewqAF30703439";  private static final String URL = "https://cd.10010.com/zop/num/select/v1";  private static final String HMAC = "oz7p1qWg+fOPlpeO3ntaKi1N23sdsdHS7+Sa+YvE/M3mCWpC/vSGYooxf1gVWTFX2OguDJLA==";  private static final String AES = "jBZsdeVr23dhJ6aFDw==";  private final OkHttpClient client = new OkHttpClient();  @Test  public void test1() throws Exception {  String uuid = String.valueOf(UUID.randomUUID());  JSONObject baseReq = new JSONObject();  baseReq.put("appCode",APP\_CODE);  ReqObj obj = new ReqObj<NumStateReqBody>();  ReqHeadBean head = new ReqHeadBean();  head.setTimestamp(Dates.format(new Date(),"yyyy-MM-dd HH:mm:ss.SSS"));  head.setUuid(uuid);  head.setSign(makeSign(head,APP\_CODE));  obj.setHead(head);  NumSelectReqBody body = new NumSelectReqBody();  body.setProvinceCode("11");  body.setCityCode("110");  body.setQryType("02");  body.setGoodsId("981804166984");  body.setSearchCategory("1");  body.setSearchType("02");  body.setSearchValue("12");  body.setAmounts("10");  obj.setBody(body);  //reOjb不需要加密时  //baseReq.put("reqObj",obj);  //reqObj节点需要加密时  baseReq.put("reqObj", SecurityTool.encrypt(AES, JSON.toJSONString(obj)));  System.out.println("请求报文："+ JSON.toJSONString(baseReq));  MediaType json = MediaType.parse("application/json; charset=utf-8");  OkHttpClient eagerClient = client.newBuilder()  .connectTimeout(1000, TimeUnit.MILLISECONDS)  .readTimeout(5000, TimeUnit.MILLISECONDS)  .build();  String myUrl = URL +"?uuid="+uuid;  RequestBody requestBody = RequestBody.create(json, JSON.toJSONString(baseReq));  Request request = new Request.Builder()  .url(myUrl)  .post(requestBody)  .build();  Response response = eagerClient.newCall(request).execute();  System.out.println("返回报文："+ response.body().string());  }  public String makeSign (ReqHeadBean req,String appCode) throws Exception {  StringBuffer sb = new StringBuffer();  //appCode+head节点（除sign节点,字母升序）+hmac密钥  sb.append("appCode").append(appCode)  .append("timestamp").append(req.getTimestamp())  .append("uuid").append(req.getUuid())  .append(HMAC);  return SecurityTool.sign(HMAC,sb.toString());  }  } |

说明1:引用相关jar

|  |
| --- |
| <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>1.2.15</version>  </dependency>  <dependency>  <groupId>com.squareup.okhttp3</groupId>  <artifactId>okhttp</artifactId>  <version>3.9.0</version>  </dependency> |

说明2：ReqObj、ReqHeadBean、NumSelectReqBody、等都是按请求报文规范定义的bean，大家根据实际情况自行按规范定义需要的bean即可；

* 说明3：okhttp框架要求jdk版本1.7+，如果低于此，可以考虑其他http框架； 如http client,见示例2

#### 示例2：（Apache HttpClient 支持jdk1.6 ；以某业务接口为例）

|  |
| --- |
| private static final String APP\_CODE = "11D3260988934769C460B10C54A";  private static final String URL = "http://demo.mall.10010.com:8104/zop/num/select/v1";  private static final String HMAC = "DbAm7868HHtBOZXlzs+JUm1uj8csNt6Lmx4RvBEIc0gNHrq9/JRC6cq/vF+wQHOhhvcBQ==";  private static final String AES = "dANxxnEtTnLA0g==";  @Test  public void test1() throws Exception {  String uuid = String.valueOf(UUID.randomUUID());  JSONObject baseReq = new JSONObject();  baseReq.put("appCode",APP\_CODE);  ReqObj obj = new ReqObj<NumStateReqBody>();  ReqHeadBean head = new ReqHeadBean();  head.setTimestamp("2018-06-27 15:48:124.000");  head.setUuid(uuid);  head.setSign(makeSign(head,APP\_CODE));  obj.setHead(head);  NumSelectReqBody body = new NumSelectReqBody();  body.setProvinceCode("11");  body.setCityCode("110");  body.setQryType("02");  body.setGoodsId("981806112052");  body.setSearchCategory("1");  body.setAmounts("10");  obj.setBody(body);  //reOjb不需要加密时  //baseReq.put("reqObj",obj);  //reqObj节点需要加密时  baseReq.put("reqObj", SecurityTool.encrypt(AES, JSON.toJSONString(obj)));  System.out.println("请求报文："+ baseReq);  CloseableHttpClient httpclient = HttpClients.createDefault();  // 创建httpPost  String myUrl = URL +"?uuid="+uuid;  HttpPost httpPost = new HttpPost(myUrl);  httpPost.setHeader("Accept", "application/json");  httpPost.setHeader("Content-Type", "application/json");  httpPost.setEntity(new StringEntity(baseReq.toString(), "UTF-8"));  CloseableHttpResponse response = httpclient.execute(httpPost);  StatusLine status = response.getStatusLine();  int state = status.getStatusCode();  if (state == HttpStatus.SC\_OK) {  HttpEntity responseEntity = response.getEntity();  String result = EntityUtils.toString(responseEntity,"UTF-8");//此处要加'UTF-8',否则响应中文乱码  System.out.println("返回报文："+ result);  } else {  System.out.println("请求返回:操作失败");  }  }  public String makeSign (ReqHeadBean req,String appCode) throws Exception {  StringBuffer sb = new StringBuffer();  //appCode+head节点（除sign节点,字母升序）+hmac密钥  sb.append("appCode").append(appCode)  .append("timestamp").append(req.getTimestamp())  .append("uuid").append(req.getUuid())  .append(HMAC);  return SecurityTool.sign(HMAC,sb.toString());  } |

* 说明1:引用相关jar

|  |
| --- |
| <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  <version>4.3.6</version>  </dependency> |

* 说明2：ReqObj、ReqHeadBean、BespeakBody、BespeakBean等都是按请求报文规范定义的bean，大家根据实际情况自行按规范定义需要的bean即可；

#### 工具类（重要）

|  |
| --- |
| import sun.misc.BASE64Decoder;  import sun.misc.BASE64Encoder;  import javax.crypto.Cipher;  import javax.crypto.Mac;  import javax.crypto.SecretKey;  import javax.crypto.spec.SecretKeySpec;  /\*\*  \* @author xiaoniu  \*/  public final class SecurityTool {  public static final String ENCODE = "UTF-8";  public static final String CIPHER\_ALGORITHM = "AES";  private static final String KEY\_MAC = "HmacMD5";  public static final String CIPHER\_ALGORITHM\_INS = "AES/ECB/PKCS5Padding";  /\*\*  \* AES加密  \* @param keyStr 密钥  \* @param dataStr 原始数据  \*/  public static String encrypt(String keyStr,String dataStr) throws Exception {  BASE64Decoder decoder = new BASE64Decoder();  SecretKey secretKey = new SecretKeySpec(decoder.decodeBuffer(keyStr), CIPHER\_ALGORITHM);  //Cipher完成加密或解密工作类  Cipher cipher = Cipher.getInstance(CIPHER\_ALGORITHM\_INS);  //对Cipher初始化，解密模式  cipher.init(Cipher.ENCRYPT\_MODE, secretKey);  //加密data  byte[] cipherByte = cipher.doFinal(dataStr.getBytes(ENCODE));  BASE64Encoder base64Encoder = new BASE64Encoder();  return base64Encoder.encode(cipherByte);  }  /\*\*  \* AES解密  \* @param keyStr 密钥  \* @param dataStr 加密数据  \*/  public static String decode(String keyStr,String dataStr) throws Exception {  BASE64Decoder decoder = new BASE64Decoder();  //恢复密钥  SecretKey secretKey = new SecretKeySpec(decoder.decodeBuffer(keyStr), CIPHER\_ALGORITHM);  //Cipher完成加密或解密工作类  Cipher cipher = Cipher.getInstance(CIPHER\_ALGORITHM\_INS);  //对Cipher初始化，解密模式  cipher.init(Cipher.DECRYPT\_MODE, secretKey);  //解密data  byte[] cipherByte = cipher.doFinal(decoder.decodeBuffer(dataStr));  return new String(cipherByte,ENCODE);  }  /\*\*  \* 签名  \* @param key 密钥  \* @param plaintext 原文  \*/  public static String sign(String key,String plaintext) throws Exception {  BASE64Decoder decoder = new BASE64Decoder();  SecretKey secretKey = new SecretKeySpec(decoder.decodeBuffer(key), KEY\_MAC);  Mac mac = Mac.getInstance(secretKey.getAlgorithm());  mac.init(secretKey);  byte[] ret = mac.doFinal(plaintext.getBytes(ENCODE));  BASE64Encoder base64Encoder = new BASE64Encoder();  return base64Encoder.encode(ret);  }  } |

# 6、1.6、python3 示例

### python3调用服务示例代码

（仅供参考，调用方可自行按接口规范实现http调用请求）

#### 示例2：（python3 request post框架 - 以推送数据为例）

|  |
| --- |
| import sys  import requests  import datetime  import base64  import hmac  import hashlib  import json  from Crypto.Cipher import AES  import uuid  class EncryptionHmac(object):  def \_\_init\_\_(self):  pass  @staticmethod  def get\_encrypt\_hmac\_md5(secret, data):  hmacse = hmac.new(secret, bytes(data, 'utf-8'), hashlib.md5).digest()  base = base64.b64encode(hmacse).decode("utf-8")  return base;  @staticmethod  def aes\_encode(data, key):  while len(data) % 16 != 0: # 补足字符串长度为16的倍数  data += (16 - len(data) % 16) \* chr(16 - len(data) % 16)  data = str.encode(data)  aes = AES.new(key, AES.MODE\_ECB) # 初始化加密器  return str(base64.encodebytes(aes.encrypt(data)), encoding='utf8').replace('\n', '') # 加密  if \_\_name\_\_ == '\_\_main\_\_':  try:  APP\_CODE = "12E6C616A3099642A49FB1AE136706915E"  URL = "https://cd.10010.com/zop/link/msgo/terminal/data/push/v1";  HMAC = "12bRLmoJfHuNRrESmF2W/sIRr07CEOPtnaEbp7IewUldQXimWqhmtWQ79e1x2HHRtR6bdoC8fM49LA+SIfvKqEHw=="  aes = "1qD3PJGDmkA8zGiyFeIr+o+g==1"  uuid = str(uuid.uuid1())  timestamp = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')[:23]  headers = {'content-type': 'application/json'}  ##对hmac 做decode  X = base64.b64decode(HMAC)  Y = "appCode"+APP\_CODE+"timestamp"+timestamp+"uuid"+uuid+HMAC  ##签名  Z = EncryptionHmac.get\_encrypt\_hmac\_md5(X,Y)  head = {  "sign": Z,  "timestamp": timestamp,  "uuid": uuid  }  body = {  "goodsList":  [  {"beginTime":"2020-06-17 00:00:00",  "brandCode":"2334",  "brandDesc":"品牌哈哈",  "goodsDesc":"1234567",  "goodsId":"12345644442255666",  "goodsState":"1",  "modelCode":"343",  "modelDesc":"型号222",  "operateType":"C",  "state":"1",  "type":"1"}  ]  }  req = {  "head":head,  "body":body  }  ##对aes做decode  aesdecode=base64.b64decode(aes)  print("aesdecode",aesdecode)  ##对称加密  reqObj = EncryptionHmac.aes\_encode(json.dumps(req),aesdecode)  requestData = {  "appCode":APP\_CODE,  "reqObj":reqObj  }  print("请求url："+URL+"----请求参数："+str(requestData))  ret = requests.post(URL, json=requestData, headers=headers)  if ret.status\_code == 200:  ##result = json.loads(ret.text)  print("响应"+ret.text)  except:  print("Unexpected error:", sys.exc\_info())  raise |

# 7、1.7、PHP7 示例

## PHP7 调用服务示例代码

>仅供参考，调用方可自行按接口规范实现 http 调用请求

在线调试工具：<https://code.y444.cn/php>

示例3：请求收货地址省市区和号码省市信息查询

|  |
| --- |
| <?php  // zop 请求地址  const ZOP\_URL = 'https://cd.10010.com/zop/';  // 定义加密参数 商城测提供  const ZOP\_KEY = [  'appCode' => 'appCode',  'HMACMD5' => 'HMACMD5',  'AES' => 'AES'  ];  /\*\*  \* uuid 唯一序列码  \* @param string $prefix  \* @return string  \*/  function uuid($prefix = '')  {  $chars = md5(uniqid(mt\_rand(), true));  $uuid = substr($chars, 0, 8) . '-';  $uuid .= substr($chars, 8, 4) . '-';  $uuid .= substr($chars, 12, 4) . '-';  $uuid .= substr($chars, 16, 4) . '-';  $uuid .= substr($chars, 20, 12);  return $prefix . $uuid;  }  //获取当前时间毫秒  function msectime()  {  list($msec, $sec) = explode(' ', microtime());  $msectime = (float)sprintf('%.0f', (floatval($msec) + floatval($sec)) \* 1000);  return $msectime;  }  // 时间戳  function getTimestamp()  {  date\_default\_timezone\_set('PRC');  $time = msectime();  return date('Y-m-d H:i:s.') . substr($time, -3);  }  // 生成签名  function genSign($uuid, $timestamp)  {  $dataStr = 'appCode' .ZOP\_KEY['appCode'] . 'timestamp' . $timestamp . 'uuid' . $uuid . ZOP\_KEY['HMACMD5'];  return sign(ZOP\_KEY['HMACMD5'], $dataStr);  }  /\*\*  \* 签名  \* @param key 密钥 平台侧提供的HmacMD5  \* @param dataStr appCode+appCode值+timestamp+timestamp值\*+uuid+uuid值+HmacMD5密钥值  \*/  function sign($key, $dataStr)  {  return base64\_encode(hash\_hmac("md5", $dataStr, base64\_decode($key), true));  }  /\*\*  \*AES加密  \*/  function encrypt($data, $key)  {  $data = openssl\_encrypt($data, 'aes-128-ecb', base64\_decode($key), OPENSSL\_RAW\_DATA);  return base64\_encode($data);  }  function sendPost($url, $post\_data)  {  $opt\_data = json\_encode($post\_data);  $header = array();  $header[] = 'Accept:application/json';  $header[] = 'Content-Type:application/json;charset=utf-8';  $curl = curl\_init(); //初始化  curl\_setopt($curl, CURLOPT\_URL, $url); //设置url  curl\_setopt($curl, CURLOPT\_HTTPHEADER, $header);  curl\_setopt($curl, CURLOPT\_POSTFIELDS, $opt\_data);  //curl\_setopt($curl,CURLOPT\_POST,1);  $result = curl\_exec($curl);  if($result === false){  echo curl\_errno($curl);  exit();  }  print\_r($result);  curl\_close($curl);  }  // 主方法  function test()  {  // 请求收货地址省市区和号码省市信息查询  // https://www.showdoc.com.cn/liantongorder?page\_id=6116529029632075  // 准备参数  $url = ZOP\_URL . '/link/king/postInfo/qry';  $body = ['provinceCode' => '11'];  // uuid  $uuid = uuid();  // 生成时间戳  $timestamp = getTimestamp();  $head = [  'sign' => genSign($uuid, $timestamp),  'timestamp' => $timestamp,  'uuid' => $uuid  ];  $reqObj = [  'head' => $head,  'body' => $body  ];  $reqObj = encrypt(json\_encode($reqObj),ZOP\_KEY['AES']);  $reqParam = [  'appCode' => ZOP\_KEY['appCode'],  'reqObj' => $reqObj  ];  sendPost($url, $reqParam);  }  test();  ?> |

响应报文

|  |
| --- |
| {"rspCode":"0000","rspDesc":"操作成功","uuid":"9b6d6fa4-6ff8-f0de-6baa-91b37de9e339","body":[{"postCityCode":"110100","postDistrictCode":"110101","postProvinceName":"北京","postDistrictName":"东城区","postProvinceCode":"110000","numProvinceCode":"11","numProvinceName":"北京","postCityName":"北京市","numCityName":"北京","numCityCode":"110"}....,{"postCityCode":"110100","postDistrictCode":"110229","postProvinceName":"北京","postDistrictName":"延庆县","postProvinceCode":"110000","numProvinceCode":"11","numProvinceName":"北京","postCityName":"北京市","numCityName":"北京","numCityCode":"110"}]} |

# 8、1.8、响应报文

* 根节点

| **节点** | **父节点** | **类型** | **长度** | **约束** | **注释** |
| --- | --- | --- | --- | --- | --- |
| rspCode | - | string | V4 | 1 | 响应编码 |
| rspDesc | - | string | V200 | 1 | 响应描述 |
| body | - | json | \* | ？ | 响应业务数据，具体见业务接口 |

#### 系统响应编码

| **编码** | **说明** |
| --- | --- |
| 0000 | 成功 |
| E1 | 请求报文不符合规范 |
| E2 | 没有此应用权限 |
| E3 | 没有此调用权限，ip问题 |
| E4 | 请求参数解密失败 |
| E5 | 请求数据不合法,签名无效 |
| E6 | head节点中的时间戳过期 |
| E7 | 内部错误 |
| E8 | 内部错误 |