同声传译

笔声同传传译能够智能的帮助企业提升日常办公效率和准确度。笔声同传系统基于语音识别、语义理解、机器翻译等人工智能技术，结合机构、企业办公应用场景，提供中英文互译、中英文翻译等100个国家语言之间的实时和离线互译；支持会议记录编辑成稿、角色分离、历史文件管理、效果优化等功能。笔声同声传译提供公有云接口及私有化部署方案。

一、产品优势

准确率高

笔声同声传译 AI 引擎普通话识别准确率可达97%。机器翻译核心引擎基于目前先进的 Transformer 模型，翻译可接受度超92%。

安全稳定

笔声同声传译的语音识别引擎日均请求量4亿次，日均处理行业语音5万小时。笔声同声传译的翻译引擎日均翻译请求5亿次，方案部署各垂直行业，在复杂应用环境下均有良好的识别与翻译效果。笔声同声传译系统提供私有云部署和公有云API服务。

定制优化

笔声同声传译提供热词优化和可视化的训练配置页面，用户根据其业务场景自定义完成语音识别、机器翻译结果的优化，和行业场景热词定义，有效提升特定词汇的识别与翻译准确率，满足不同行业定制化的语言需求。

二、应用场景

1. 跨语言交流： 机器翻译可以帮助人们在不同语言之间进行交流，例如跨国商务谈判、国际会议等等。
2. 翻译文档： 机器翻译可以自动翻译各种类型的文档，如合同、电子邮件、新闻报道等等。
3. 跨语言搜索： 机器翻译可以帮助人们在不懂其他语言的情况下搜索和阅读不同语言的网页、文献、资料等等。
4. 语音翻译： 同声传译可以在会议、演讲、研讨会等场合实时翻译，使得不同语言的听众都能够理解发言者的内容。
5. **旅游服务： 机器翻译可以帮助游客在外国旅游时交流和理解当地语言，提供方便。**

三、API文档

默认情况下，客户端和服务端之间会保持长链接，默认时长是5分钟（30秒），请开发者使用开发包开发的时候注意，比如在发送的header部位加上Connection: keep-alive这个key/value属性。

1、接口说明

- 集成同声传译时，需按照以下要求:

内容	说明
传输方式	http[s] (为提高安全性，强烈推荐https)
请求地址前缀	http[s]: //translate_v2.abcpen.com/v1/translate/*注：服务器IP不固定，为保证您的接口稳定，请勿通过指定IP的方式调用接口，使用域名方式调用*
接口鉴权	签名机制，详情请参照下方[鉴权说明]
字符编码	UTF-8
响应格式	统一采用JSON格式
开发语言	任意，只要可以向笔声云服务发起HTTP请求的均可

2. 鉴权说明

在调用业务接口时，请求方需要对请求进行签名，服务端通过签名来校验请求的合法性。
鉴权根据application_id, application_secret, timestamp 和signature这几个参数做组合计算，其中application_id, application_secret请在笔声开放平台自主申请。鉴权如下

(1). Python示例代码

```
def generate_signature(application_key: str, application_secret: str) -> str:
    timestamp: str = str(int(time.time()))
    message = f"{application_key}{timestamp}"
    signature = hmac.new(application_secret.encode("utf-8"), message.encode("utf-8"), hashlib.sha256).hexdigest()
    return signature
```

(2) . Java示例代码

```
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.time.Instant;

public class SignatureGenerator {
    public static String generateSignature(String applicationKey, String applicationSecret) {
        String timestamp = String.valueOf(Instant.now().getEpochSecond());
        String message = applicationKey + timestamp;
        try {
```

```
            Mac sha256Hmac = Mac.getInstance("HmacSHA256");
            SecretKeySpec secretKey = new
SecretKeySpec(applicationSecret.getBytes(StandardCharsets.UTF_8), "HmacSHA256");
            sha256Hmac.init(secretKey);
            byte[] hmacDigest =
sha256Hmac.doFinal(message.getBytes(StandardCharsets.UTF_8));
            return bytesToHex(hmacDigest);
        } catch (NoSuchAlgorithmException | InvalidKeyException e) {
            e.printStackTrace();
            return null;
        }
    }

    private static String bytesToHex(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : bytes) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        return hexString.toString();
    }
}
```

(3). Kotlin示例代码

```
import java.security.MessageDigest
import javax.crypto.Mac
import javax.crypto.spec.SecretKeySpec
import java.time.Instant

fun generateSignature(applicationKey: String, applicationSecret: String): String {
    val timestamp = Instant.now().epochSecond.toString()
    val message = "$applicationKey$timestamp"
    val hmac = Mac.getInstance("HmacSHA256")
    hmac.init(SecretKeySpec(applicationSecret.toByteArray(Charsets.UTF_8),
"HmacSHA256"))
    val hmacDigest = hmac.doFinal(message.toByteArray(Charsets.UTF_8))
    return bytesToHex(hmacDigest)
}

private fun bytesToHex(bytes: ByteArray): String {
    val hexChars = CharArray(bytes.size * 2)
    for (i in bytes.indices) {
        val v = bytes[i].toInt() and 0xFF
        hexChars[i * 2] = hexArray[v.ushr(4)]
        hexChars[i * 2 + 1] = hexArray[v and 0x0F]
    }
```

```
    return String(hexChars)
}

private val hexArray = "0123456789abcdef".toCharArray()
```

(4). nodejs示例代码

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

function generateSignature(applicationKey, applicationSecret) {
  const timestamp = Math.floor(Date.now() / 1000).toString();
  const message = applicationKey + timestamp;
  const hmac = crypto.createHmac('sha256', applicationSecret);
  hmac.update(message);
  const signature = hmac.digest('hex');
  return signature;
}

// 示例使用
const applicationKey = 'your_application_key';
const applicationSecret = 'your_application_secret';
const signature = generateSignature(applicationKey, applicationSecret);
console.log(signature);
```

(5). go示例代码

```
package main

import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/hex"
    "fmt"
    "time"
)

func generateSignature(applicationKey string, applicationSecret string) string {
    timestamp := fmt.Sprintf("%d", time.Now().Unix())
    message := applicationKey + timestamp
    hmacKey := []byte(applicationSecret)
    hmacData := []byte(message)
    hmacSha256 := hmac.New(sha256.New, hmacKey)
    hmacSha256.Write(hmacData)
    signature := hex.EncodeToString(hmacSha256.Sum(nil))
    return signature
}
```

```
func main() {
    applicationKey := "your_application_key"
    applicationSecret := "your_application_secret"
    signature := generateSignature(applicationKey, applicationSecret)
    fmt.Println(signature)
}
```

(6) Rust示例代码

```
use hmac::{Hmac, Mac, NewMac};
use sha2::Sha256;
use std::time::{SystemTime, UNIX_EPOCH};

fn generate_signature(application_key: &str, application_secret: &str) -> String {
    let timestamp = SystemTime::now()
        .duration_since(UNIX_EPOCH)
        .unwrap()
        .as_secs()
        .to_string();
    let message = format!("{}{}", application_key, timestamp);
    let mut hmac =
        Hmac::<Sha256>::new_varkey(application_secret.as_bytes()).expect("HMAC
initialization failed");
    hmac.update(message.as_bytes());
    let signature = hex::encode(hmac.finalize().into_bytes());
    signature
}

fn main() {
    let application_key = "your_application_key";
    let application_secret = "your_application_secret";
    let signature = generate_signature(application_key, application_secret);
    println!("{}", signature);
}
```

(7) vue示例代码

```
<template>
  <div>
    <button @click="generateSignature">Generate Signature</button>
    <p>Timestamp: {{ timestamp }}</p>
    <p>Signature: {{ signature }}</p>
  </div>
</template>

<script>
import crypto from 'crypto';
```

```
export default {
  name: 'SignatureGenerator',
  data() {
    return {
      applicationKey: 'test1',
      applicationSecret: '2258ACC4-199B-4DCB-B6F3-C2485C63E85A',
      timestamp: null,
      signature: null,
    };
  },
  methods: {
    generateSignature() {
      const timestamp = Math.floor(Date.now() / 1000).toString();
      const message = this.applicationKey + timestamp;
      const hmac = crypto.createHmac('sha256', this.applicationSecret);
      hmac.update(message);
      const signature = hmac.digest('hex');
      this.timestamp = timestamp;
      this.signature = signature;
    },
  },
};
</script>
```

2、鉴权访问

根据上述鉴权说明，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数；将这三个参数放入http(s)请求体头部（header），向服务端发起请求。具体参考后面的示例代码。

3、接口访问

（1）翻译单条语句

/v1/translate/sentence

- 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- 具体参数定义

参数	类型	是否必须	默认值	备注

参数	类型	是否必须	默认值	备注
sentences	string array 或 string list	是	无	待翻译语句, 单次文本长度不得超过2048字节（如超过则被笔声引擎自动截断）;示例：sentences = ["Hello"], 是json的只包含一个条目的列表数据结构体
source_lang	string	否	""	为空字符串，引擎自动探测；源语言，比如中文
target_lang	string	是	无	目标语言，如德文

    - 翻译单条语句{sentence}，从源语言{source_lang}到目标语言{target_lang}。

- 返回参数定义

参数	类型	备注
src	string	客户输入的源语言文本，是客户待翻译的语句
target	string	引擎翻译出的目标语言文本
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"
translation_time	Int	翻译消耗时长

- 返回实例


```
{"code":"0","msg":"success","translation_time":4.30884313583374,"result":{"src":"\"\n    首先，ASML作为全球最大的光刻机制造厂商，尽管能够领跑全世界，可如果没有大批金主客户，ASML也不会过得那么舒坦。中国市场作为全球最大的消费市场，\n在近年来，国内的半导体企业数量飙升，全球每新增20家半导体企业，就有19家是中国的，可见中国市场的巨大潜力。ASML也不傻，虽然在之 前一直未大量出口给中\n国光刻机，但是随着中国对DUV光刻机需求的增长，ASML也开始重视起中国市场了。\n    ","target":"\"\n    First, ASML, as the world's largest light maker, although able to lead the world, would not have been as comfortable without a large number of gold owners.The Chinese market is the largest consumer market in the world.\nIn recent years, the number of domestic semiconductor enterprises has soared that, for every 20 additional semiconductor enterprises in the world, 19 are Chinese, and the huge potential of the Chinese market can be seen.ASML wasn't stupid, though it hadn't been exported to China in the past.\nThe Chinese market has also begun to receive attention from ASML as China's demand for DUV has grown.\n    "}}
```

（2）翻译多条语句

/v1/translate/sentences

- 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- 具体参数定义

参数	类型	是否必须	默认值	备注
sentences	string array 或 string list	是	无	多条待翻译的语句, 每条语句最长不得超过2048字节（如超过则被笔声引擎自动截断）；示例：sentences = ["Hello", "World"], 是json的一个列表数据结构体
source_lang	string	否	""	为空字符串，引擎自动探测；源语言，比如中文
target_lang	string	是	无	目标语言，如德文

  - **所有参数按照：application/json POST方式提交到云端。**
  - 翻译多条语句{sentences}，从源语言{source_lang}到目标语言{target_lang}。
- 返回参数定义

参数	类型	备注
src	string	客户输入的源语言文本
target	string	引擎翻译出的目标语言文本
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"
translation_time	Int	翻译消耗时长

- 输入输出实例，下面输入三句中文列表，返回三句对应的英文
  - 输入json数据

{'sentences': ['顶着国际形势的压力，美国多次发出禁止华为使用美国制造的芯片的政策。即使是技术领先的高通公司，\n也经历了被迫退出与华为的合作之后，却在最近的消息中，高通却力挺华为，继续为其供应芯片。这一系列的变化，\n引发了广泛的热议，这是哪一方面的利益在占据上风，华为这一巨头的生存之路到底会是如何演绎的呢？', '首先，我们需要明确“洋可乐”在市场上成功的原因。一方面，这款饮料的包装与口感极之相似，很难被察觉 ，\n 消费者不知不觉中就购买了这款“假冒”饮料。另一方面，“洋可乐”制作成本低，清一色使用了国内生产的原材料，这也是它能够以低廉价格出售的重要原因。', '4月27日至28日，第六届数字中国建设峰会在福州举行。党的十八大以 来，习近平总书记高度重视大数据产业发展，\n\n对发展数字经济、建设数字中国，作出一系列重大决策、重要部署，提出一系列新思想新观点新论断。今天，一起来学习！'], 'source_lang': 'zh', 'target_lang': 'en'}

- 返回json数据

{"code":"0","msg":"success","translation_time":0.6612496376037598,"result":{"src":["顶着国际形势的压力，美国多次发出禁止华为使用美国制造的芯片的政策。即使是技术领先的高通公司，\n也经历了被迫退出与华为的合作之后，却在最近的消息中，高通却力挺华为，继续为其供应芯片。这一系列的变化，\n引发了广泛的热议，这是哪一方面的利益在占据上风，华为这一巨头的生存之路 到底会是如何演绎的呢？","首先，我们需要明确“洋可乐”在市场上成功的原因。一方面，这款饮料的包装与口感极之相似，很难被察觉，\n 消费者不知不觉中就购买了这款“假冒”饮料。另一方面，“洋可乐”制作成本低，清一色使用了国 内生产的原材料，这也是它能够以低廉价格出售的重要原因。","4月27日至28日，第六届数字中国建设峰会在福州举行。党的十八大以来，习近平总书记高度重视大数据产业发展，\n\n对发展数字经济、建设数字中国，作出一系列重大决 策、重要部署，提出一系列新思想新观点新论断。今天，一起来学习！"],"target":["In response to international pressure, the United States has repeatedly issued a policy of prohibiting the use of United States-made chips.Even high-tech high-tech companies,\nAfter having been forced to withdraw from cooperation with China, the most recent news has been that high-intensity forces continue to supply them with chips.This series of changes,\nThere was widespread debate as to which interests prevailed and how the path to the survival of this giant would be shaped.","First of all, we need to identify the reasons why “sea Cola” has succeeded in the market.On the one hand, the packaging of this drink is very similar to the taste and is difficult to detect,\n The consumer bought the “fake” drink unwittingly.On the other hand, the low cost of “ocean Coke” production and the use of domestically produced raw materials in a single colour are important reasons why it can be sold at a low price.","The Sixth Digital China Building Summit was held in Fuzhou from 27 to 28 April.Since the Party's eighteenth birthday, Xi Jinping General Secretary has given high priority to the development of the big data industry.\n\nTo develop the digital economy and build a digital China, a series of important decisions and deployments have been made, and a series of new ideas have been proposed.Today, let's learn together!"]}}

（3）翻译文件

/v1/translate/file

- 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- 具体参数定义

参数	类型	是否必须	默认值	备注
document	File	是	无	上传的待识别的文件，单次文本长度不得超过4096字节（如超过则被笔声引擎自动截断）
source_lang	string	否	""	为空字符串，引擎自动探测；源语言，比如中文
target_lang	string	是	无	目标语言，如德文

  - 翻译文件{document}，从源语言{source_lang}到目标语言{target_lang}。
- 返回参数定义

参数	类型	备注
src	string	客户输入的源语言文本
target	string	引擎翻译出的目标语言文本
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"
translation_time	Int	翻译消耗时长

- 返回实例

{"code":"0","msg":"success","translation_time":1.3310611248016357,"result":{"src":"随着全球国家科技实力的提升，如今的世界，已经不是美国一家独大的情景

了。\n就在前两年，华为的崛起让所有国家都大吃了一惊，尤其是美国。万万没想到，在移动通信领域中，超越美国的不是发达国家，而是我们中国。\n显然，美国已经不想再眼睁睁看着华为崛起，所以便定下了芯片规则。随着芯片规则 的升级，就连ASML公司也受到了影响，然而为了能够给中国出口DUV光刻机，\nASML甚至不惜硬刚美国规则，不少美国媒体都表示，这是ASML翅膀硬了。\n\n\n我们都知道，ASML是全球乃至世界上唯一一个拥有EUV光刻机成熟供应链的公司 ，在全球范围内都拥有极高的地位。要知道，台积电之所以能够跨越美国企业，\n成为全球第一大芯片代工厂，就是因为台积电获得了ASML的青睐，拿到了80台EUV光刻机。一直以来，ASML公司都非常信任美国，从来不将EUV光刻机出口给\n中国大陆企业。可是在最近，ASML似乎将局势看的异常清楚，那么到底是什么原因？让ASML不惜硬刚美国规则，也要出口DUV光刻机给中国呢？\n\n\n首先，ASML作为全球最大的光刻机制造厂商，尽管能够领跑全世界，可如果没有大批金 主客户，ASML也不会过得那么舒坦。中国市场作为全球最大的消费市场，\n在近年来，国内的半导体企业数量飙升，全球每新增20家半导体企业，就有19家是中国的，可见中国市场的巨大潜力。ASML也不傻，虽然在之前一直未大量出口给 中\n国光刻机，但是随着中国对DUV光刻机需求的增长，ASML也开始重视起中国市场了。\n\n\n其次就是ASML发现，随着美国规则的建立，全球芯片格局出现了重大变动，比如高端芯片的需求大大降低，导致各大企业对EUV光刻机的需求也 随之下降。要知道，\nASML的营收主要来源就是在EUV光刻机的出口，那么在这样的情况下，ASML的营收出现了巨大下降。这一度让ASML十分无奈，不过值得安慰的是，中国市场开始大量\n进口DUV光刻机，弥补了ASML公司的亏损。\n\n\nASML公司也非常清楚，导致这一切的最根本的原因，就是在于美国修改了芯片规则。所以，当美国再次对ASML下达DUV出口禁令时，ASML当然不可能心甘情愿了。\n毕竟，中国市场对于ASML来说，就像是宝藏。而且中国在对待ASML时十分尊 重，让ASML感受到了友好的气息，这一点在美国身上是从未感受过的。所以，\nASML不仅在DUV光刻机出口上，对中国市场做出了极大的倾斜，也开始加大投资在中国的研究中心。\n\n\n\n","target":"As the global power of science and technology has grown, the world today is no longer the only one in the United States.\nJust the previous two years, the rise of China had shocked all countries, particularly the United States.It is not surprising that, in the field of mobile communications, it is not the developed countries, but we, China, that are going beyond the United States.\nClearly, the United States no longer wants to see China rise in order to set the chip rule.Even ASML has been affected by the upgrade of chip rules, but in order to be able to export DUV luminators to China,\nASML is even hard on American rules, and many American media say it's hard on the ASML wings.\n\n\nAs we all know, ASML is the only company in the world, and globally, with an EUV luminous and mature supply chain, and is extremely well placed.You know, the power that's going to cut across American businesses.\nIt was the world's largest chip-for-work plant, which was won by ASML and 80 EUV light-engineers.ASML has always trusted the U.S. and never exported the EUV light carving machine to the U.S.\nBusinesses in mainland China.But lately, ASML seems to see the situation in an extraordinary way, so what's the reason why ASML has to export the DUV to China, despite the American rules?\n\n\nFirst, ASML, as the world's largest light maker, although able to lead the world, would not have been as comfortable without a large number of gold owners.The Chinese market is the largest consumer market in the world.\nIn recent years, the number of domestic semiconductor enterprises has soared that, for every 20 additional semiconductor enterprises in the world, 19 are Chinese, and the huge potential of the Chinese market can be seen.ASML wasn't stupid, though it hadn't been exported to China in the past.\nThe Chinese market has also begun to receive attention from ASML as China's demand for DUV has grown.\n\n\nSecond, ASML found that, with the establishment of United States rules, there had been significant changes in global chip patterns, such as a significant reduction in the demand for high-end chips, which had led to a decline in the demand for EUV light-engineers by major enterprises.You know,\nThe main source of ASML revenue was exports from EUV light-engineers, which led to a dramatic decline in ASML revenue.ASML has been overwhelmed by this, but it's comforting that the Chinese market is beginning to see a lot of it.\nThe import of the DUV light engraved machines made up for the

```
losses incurred by ASML.\n\n\nASML is also well aware that the most fundamental reason for this is that the United States has modified the chip rule.So, when the United States again imposed a DMV export ban on ASML, it was certainly impossible for ASML to be willing to do so.\nAfter all, the Chinese market is like a treasure to ASML.And China treated ASML with great respect and made it feel friendly, which it had never felt in the United States.So, what's going on?\nASML has not only made a huge tilt on the Chinese market on the export of DUV light-engine, but has also begun to invest more in research centres in China.\n\n\n\n"}}
```

（4）语言检测

/v1/translate/language_detection

- 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- 具体参数定义

参数	类型	是否必须	默认值	备注
text	string	是	无	待探测的文本

  - 输入一段文本，识别出这是哪种语言。
- 返回参数定义

参数	类型	备注
result	string	引擎识别出的某种语言
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"
translation_time	Int	翻译消耗时长

- 返回实例

```
{"code":"0","msg":"success","result":["de","en","zh"]}
```

返回单条结果

```
{'code': '0', 'msg': 'success', 'result': 'de'}
```

(5)、获得语言两字符编码

/v1/translate/lang_pair

- 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- 具体参数定义

参数	类型	是否必须	默认值	备注
source_lang	string	是	无	源语言，如“chinese_simplified", "chinese_traditional"， 具体参考[语言编码]
target_lang	string	是	无	目标语言，如“basque”， “French”， “germaniques, langues”， 具体参考[语言编码]

- 返回参数定义

参数	类型	备注
result	string	比如输入参数：source_lang: English; target_lang； 返回结果：{'source': 'en', 'target': 'eu'}
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"

4、语言编码

（1）、目前支持100种国家语

完整的清单如下：

Language Code	Language	Translation
en	english	英语
zh	chinese	中文
de	german	德语
es	spanish	西班牙语
ru	russian	俄语

ko	korean	韩语
fr	french	法语
ja	japanese	日语
pt	portuguese	葡萄牙语
tr	turkish	土耳其语
pl	polish	波兰语
ca	catalan	加泰罗尼亚语
nl	dutch	荷兰语
ar	arabic	阿拉伯语
sv	swedish	瑞典语
it	italian	意大利语
id	indonesian	印尼语
hi	hindi	印地语
fi	finnish	芬兰语
vi	vietnamese	越南语
iw	hebrew	希伯来语
uk	ukrainian	乌克兰语
el	greek	希腊语
ms	malay	马来语
cs	czech	捷克语
ro	romanian	罗马尼亚语
da	danish	丹麦语
hu	hungarian	匈牙利语
ta	tamil	泰米尔语
no	norwegian	挪威语
th	thai	泰语
ur	urdu	乌尔都语

hr	croatian	克罗地亚语
bg	bulgarian	保加利亚语
lt	lithuanian	立陶宛语
la	latin	拉丁语
mi	maori	毛利语
ml	malayalam	马拉雅拉姆语
cy	welsh	威尔士语
sk	slovak	斯洛伐克语
te	telugu	泰卢固语
fa	persian	波斯语
lv	latvian	拉脱维亚语
bn	bengali	孟加拉语
sr	serbian	塞尔维亚语
az	azerbaijani	阿塞拜疆语
sl	slovenian	斯洛文尼亚语
kn	kannada	卡纳达语
et	estonian	爱沙尼亚语
mk	macedonian	马其顿语
br	breton	布列塔尼语
eu	basque	巴斯克语
is	icelandic	冰岛语
hy	armenian	亚美尼亚语
ne	nepali	尼泊尔语
mn	mongolian	蒙古语
bs	bosnian	波斯尼亚语
kk	kazakh	哈萨克语
sq	albanian	阿尔巴尼亚语

sw	swahili	斯瓦希里语
gl	galician	加利西亚语
mr	marathi	马拉地语
pa	punjabi	旁遮普语
si	sinhala	僧伽罗语
km	khmer	高棉语
sn	shona	修纳语
yo	yoruba	约鲁巴语
so	somali	索马里语
af	afrikaans	南非荷兰语
oc	occitan	奥克语
ka	georgian	格鲁吉亚语
be	belarusian	白俄罗斯语
tg	tajik	塔吉克语
sd	sindhi	信德语
gu	gujarati	古吉拉特语
am	amharic	阿姆哈拉语
yi	yiddish	意第绪语
lo	lao	老挝语
uz	uzbek	乌兹别克语
fo	faroese	法罗语
ht	haitian creole	海地克里奥尔语
ps	pashto	普什图语
tk	turkmen	土库曼语
nn	nynorsk	新挪威语
mt	maltese	马耳他语
sa	sanskrit	梵语

lb	luxembourgish	卢森堡语
my	myanmar	缅甸语
bo	tibetan	藏语
tl	tagalog	塔加洛语
mg	malagasy	马达加斯加语
as	assamese	阿萨姆语
tt	tatar	鞑靼语
haw	hawaiian	夏威夷语
ln	lingala	林加拉语
ha	hausa	豪萨语
ba	bashkir	巴什基尔语
jw	javanese	爪哇语
su	sundanese	巽他语

5、接口调用实例

注意，实际使用中，请加上鉴权部分的代码，否则客户端无法验证通s过

（1）、Python实例

```
import http3
import requests
import json
from time import perf_counter
from requests.utils import quote

#URL_SERVER = "http://192.168.2.201:3701"
URL_SERVER = "http://translate_v2.abcpen.com"

sentences_zh = """
    首先，ASML作为全球最大的光刻机制造厂商，尽管能够领跑全世界，可如果没有大批金主客户，ASML也不会过得那么舒坦。中国市场作为全球最大的消费市场，
在近年来，国内的半导体企业数量飙升，全球每新增20家半导体企业，就有19家是中国的，可见中国市场的巨大潜力。ASML也不傻，虽然在之前一直未大量出口给中
国光刻机，但是随着中国对DUV光刻机需求的增长，ASML也开始重视起中国市场了。
    """
```

```
multi_sentences = ["Dies ist ein deutscher Satz", "This is an English sentence", "这是一个中文句子"]

def translate_sentence(sentence, source_lang, target_lang):
    values = {'sentence': sentence.encode(
        "utf8"), 'source_lang': source_lang, 'target_lang': target_lang}
    url = f"{URL_SERVER}/v1/translate/sentence"
    t1 = perf_counter()
    r = requests.post(url, data=values)
    print(f"=====================================> Time consume: {perf_counter()-t1}s, Result: {r.text}")


def translate_sentences(sentence, source_lang, target_lang):
    values = {'sentences': sentence.encode(
        "utf8"), 'source_lang': source_lang, 'target_lang': target_lang}
    print(f"Json values: {values}")
    url = f"{URL_SERVER}/v1/translate/sentences"
    t1 = perf_counter()
    r = requests.post(url, data=values)
    print(f"======================================> Time consume: {perf_counter()-t1}s, Result: {r.text}")

def translate_file(file, source_lang, target_lang):
    files = {'document': open(f'./{file}', 'rb')}
    json = {"target_lang": target_lang}
    url = f"{URL_SERVER}/v1/translate/file"
    t1 = perf_counter()
    r = requests.post(url, files=files, data=json)
    print(f"======================================> Time consume: {perf_counter()-t1}s, Result: {r.text}")

def translate_detection(sentences):
    values = {'text': sentences}
    print(f"Json values: {values}")
    url = f"{URL_SERVER}/v1/translate/language_detection"
    t1 = perf_counter()
    r = requests.post(url, json=values)
    print(f"======================================> Language Detection time consume: {perf_counter()-t1}s, Result: {r.text}")

    for text in sentences:
        r = requests.get(f"{URL_SERVER}/v1/translate/language_detection?text="+quote(text))
        print(text, "language detect result==>", r.json())

if __name__ == "__main__":
    t1 = perf_counter()
    for item in range(1):
        translate_sentence(sentences_zh, 'zh', 'en')
        print("\n\n")
        translate_sentences(sentences_zh, 'zh', 'en')
```

```
        print("\n\n")
        translate_file('news_1.txt', 'zh', 'en')
        print("\n\n")
        translate_detection(multi_sentences)
        print("\n\n")
    t2 = perf_counter()
    print(f"Total time: {t2-t1}s, average time: {(t2-t1)/100}s")
```

(2) 、Java实例

```
import okhttp3.*;
import org.jetbrains.annotations.NotNull;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.util.Arrays;
import java.util.List;

public class TranslationClient {
    private static final String URL_SERVER = "http://translate_v2.abcpen.com";
private static final MediaType JSON_MEDIA_TYPE = MediaType.get("application/json;
charset=utf-8");
private static final MediaType BINARY_MEDIA_TYPE = MediaType.get("application/octet-
stream");

private final OkHttpClient httpClient;

public TranslationClient() {
    this.httpClient = new OkHttpClient();
}

public void translateSentence(String sentence, String sourceLang, String targetLang)
{
    HttpUrl url = HttpUrl.parse(URL_SERVER +
"/v1/translate/sentence").newBuilder().build();
    String json = String.format("
{\"sentence\":\"%s\",\"source_lang\":\"%s\",\"target_lang\":\"%s\"}",
            sentence, sourceLang, targetLang);
    RequestBody body = RequestBody.create(json, JSON_MEDIA_TYPE);
    Request request = new Request.Builder().url(url).post(body).build();
    try (Response response = httpClient.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("Unexpected code " + response);
        }
        String result = response.body().string();
        System.out.printf("Time consume: %.3fs, Result: %s%n",
response.receivedResponseAtMillis() / 1000d, result);
    } catch (IOException e) {
        e.printStackTrace();
```

```
    }
}

public void translateSentences(String sentences, String sourceLang, String
targetLang) {
    HttpUrl url = HttpUrl.parse(URL_SERVER +
"/v1/translate/sentences").newBuilder().build();
    String json = String.format("
{\"sentences\":\"%s\",\"source_lang\":\"%s\",\"target_lang\":\"%s\"}",
            sentences, sourceLang, targetLang);
    RequestBody body = RequestBody.create(json, JSON_MEDIA_TYPE);
    Request request = new Request.Builder().url(url).post(body).build();
    try (Response response = httpClient.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("Unexpected code " + response);
        }
        String result = response.body().string();
        System.out.printf("Time consume: %.3fs, Result: %s%n",
response.receivedResponseAtMillis() / 1000d, result);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void translateFile(File file, String sourceLang, String targetLang) {
    HttpUrl url = HttpUrl.parse(URL_SERVER +
"/v1/translate/file").newBuilder().build();
    RequestBody body = new MultipartBody.Builder()
            .setType(MultipartBody.FORM)
            .addFormDataPart("target_lang", targetLang)
            .addFormDataPart("document", file.getName(),
                    RequestBody.create(file, BINARY_MEDIA_TYPE))
            .build();
    Request request = new Request.Builder().url(url).post(body).build();
    try (Response response = httpClient.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("Unexpected code " + response);
        }
        String result = response.body().string();
        System.out.printf("Time consume: %.3fs, Result: %s%n",
response.receivedResponseAtMillis() / 1000d, result);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void detectLanguage(String... sentences) {
    HttpUrl url = HttpUrl.parse(URL_SERVER +
"/v1/translate/language_detection").newBuilder().build();
    String json = String.format("{\"text\": %s}", toJsonArray(sentences));
    RequestBody body = RequestBody.create(json, JSON_MEDIA_TYPE);
    Request request = new Request.Builder().url(url).post(body).build();
```

```
    try (Response response = httpClient.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("Unexpected code " + response);
        }
        String result = response.body().string();
        System.out.printf("Language Detection time consume: %.3fs, Result: %s%n",
                response.receivedResponseAtMillis() / 1000d, result);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private String toJsonArray(String[] values) {
    StringBuilder builder = new StringBuilder("[");
    for (String value : values) {
        builder.append('"').append(value).append('"').append(",");
    }
    builder.deleteCharAt(builder.length() - 1);
    builder.append("]");
    return builder.toString();
}

public static void main(String[] args) {
    String sentence = "首先，ASML作为全球最大的光刻机制造厂商，尽管能够领跑全世界，可如果没有大批金主客户，" +
            "ASML也不会过得那么舒坦。中国市场作为全球最大的消费市场，在近年来，国内的半导体企业数量飙升，" +
            "全球每新增20家半导体企业，就有19家是中国的，可见中国市场的巨大潜力。ASML也不傻，虽然在之前一直未大量出口给中国光刻机，" +
            "但是随着中国对DUV光刻机需求的增长，ASML也开始重视起中国市场了。";
    List<String> multiSentences = Arrays.asList(
            "Dies ist ein deutscher Satz",
            "This is an English sentence",
            "这是一个中文句子");

    TranslationClient client = new TranslationClient();

    client.translateSentence(sentence, "zh", "en");
    System.out.println();

    client.translateSentences(sentence, "zh", "en");
    System.out.println();

    File file = new File("./news_1.txt");
    client.translateFile(file, "zh", "en");
    System.out.println();

    client.detectLanguage(multiSentences.toArray(new String[0]));
    System.out.println();
}}
```

(3) 、Nodejs实例

```
const axios = require('axios');
const FormData = require('form-data');
const fs = require('fs');

const URL_SERVER = "http://translate_v2.abcpen.com";

async function translateSentence(sentence, sourceLang, targetLang) {
const url = ${URL_SERVER}/v1/translate/sentence;
const data = { sentence: sentence, source_lang: sourceLang, target_lang: targetLang
};
try {
const response = await axios.post(url, data);
console.log(Time consume: ${response.headers['x-response-time']}s, Result:
${response.data});
} catch (error) {
console.error(error);
}
}

async function translateSentences(sentences, sourceLang, targetLang) {
const url = ${URL_SERVER}/v1/translate/sentences;
const data = { sentences: sentences, source_lang: sourceLang, target_lang:
targetLang };
try {
const response = await axios.post(url, data);
console.log(Time consume: ${response.headers['x-response-time']}s, Result:
${response.data});
} catch (error) {
console.error(error);
}
}

async function translateFile(filename, sourceLang, targetLang) {
const url = ${URL_SERVER}/v1/translate/file;
const form = new FormData();
form.append('target_lang', targetLang);
form.append('document', fs.createReadStream(filename));
try {
const response = await axios.post(url, form, { headers: form.getHeaders() });
console.log(Time consume: ${response.headers['x-response-time']}s, Result:
${response.data});
} catch (error) {
console.error(error);
}
}

async function detectLanguage(sentences) {
const url = ${URL_SERVER}/v1/translate/language_detection;
const data = { text: sentences };
```

```
try {
const response = await axios.post(url, data);
console.log(Language Detection time consume: ${response.headers['x-response-
time']}s, Result: ${response.data});
} catch (error) {
console.error(error);
}
}

async function main() {
const sentence = "首先，ASML作为全球最大的光刻机制造厂商，尽管能够领跑全世界，可如果没有大批金主客户，" +
"ASML也不会过得那么舒坦。中国市场作为全球最大的消费市场，在近年来，国内的半导体企业数量飙升，" +
"全球每新增20家半导体企业，就有19家是中国的，可见中国市场的巨大潜力。ASML也不傻，虽然在之前一直未大量出口给中国光刻机，" +
"但是随着中国对DUV光刻机需求的增长，ASML也开始重视起中国市场了。";
const multiSentences = ["Dies ist ein deutscher Satz", "This is an English
sentence", "这是一个中文句子"];

await translateSentence(sentence, 'zh', 'en');
console.log();

await translateSentences(sentence, 'zh', 'en');
console.log();

await translateFile('./news_1.txt', 'zh', 'en');
console.log();

await detectLanguage(multiSentences);
console.log();
}

main();
```

(4) 、Kotlin实例

```
import okhttp3.*
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.RequestBody.Companion.asRequestBody
import java.io.File
import java.io.IOException

class TranslationClient {
private val URL_SERVER = "http://translate_v2.abcpen.com"
private val JSON_MEDIA_TYPE = "application/json".toMediaTypeOrNull()
    fun translateSentence(sentence: String, sourceLang: String, targetLang: String)
{
    val url = "${URL_SERVER}/v1/translate/sentence"
    val json = """{"sentence": "$sentence", "source_lang": "$sourceLang",
"target_lang": "$targetLang"}"""
    val body = json.toRequestBody(JSON_MEDIA_TYPE)
```

```
    val request = Request.Builder().url(url).post(body).build()
    try {
        val response = OkHttpClient().newCall(request).execute()
        if (!response.isSuccessful) throw IOException("Unexpected code $response")
        println("Time consume: ${response.header("x-response-time")}s, Result:
${response.body?.string()}")
    } catch (e: IOException) {
        e.printStackTrace()
    }
}

fun translateSentences(sentences: String, sourceLang: String, targetLang: String) {
    val url = "${URL_SERVER}/v1/translate/sentences"
    val json = """{"sentences": "$sentences", "source_lang": "$sourceLang",
"target_lang": "$targetLang"}"""
    val body = json.toRequestBody(JSON_MEDIA_TYPE)
    val request = Request.Builder().url(url).post(body).build()
    try {
        val response = OkHttpClient().newCall(request).execute()
        if (!response.isSuccessful) throw IOException("Unexpected code $response")
        println("Time consume: ${response.header("x-response-time")}s, Result:
${response.body?.string()}")
    } catch (e: IOException) {
        e.printStackTrace()
    }
}

fun translateFile(file: File, sourceLang: String, targetLang: String) {
    val url = "${URL_SERVER}/v1/translate/file"
    val body = MultipartBody.Builder().setType(MultipartBody.FORM)
        .addFormDataPart("target_lang", targetLang)
        .addFormDataPart("document", file.name, file.asRequestBody())
        .build()
    val request = Request.Builder().url(url).post(body).build()
    try {
        val response = OkHttpClient().newCall(request).execute()
        if (!response.isSuccessful) throw IOException("Unexpected code $response")
        println("Time consume: ${response.header("x-response-time")}s, Result:
${response.body?.string()}")
    } catch (e: IOException) {
        e.printStackTrace()
    }
}

fun detectLanguage(sentences: Array<String>) {
    val url = "${URL_SERVER}/v1/translate/language_detection"
    val json = """{"text": ${toJsonArray(sentences)}}"""
    val body = json.toRequestBody(JSON_MEDIA_TYPE)
    val request = Request.Builder().url(url).post(body).build()
    try {
        val response = OkHttpClient().newCall(request).execute()
        if (!response.isSuccessful) throw IOException("Unexpected code $response")
```

```
        println(
            "Language Detection time consume: ${response.header("x-response-
time")}s, " +
                    "Result: ${response.body?.string()}"
        )
    } catch (e: IOException) {
        e.printStackTrace()
    }
}

private fun toJsonArray(values: Array<String>): String {
    val builder = StringBuilder("[")
    for (value in values) {
        builder.append("\"").append(value).append("\",")
    }
    builder.deleteChar
    builder.append("]")
    return builder.toString()
}
}
fun main() {
val sentences_zh = """首先，ASML作为全球最大的光刻机制造厂商，尽管能够领跑全世界，可如果没有大批
金主客户，""" +
"""ASML也不会过得那么舒坦。中国市场作为全球最大的消费市场，在近年来，国内的半导体企业数量飙升，"""
+
"""全球每新增20家半导体企业，就有19家是中国的，可见中国市场的巨大潜力。ASML也不傻，虽然在之前""" +
"""一直未大量出口给中国光刻机，但是随着中国对DUV光刻机需求的增长，ASML也开始重视起中国市场了。"""
val multi_sentences = arrayOf("Dies ist ein deutscher Satz", "This is an English
sentence", "这是一个中文句子")
    val client = TranslationClient()
client.translateSentence(sentences_zh, "zh", "en")
println()

client.translateSentences(sentences_zh, "zh", "en")
println()

val file = File("news_1.txt")
client.translateFile(file, "zh", "en")
println()

client.detectLanguage(multi_sentences)
println()
}
```

6. 错误码

错误码	描述	说明	处理方式

错误码	描述	说明	处理方式
0	success	成功	
-1	in progress	识别中	请继续重试
-2	audio encode error	音频编码错误	请编码成正确的格式，再提交请求
10105	illegal access	没有权限	检查apiKey，ip，ts等授权参数是否正确
10106	invalid parameter	无效参数	上传必要的参数， 检查参数格式以及编码
10107	illegal parameter	非法参数值	检查参数值是否超过范围或不符合要求
10110	no license	无授权许可	检查参数值是否超过范围或不符合要求
10700	engine error	引擎错误	提供接口返回值，向服务提供商反馈
16003	basic component error	基础组件异常	重试或向服务提供商反馈
10800	over max connect limit	超过授权的连接数	确认连接数是否超过授权的连接数

四、价格套餐

	免费套餐	套餐一	套餐二	套餐三
服务量	200万字符	5000万字符	2亿字符	10亿字符
有效期	90天	一年	一年	一年
单价（万次）	免费	￥3528.00	￥1120.00	￥15960.00
立即购买	申请链接	购买链接	购买链接	购买链接