ALIBABA CLOUD

阿里云

智能语音交互

长文本语音合成

文档版本：20200930

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.接口说明

长文本语音合成功能提供了将超长文本（如千字或者万字）合成为语音二进制数据的功能。

功能介绍

- 支持输出PCM、WAV和MP3编码格式数据。
- 支持设置语速、语调和音量。
- 支持设置男声、女声。
- 支持通过实时或离线两种方式获取合成结果。
- 长文本语音合成服务相比语音合成服务有其独特优势：
  - 支持更长文字输入：一次性合成最高10万字。
  - 合成速度快：每合成5万字最快仅需10分钟。
  - 循环使用：合成文件支持应用端缓存，可循环使用。
  - 专属声音：按场景打造专属精品声音，完美贴合阅读小说、文章等场景。

> 说明
>
> 使用长文本语音合成功能，需要将SDK更新至最新版本。

声音类型

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	备注
小云	Xiaoyun	标准女声	通用场景	中文及中英文混合场景	8K/16K	无
小刚	Xiaogang	标准男声	通用场景	中文及中英文混合场景	8K/16K	无
若兮	Ruoxi	温柔女声	通用场景	中文及中英文混合场景	8K/16K/24K	无
思琪	Siqi	温柔女声	通用场景	中文及中英文混合场景	8K/16K/24K	无
思佳	Sijia	标准女声	通用场景	中文及中英文混合场景	8K/16K/24K	无

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	备注
思诚	Sicheng	标准男声	通用场景	中文及中英文混合场景	8K/16K/24K	无
艾琪	Aiqi	温柔女声	通用场景	中文及中英文混合场景	8K/16K	无
艾佳	Aijia	标准女声	通用场景	中文及中英文混合场景	8K/16K	无
艾诚	Aicheng	标准男声	通用场景	中文及中英文混合场景	8K/16K	无
艾达	Aida	标准男声	通用场景	中文及中英文混合场景	8K/16K	无
宁儿	Ninger	标准女声	通用场景	纯中文场景	8K/16K/24K	无
瑞琳	Ruilin	标准女声	通用场景	纯中文场景	8K/16K/24K	无
思悦	Siyue	温柔女声	客服场景	中文及中英文混合场景	8K/16K/24K	无
艾雅	Aiya	严厉女声	客服场景	中文及中英文混合场景	8K/16K	无
艾夏	Aixia	亲和女声	客服场景	中文及中英文混合场景	8K/16K	无
艾美	Aimei	甜美女声	客服场景	中文及中英文混合场景	8K/16K	无
艾雨	Aiyu	自然女声	客服场景	中文及中英文混合场景	8K/16K	无
艾悦	Aiyue	温柔女声	客服场景	中文及中英文混合场景	8K/16K	无

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	备注
艾婧	Aijing	严厉女声	客服场景	中文及中英文混合场景	8K/16K	无
小美	Xiaomei	甜美女声	客服场景	中文及中英文混合场景	8K/16K/24K	无
艾娜	Aina	浙普女声	客服场景	纯中文场景	8K/16K	无
伊娜	Yina	浙普女声	客服场景	纯中文场景	8K/16K/24K	无
思婧	Sijing	严厉女声	客服场景	纯中文场景	8K/16K/24K	无
思彤	Sitong	儿童音	童声场景	纯中文场景	8K/16K/24K	无
小北	Xiaobei	萝莉女声	童声场景	纯中文场景	8K/16K/24K	无
艾彤	Aitong	儿童音	童声场景	纯中文场景	8K/16K	无
艾薇	Aiwei	萝莉女声	童声场景	纯中文场景	8K/16K	无
艾宝	Aibao	萝莉女声	童声场景	纯中文场景	8K/16K	无
Harry	Harry	英音男声	英文场景	英文场景	8K/16K	无
Abby	Abby	美音女声	英文场景	英文场景	8K/16K	无
Andy	Andy	美音男声	英文场景	英文场景	8K/16K	无
Eric	Eric	英音男声	英文场景	英文场景	8K/16K	无
Emily	Emily	英音女声	英文场景	英文场景	8K/16K	无

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	备注
Luna	Luna	英音女声	英文场景	英文场景	8K/16K	无
Luca	Luca	英音男声	英文场景	英文场景	8K/16K	无
Wendy	Wendy	英音女声	英文场景	英文场景	8K/16K/24K	无
William	William	英音男声	英文场景	英文场景	8K/16K/24K	无
Olivia	Olivia	英音女声	英文场景	英文场景	8K/16K/24K	无
姗姗	Shanshan	粤语女声	方言场景	标准粤文（简体）及粤英文混合场景	8K/16K/24K	无
艾媛	Aiyuan	知心姐姐	文学场景	中文及中英文混合场景	8K/16K	无
艾颖	Aiying	软萌童声	文学场景	中文及中英文混合场景	8K/16K	无
艾祥	Aixiang	磁性男声	文学场景	中文及中英文混合场景	8K/16K	无
艾墨	Aimo	情感男声	文学场景	中文及中英文混合场景	8K/16K	无
艾晔	Aiye	青年男声	文学场景	中文及中英文混合场景	8K/16K	无
艾婷	Aiting	电台女声	文学场景	中文及中英文混合场景	8K/16K	无
艾凡	Aifan	情感女声	文学场景	中文及中英文混合场景	8K/16K	无

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	备注
Lydia	Lydia	英中双语女声	英文场景	中文及中英文混合场景	8K/16K	公测版
小玥	Xiaoyue	四川话女声	方言场景	中文及中英文混合场景	8K/16K	公测版
艾硕	Aishuo	自然男声	客服场景	中文及中英文混合场景	8K/16K	公测版
艾德	Aide	新闻男声	文学场景	中文及中英文混合场景	8K/16K	公测版
青青	Qingqing	台湾话女声	方言场景	纯中文场景	8K/16K	公测版
翠姐	Cuijie	东北话女声	方言场景	纯中文场景	8K/16K	公测版
小泽	Xiaoze	湖南重口音男声	方言场景	纯中文场景	8K/16K	公测版
艾楠	Ainan	广告男声	文学场景	中文及中英文混合场景	8K/16K	公测版
艾浩	Aihao	资讯男声	文学场景	中文及中英文混合场景	8K/16K	公测版
艾茗	Aiming	诙谐男声	文学场景	中文及中英文混合场景	8K/16K	公测版
艾笑	Aixiao	资讯女声	文学场景	中文及中英文混合场景	8K/16K	公测版

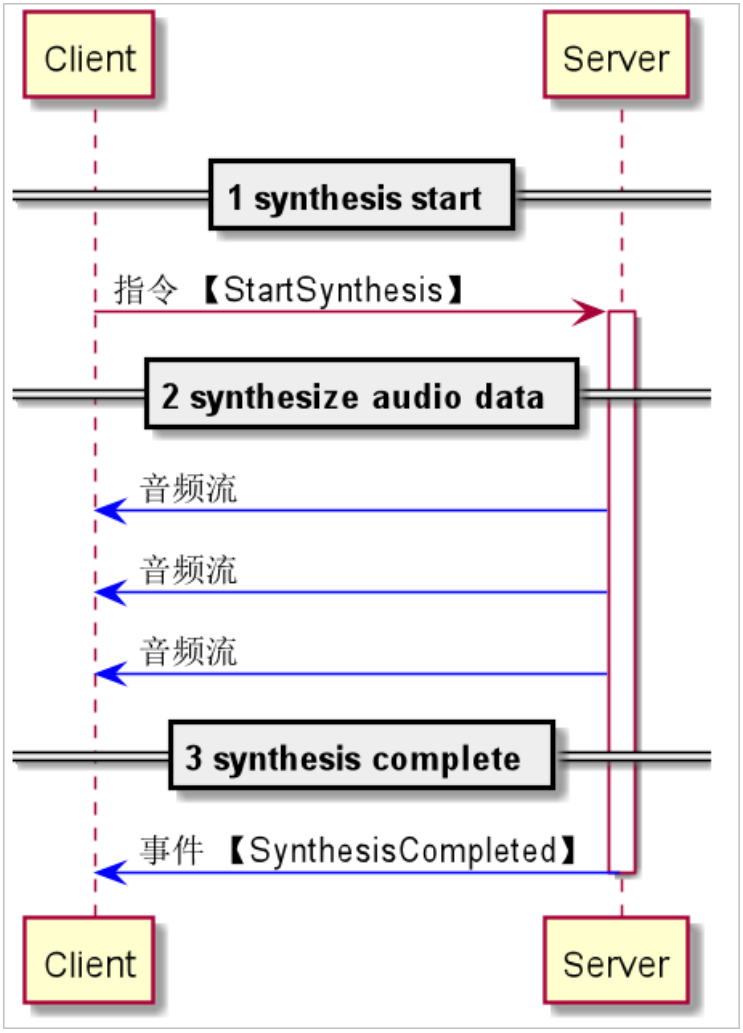
调用说明

- 传入文本必须采用 `UTF-8` 编码。
- 长文本语音合成和语音合成在很多地方都是相似的，可进行对比。

服务地址

访问类型	说明	URL
外网访问	所有服务器均可使用外网访问URL（SDK中默认设置了外网访问URL，不需您设置）	wss://nls-gateway.cn-shanghai.aliyuncs.com/ws/v1
阿里云上海ECS内网访问	使用阿里云上海ECS（ECS地域为华东2（上海）），可使用内网访问URL。 ECS的经典网络不能访问AnyTunnel，即不能在内网访问语音服务；如果希望使用AnyTunnel，需要创建专有网络在其内部访问。 说明 • 使用内网访问方式，将不产生ECS实例的公网流量费用。 • 关于ECS的网络类型请参见网络类型。	ws://nls-gateway.cn-shanghai-internal.aliyuncs.com:80/ws/v1

交互流程



> 说明
>
> - 上图不包含RESTful API的交互流程，关于RESTful API的交互流程图请参见RESTful API。
> - 服务端的响应除了音频流之外，都会在返回信息的header包含本次识别任务的task_id参数，请记录该值，如果出现错误，请将task_id和错误信息提交到工单。

1. 鉴权

客户端在与服务端建立WebSocket连接时，使用Token进行鉴权。Token获取请参见获取Token。

2. 开始合成

客户端发送语音合成请求，在请求消息中进行参数设置，各参数通过SDK中SpeechSynthesizer对象的相关set方法设置，含义如下。

参数	类型	是否必选	说明
appkey	String	是	管控台创建的项目appkey。

参数	类型	是否必选	说明
text	String	是	待合成文本，文本内容必须采用 UTF-8 编码（英文单词之间需要添加空格）。
voice	String	否	发音人，默认是xiaoyun。
format	String	否	音频编码格式，默认值：PCM。支持的格式：PCM、WAV、MP3。
sample_rate	Integer	否	音频采样率，默认值：16000。
volume	Integer	否	音量，范围是0~100。默认值：50。
speech_rate	Integer	否	语速，取值范围：-500~500。默认值：0。
pitch_rate	Integer	否	语调，取值范围：-500~500。默认值：0。

3. 接收合成数据

服务端返回合成的语音二进制数据，SDK接收并处理二进制数据。

4. 结束合成

语音合成完毕，服务端发送合成完毕事件通知，示例如下。

```
{
  "header":{
    "namespace":"SpeechLongSynthesizer",
    "name":"SynthesisCompleted",
    "status":20000000,
    "message_id":"396c80b3abf84082a48cb9e5c424****",
    "task_id":"f5805be640364cdcafc8da63e512****",
    "status_text":"Gateway:SUCCESS:Success."
  }
}
```

> **说明**
>
> 文档示例将合成的音频保存在文件中，如果您需要播放音频且对实时性要求较高，建议使用流式播放，即边接收语音数据边播放，减少延时。

服务状态码

在服务的每一次响应中，都包含status字段，即服务状态码，状态码各种取值含义如下。

- 通用错误

错误码	原因	解决办法
40000001	身份认证失败	检查使用的令牌是否正确，是否过期。
40000002	无效的消息	检查发送的消息是否符合要求。
403	令牌过期或无效的参数	i. 检查使用的令牌是否过期。 ii. 检查参数值设置是否合理。
40000004	空闲超时	确认是否长时间（10秒）未发送数据到服务端。
40000005	请求数量过多	检查是否超过了并发连接数或者每秒钟请求数。如果超过并发数，建议从免费版升级到商用版，或者商用版扩容并发资源。
40000000	默认的客户端错误码	查看错误消息或提交工单。

错误码	原因	解决办法
50000000	默认的服务端错误	如果偶现可以忽略，重复出现请提交工单。
50000001	内部调用错误	如果偶现可以忽略，重复出现请提交工单。

- 网关错误

错误码	原因	解决办法
40010001	不支持的接口	检查是否使用了未支持的接口，如果没有，请提交工单。
40010002	不支持的指令	检查是否使用了未支持的指令，如果没有，请提交工单。
40010003	无效的指令	检查指令格式是否错误，如果没有，请提交工单。
40010004	客户端提前断开连接	检查是否在请求正常完成之前已关闭连接。
40010005	任务状态错误	检查是否发送当前任务状态不能处理的指令。

- 配置错误

错误码	原因	解决办法
40020105	应用不存在	检查应用appkey是否正确，是否与令牌归属同一个账号。

- TTS（Text to Speech）错误

错误码	原因	解决办法
41020001	参数错误	检查是否传递了正确的参数。

错误码	原因	解决办法
51020001	TTS服务端错误	如果偶现可以忽略，重复出现请提交工单。

2.Java SDK

本文介绍如何使用阿里云智能语音服务提供的Java SDK，包括SDK的安装方法及SDK代码示例。

前提条件

- 在使用SDK之前，请先阅读接口说明，详情请参见接口说明。
- 为使用长文本语音合成服务，请将SDK版本更新至2.1.1及以上。

下载安装

从maven 服务器下载最新版本SDK，下载nls-sdk-java-demo.zip。

依赖文件内容如下。

```
<dependency>
    <groupId>com.alibaba.nls</groupId>
    <artifactId>nls-sdk-tts</artifactId>
    <version>2.1.6</version>
</dependency>
```

解压ZIP文件，在pom目录运行 `mvn package` ，会在target目录生成可执行JAR：nls-example-long-tts-2.0.0-jar-with-dependencies.jar，将此JAR包拷贝到目标服务器，用于快速验证及压测服务。

服务验证：

运行如下代码，并按提示提供相应参数。

运行后在命令执行目录生成logs/nls.log。

```
java -cp nls-example-long-tts-2.0.0-jar-with-dependencies.jar com.alibaba.nls.client.SpeechSynthesizerDemo
```

服务压测：

运行如下代码，并按提示提供相应参数。

其中阿里云服务url参数为： `wss://nls-gateway.cn-shanghai.aliyuncs.com/ws/v1` ，并发数根据您的购买情况进行选择。

```
java -jar nls-example-long-tts-2.0.0-jar-with-dependencies.jar
```

> **说明**
>
> 自行压测超过2路并发会产生费用。

关键接口

- NlsClient：语音处理客户端，利用该客户端可以进行一句话识别、实时语音识别和语音合成的语音处理任务。该客户端为线程安全，建议全局仅创建一个实例。

- SpeechSynthesizer：语音合成处理类，设置请求参数，发送请求。非线程安全。
- SpeechSynthesizerListener：语音合成监听类，监听返回结果。非线程安全。需要实现如下两个抽象方法：

```
/**
 * 接收语音合成二进制数据
 */
abstract public void onMessage(ByteBuffer message);
/**
 * 语音合成结束事件通知
 */
abstract public void onComplete(SpeechSynthesizerResponse response);
```

> 说明
>
> SDK调用注意事项：
>
> - NlsClient使用Netty框架，NlsClient对象的创建会消耗一定时间和资源，一经创建可以重复使用。建议调用程序将NlsClient的创建和关闭与程序本身的生命周期相结合。
> - SpeechSynthesizer对象不可重复使用，一个语音合成任务对应一个SpeechSynthesizer对象。例如，N个文本要进行N次语音合成任务，创建N个SpeechSynthesizer对象。
> - SpeechSynthesizerListener对象和SpeechSynthesizer对象是一一对应的，不能将一个SpeechSynthesizerListener对象设置到多个SpeechSynthesizer对象中，否则不能将各语音合成任务区分开。
> - Java SDK依赖Netty网络库，如果您的应用依赖Netty，其版本需更新至4.1.17.Final及以上。

代码示例

> 说明
>
> - 示例中使用SDK内置的默认语音合成服务的外网访问服务URL，如果您使用阿里云上海ECS，且需要通过内网访问服务URL，则在创建NlsClient对象时，设置内网访问的URL：
>
>   ```
>   client = new NlsClient("ws://nls-gateway.cn-shanghai-internal.aliyuncs.com/ws/v1", accessToken);
>   ```
>
> - 示例中将合成的音频保存在文件中，如果您需要播放音频且对实时性要求较高，建议使用流式播放，即边接收语音数据边播放，减少延时。

```
package com.alibaba.nls.client;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import com.alibaba.nls.client.protocol.NlsClient;
```

```
import com.alibaba.nls.client.protocol.OutputFormatEnum;
import com.alibaba.nls.client.protocol.SampleRateEnum;
import com.alibaba.nls.client.protocol.tts.SpeechSynthesizer;
import com.alibaba.nls.client.protocol.tts.SpeechSynthesizerListener;
import com.alibaba.nls.client.protocol.tts.SpeechSynthesizerResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
/**
 * 此示例演示了:
 *     长文本语音合成API调用（setLongText）。
 *     流式合成TTS。
 *     首包延迟计算。
 *
 * 说明：该示例和nls-example-tts下的SpeechSynthesizerLongTextDemo不完全相同，长文本语音合成是单独的产品功能，是将一长串文本直接发送给服务端去合成，
 * 而SpeechSynthesizerLongTextDemo演示的是将一长串文本在调用方处切割然后分段调用语音合成接口。
 */
public class SpeechLongSynthesizerDemo {
    private static final Logger logger = LoggerFactory.getLogger(SpeechLongSynthesizerDemo.class);
    private static long startTime;
    private String appKey;
    NlsClient client;
    public SpeechLongSynthesizerDemo(String appKey, String token, String url) {
        this.appKey = appKey;
        //创建NlsClient实例应用全局创建一个即可。生命周期可和整个应用保持一致，默认服务地址为阿里云线上服务地址。
        if(url.isEmpty()) {
            client = new NlsClient(token);
        } else {
            client = new NlsClient(url, token);
        }
    }
    private static SpeechSynthesizerListener getSynthesizerListener() {
        SpeechSynthesizerListener listener = null;
        try {
            listener = new SpeechSynthesizerListener() {
                File f=new File("ttsForLongText.wav");
                FileOutputStream fout = new FileOutputStream(f);
                private boolean firstRecvBinary = true;
                //语音合成结束
                @Override
```

```
            public void onComplete(SpeechSynthesizerResponse response) {
                // 调用onComplete时，表示所有TTS数据已经接收完成，因此为整个合成数据的延迟。该延迟可能较大，不一定满足实时场景。
                System.out.println("name: " + response.getName() + ", status: " + response.getStatus()+", output file :"+f.getAbsolutePath());
            }
            //语音合成的语音二进制数据
            @Override
            public void onMessage(ByteBuffer message) {
              try {
                if(firstRecvBinary) {
                  // 此处计算首包语音流的延迟，收到第一包语音流时，即可以进行语音播放，以提升响应速度（特别是实时交互场景下）。
                  firstRecvBinary = false;
                  long now = System.currentTimeMillis();
                  logger.info("tts first latency : " + (now - SpeechLongSynthesizerDemo.startTime) + " ms");
                }
                byte[] bytesArray = new byte[message.remaining()];
                message.get(bytesArray, 0, bytesArray.length);
                //System.out.println("write array:" + bytesArray.length);
                fout.write(bytesArray);
              } catch (IOException e) {
                e.printStackTrace();
              }
            }
            @Override
            public void onFail(SpeechSynthesizerResponse response){
              // task_id是调用方和服务端通信的唯一标识，当遇到问题时，需要提供此task_id以便排查。
              System.out.println(
                "task_id: " + response.getTaskId() +
                  //状态码
                  ", status: " + response.getStatus() +
                  //错误信息
                  ", status_text: " + response.getStatusText());
            }
          };
      } catch (Exception e) {
        e.printStackTrace();
      }
```

```
        return listener;
    }
    public void process(String text) {
        SpeechSynthesizer synthesizer = null;
        try {
            //创建实例，建立连接。
            synthesizer = new SpeechSynthesizer(client, getSynthesizerListener());
            synthesizer.setAppKey(appKey);
            //设置返回音频的编码格式。
            synthesizer.setFormat(OutputFormatEnum.WAV);
            //设置返回音频的采样率。
            synthesizer.setSampleRate(SampleRateEnum.SAMPLE_RATE_16K);
            //发音人
            synthesizer.setVoice("siyue");
            //语调，范围是-500~500，可选，默认是0。
            synthesizer.setPitchRate(0);
            //语速，范围是-500~500，默认是0。
            synthesizer.setSpeechRate(0);
            //设置用于语音合成的文本
            // 此处调用的是setLongText接口（原语音合成接口是setText）。
            synthesizer.setLongText(text);
            //此方法将以上参数设置序列化为JSON发送给服务端，并等待服务端确认。
            long start = System.currentTimeMillis();
            synthesizer.start();
            logger.info("tts start latency " + (System.currentTimeMillis() - start) + " ms");
            SpeechLongSynthesizerDemo.startTime = System.currentTimeMillis();
            //等待语音合成结束
            synthesizer.waitForComplete();
            logger.info("tts stop latency " + (System.currentTimeMillis() - start) + " ms");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            //关闭连接
            if (null != synthesizer) {
                synthesizer.close();
            }
        }
    }
    public void shutdown() {
        client.shutdown();
    }
```

```
    public static void main(String[] args) throws Exception {
        String appKey = "";
        String token = "填写你的token";
        // url取默认值
        String url = "wss://nls-gateway.cn-shanghai.aliyuncs.com/ws/v1";
        if (args.length == 2) {
            appKey  = args[0];
            token   = args[1];
        } else if (args.length == 3) {
            appKey  = args[0];
            token   = args[1];
            url     = args[2];
        } else {
            System.err.println("run error, need params(url is optional): " + "<app-key> <token> [url]");
            System.exit(-1);
        }
        String ttsTextLong = "百草堂与三味书屋 鲁迅 \n" +
            "我家的后面有一个很大的园，相传叫作百草园。现在是早已并屋子一起卖给朱文公的子孙了，连那最末次的相见也已经隔了七八年，其中似乎确凿只有一些野草；但那时却是我的乐园。\n" +
            "不必说碧绿的菜畦，光滑的石井栏，高大的皂荚树，紫红的桑葚；也不必说鸣蝉在树叶里长吟，肥胖的黄蜂伏在菜花上，轻捷的叫天子(云雀)忽然从草间直窜向云霄里去了。\n" +
            "单是周围的短短的泥墙根一带，就有无限趣味。油蛉在这里低唱，蟋蟀们在这里弹琴。翻开断砖来，有时会遇见蜈蚣；还有斑蝥，倘若用手指按住它的脊梁，便会啪的一声，\n" +
            "从后窍喷出一阵烟雾。何首乌藤和木莲藤缠络着，木莲有莲房一般的果实，何首乌有臃肿的根。有人说，何首乌根是有像人形的，吃了便可以成仙，我于是常常拔它起来，牵连不断地拔起来，\n" +
            "也曾因此弄坏了泥墙，却从来没有见过有一块根像人样! 如果不怕刺，还可以摘到覆盆子，像小珊瑚珠攒成的小球，又酸又甜，色味都比桑葚要好得远.....";
        SpeechLongSynthesizerDemo demo = new SpeechLongSynthesizerDemo(appKey, token, url);
        demo.process(ttsTextLong);
        demo.shutdown();
    }
}
```

3.C++ SDK（新）

本文介绍如何使用阿里云智能语音服务提供的C++ SDK，包括SDK的安装方法及SDK代码示例。

> 说明
>
> - 当前最新版本：3.0.8。发布日期：2020年1月9日。
>
> 该版本只支持Linux平台，暂不支持Windows平台。
> - 请先阅读接口说明，详情请参见接口说明。
> - 之前版本C++ SDK不支持长文本语音合成，本文以当前版本为例进行介绍。

下载安装

SDK下载：

下载CppSdk，文件包含如下几部分：

- CMakeLists.txt：示例代码工程的CMakeList文件。
- readme.txt：SDK说明。
- release.log：版本说明。
- version：版本号。
- build.sh：示例编译脚本。
- lib：SDK库文件。
- build：编译目录。
- demo：示例，语音服务配置文件，如下表所示。

文件名	描述
speechRecognizerDemo.cpp	一句话识别示例
speechSynthesizerDemo.cpp	语音合成示例
speechTranscriberDemo.cpp	实时语音识别示例
speechLongSynthesizerDemo.cpp	长文本语音合成示例
test0.wav/test1.wav	测试音频（16k采样频率、16bit采样位数的音频文件）

- include：SDK头文件，如下表所示。

文件名	描述
nlsClient.h	SDK实例
nlsEvent.h	回调事件说明
speechRecognizerRequest.h	一句哈识别
speechSynthesizerRequest.h	语音合成/长文本语音合成
speechTranscriberRequest.h	实时音频流识别

编译运行：

1. 安装工具的最低版本要求如下：
   i. Cmake 3.1
   ii. Glibc 2.5
   iii. Gcc 4.1.2
2. 在Linux终端运行如下脚本。

```
mkdir build
cd build && cmake .. && make
cd ../demo #生成示例可执行程序：srDemo（一句话识别）、stDemo（实时语音识别）、syDemo（语音合成）、syLongDemo（长文本语音合成）。
./syLongDemo appkey  <yourAccessKey Id> <yourAccessKey Secret>  # 测试使用。
```

关键接口

- 基础接口
  - NlsClient：语音处理客户端，利用该客户端可以进行一句话识别、实时语音识别和语音合成的语音处理任务。该客户端为线程安全，建议全局仅创建一个实例。
  - NlsEvent：事件对象，您可以从中获取Request状态码、云端返回结果、失败信息等。
- 识别接口

  SpeechSynthesizerRequest：语音合成请求对象，用于语音合成及长文本语音合成。

C++ SDK错误码

错误码	错误描述	解决方案
10000001	SSL: couldn't create a ……!	建议重试。
10000002	openssl官方错误描述	根据描述提示处理后，建议重试。
10000003	系统错误描述	根据系统错误描述提示处理。
10000004	URL: The url is empty.	检查是否设置云端URL地址。
10000005	URL: Could not parse WebSocket url.	检查是否正确设置云端URL地址。
10000006	MODE: unsupport mode.	检查是否正确设置语音功能模式。
10000007	JSON: Json parse failed.	服务端发送错误响应内容，请提供task_id，并反馈给阿里云。
10000008	WEBSOCKET: unkown head type.	服务端发送错误WebSocket类型，请提供task_id，并通过工单系统反馈给我们。
10000009	HTTP: connect failed.	与云端连接失败，请检查网络，重试。
HTTP协议官方状态码	HTTP: Got bad status.	根据HTTP协议官方描述提示处理。
系统错误码	IP: ip address is not valid.	根据系统错误描述提示处理。
系统错误码	ENCODE: convert to utf8 error.	根据系统错误描述提示处理。
10000010	please check if the memory is enough.	内存不足，请检查本地机器内存。

错误码	错误描述	解决方案
10000011	Please check the order of execution.	接口调用顺序错误（接收到Failed/complete事件时，SDK内部会关闭连接。此时再调用send方法会上报错误。
10000012	StartCommand/StopCommand Send failed.	参数错误。请检查参数设置是否正确。
10000013	The sent data is null or dataSize <= 0.	发送错误。请检查发送参数是否正确。
10000014	Start invoke failed.	调用start方法超时错误。请调用stop方法释放资源，重新开始识别流程。
10000015	connect failed.	调用connect方法失败。请调用stop方法释放资源，重新开始识别流程。

服务端响应状态码

关于服务状态码，请参见服务状态码。

代码示例

说明

- 示例中使用了SDK内置的默认外网访问服务URL，如果您使用阿里云上海ECS且需要使用内网访问服务URL，则在创建SpeechSynthesizerRequest的对象中设置内网访问的URL：

  ```
  request->setUrl("ws://nls-gateway.cn-shanghai-internal.aliyuncs.com/ws/v1");
  ```

- 示例中将合成的音频保存在文件中，如果您需要播放音频且对实时性要求较高，建议使用流式播放，即边接收语音数据边播放，减少延时，而无需等待合成结束后再处理语音流。
- 完整示例参见SDK文件中demo目录的speechLongSynthesizerDemo.cpp文件。

```
#include <pthread.h>
#include <unistd.h>
#include <sys/time.h>
#include <ctime>
#include <string>
```

```
#include <vector>
#include <fstream>
#include "nlsClient.h"
#include "nlsEvent.h"
#include "speechSynthesizerRequest.h"
#include "nlsCommonSdk/Token.h"
using namespace AlibabaNlsCommon;
using AlibabaNls::NlsClient;
using AlibabaNls::NlsEvent;
using AlibabaNls::LogDebug;
using AlibabaNls::LogInfo;
using AlibabaNls::SpeechSynthesizerRequest;
// 自定义线程参数
struct ParamStruct {
    std::string text;
    std::string token;
    std::string appkey;
    std::string audioFile;
};
// 自定义事件回调参数
struct ParamCallBack {
    std::string binAudioFile;
    std::ofstream audioFile;
    uint64_t startMs;
};
//全局维护一个服务鉴权token和其对应的有效期时间戳，
//每次调用服务之前，首先判断token是否已经过期，
//如果已经过期，则根据AccessKey ID和AccessKey Secret重新生成一个token，并更新该全局的token和其有效期时间戳。
//注意：只需在token即将过期时重新生成即可，所有服务并发可共用一个token。
std::string g_akId = "";
std::string g_akSecret = "";
std::string g_token = "";
long g_expireTime = -1;
uint64_t getNow() {
    struct timeval now;
    gettimeofday(&now, NULL);
    return now.tv_sec * 1000 * 1000 + now.tv_usec;
}
int generateToken(std::string akId, std::string akSecret, std::string* token, long* expireTime) {
    NlsToken nlsTokenRequest;
```

```
    nlsTokenRequest.setAccessKeyId(akId);
    nlsTokenRequest.setKeySecret(akSecret);
    if (-1 == nlsTokenRequest.applyNlsToken()) {
        // 获取失败原因
        printf("generateToken Failed: %s\n", nlsTokenRequest.getErrorMsg());
        return -1;
    }
    *token = nlsTokenRequest.getToken();
    *expireTime = nlsTokenRequest.getExpireTime();
    return 0;
}
//@brief SDK在接收到云端返回合成结束消息时，其内部线程上报Completed事件。
//@note 上报Completed事件之后，SDK内部会关闭识别连接通道。
//@param cbEvent 回调事件结构，详见nlsEvent.h。
//@param cbParam 回调自定义参数，默认为NULL，可以根据需求自定义参数。
void OnSynthesisCompleted(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    // 演示如何打印/使用用户自定义参数示例。
    printf("OnSynthesisCompleted: %s\n", tmpParam->binAudioFile.c_str());
    // 获取消息的状态码，成功为0或者20000000，失败时对应失败的错误码。
    // 当前任务的task id。
    printf("OnSynthesisCompleted: status code=%d, task id=%s\n", cbEvent->getStatusCode(), cbEvent-
>getTaskId());
    // 获取服务端返回的全部信息。
    //printf("OnSynthesisCompleted: all response=%s\n", cbEvent->getAllResponse());
}
//@brief 合成过程发生异常时，SDK内部线程上报TaskFailed事件。
//@note 上报TaskFailed事件后，SDK内部会关闭识别连接通道。
//@param cbEvent 回调事件结构，详见nlsEvent.h。
//@param cbParam 回调自定义参数，默认为NULL，可以根据需求自定义参数。
void OnSynthesisTaskFailed(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    // 演示如何打印/使用用户自定义参数示例。
    printf("OnSynthesisTaskFailed: %s\n", tmpParam->binAudioFile.c_str());
    // 当前任务的task id。
    printf("OnSynthesisTaskFailed: status code=%d, task id=%s, error message=%s\n", cbEvent->getSt
atusCode(), cbEvent->getTaskId(), cbEvent->getErrorMessage());
}
//@brief 识别结束或发生异常时，会关闭连接通道，SDK内部线程上报ChannelCloseed事件。
//@param cbEvent 回调事件结构，详见nlsEvent.h。
//@param cbParam 回调自定义参数，默认为NULL，可以根据需求自定义参数。
```

```
//@param cbParam 回调自定义参数，默认为NULL，可以根据需求自定义参数。
void OnSynthesisChannelClosed(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    // 演示如何打印/使用用户自定义参数示例。
    printf("OnSynthesisChannelClosed: %s\n", tmpParam->binAudioFile.c_str());
    printf("OnSynthesisChannelClosed: %s\n", cbEvent->getAllResponse());
    tmpParam->audioFile.close();
    delete tmpParam; //识别流程结束，释放回调参数。
}
//@brief 文本上报服务端之后，收到服务端返回的二进制音频数据，SDK内部线程通过BinaryDataRecved事件上报给您。
//@param cbEvent 回调事件结构，详见nlsEvent.h。
//@param cbParam 回调自定义参数，默认为NULL，可以根据需求自定义参数。
void OnBinaryDataRecved(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
    if(tmpParam->startMs > 0 ) {
        // 说明：一旦获取到语音流，如第一次从服务端返回合成语音流，即可开始进行播放或者其他处理，本示例为保存到本地文件。
        // 第一次收到语音流数据，计算TTS合成首包延迟。另外此处计算首包延迟时也包括了start操作（即本程序连接公共云服务端的时间），而该时间受不同网络因素影响可能有较大差异。
        uint64_t now = getNow();
        printf("first latency = %lld ms, task id = %s\n", (now - tmpParam->startMs) / 1000, cbEvent->getTaskId());
        tmpParam->startMs = 0;
    }
    // 演示如何打印/使用用户自定义参数示例。
    printf("OnBinaryDataRecved: %s\n", tmpParam->binAudioFile.c_str());
    const std::vector<unsigned char>& data = cbEvent->getBinaryData(); // getBinaryData()为获取文本合成的二进制音频数据。
    printf("OnBinaryDataRecved: status code=%d, task id=%s, data size=%d\n", cbEvent->getStatusCode(), cbEvent->getTaskId(), data.size());
    // 以追加形式将二进制音频数据写入文件。
    if (data.size() > 0) {
        tmpParam->audioFile.write((char*)&data[0], data.size());
    }
}
//@brief 返回tts文本对应的日志信息，增量返回对应的字幕信息。
//@param cbEvent 回调事件结构，详见nlsEvent.h。
//@param cbParam 回调自定义参数，默认为NULL，可以根据需求自定义参数。
void OnMetaInfo(NlsEvent* cbEvent, void* cbParam) {
    ParamCallBack* tmpParam = (ParamCallBack*)cbParam;
```

```
    // 演示如何打印/使用用户自定义参数示例。
    printf("OnBinaryDataRecved: %s\n", tmpParam->binAudioFile.c_str());
    printf("OnMetaInfo: task id=%s, respose=%s\n", cbEvent->getTaskId(), cbEvent->getAllResponse());
}
// 工作线程
void* pthreadFunc(void* arg) {
    // 0: 从自定义线程参数中获取token，配置文件等参数。
    ParamStruct* tst = (ParamStruct*)arg;
    if (tst == NULL) {
        printf("arg is not valid\n");
        return NULL;
    }
    // 1: 初始化自定义回调参数。
    ParamCallBack* cbParam = new ParamCallBack;
    cbParam->binAudioFile = tst->audioFile;
    cbParam->audioFile.open(cbParam->binAudioFile.c_str(), std::ios::binary | std::ios::out);
    // 2: 创建语音识别SpeechSynthesizerRequest对象。
    SpeechSynthesizerRequest* request = NlsClient::getInstance()->createSynthesizerRequest(AlibabaNls::LongTts);
    if (request == NULL) {
        printf("createSynthesizerRequest failed.\n");
        cbParam->audioFile.close();
        return NULL;
    }
    request->setOnSynthesisCompleted(OnSynthesisCompleted, cbParam); // 设置音频合成结束回调函数
    request->setOnChannelClosed(OnSynthesisChannelClosed, cbParam); // 设置音频合成通道关闭回调函数
    request->setOnTaskFailed(OnSynthesisTaskFailed, cbParam); // 设置异常失败回调函数
    request->setOnBinaryDataReceived(OnBinaryDataRecved, cbParam); // 设置文本音频数据接收回调函数
    request->setOnMetaInfo(OnMetaInfo, cbParam); // 设置字幕信息
    request->setAppKey(tst->appkey.c_str());
    request->setText(tst->text.c_str()); // 设置待合成文本，必填参数。文本内容必须为UTF-8编码。
    request->setVoice("siqi");          // 发音人。可选参数，默认是xiaoyun。
    request->setVolume(50);           // 音量，范围是0~100，可选参数，默认50。
    request->setFormat("wav");          // 音频编码格式，可选参数，默认是wav。支持的格式PCM、WAV和MP3。
    request->setSampleRate(8000);       // 音频采样率：8000/16000。可选参数，默认是16000。
    request->setSpeechRate(0);          // 语速，范围是-500~500，可选参数，默认是0。
    request->setPitchRate(0);           // 语调，范围是-500~500，可选参数，默认是0。
    //request->setEnableSubtitle(true);     //是否开启字幕，非必须，需要注意的是并不是所有发音人都支持字幕功能。
```

```
    request->setToken(tst->token.c_str()); // 设置账号校验token，必填参数。
    cbParam->startMs = getNow();
    // 3：start()为异步操作。成功返回BinaryRecv事件。失败返回TaskFailed事件。
    if (request->start() < 0) {
        printf("start() failed. may be can not connect server. please check network or firewalld\n");
        NlsClient::getInstance()->releaseSynthesizerRequest(request); // start()失败，释放request对象
        cbParam->audioFile.close();
        return NULL;
    }
    //4：通知云端数据发送结束。
    //stop()为异步操作，失败返回TaskFailed事件。
    request->stop();
    // 5：识别结束，释放request对象。
    NlsClient::getInstance()->releaseSynthesizerRequest(request);
    return NULL;
}
// 合成单个文本数据
int speechLongSynthesizerFile(const char* appkey) {
    //获取当前系统时间戳，判断token是否过期。
    std::time_t curTime = std::time(0);
    if (g_expireTime - curTime < 10) {
        printf("the token will be expired, please generate new token by AccessKey-ID and AccessKey-Secr
et.\n");
        if (-1 == generateToken(g_akId, g_akSecret, &g_token, &g_expireTime)) {
            return -1;
        }
    }
    ParamStruct pa;
    pa.token = g_token;
    pa.appkey = appkey;
    // 注意：Windows平台下，合成文本中如果包含中文，请将本CPP文件设置为带签名的UTF-8编码或者GB2312编
码。
    pa.text = "今天天气很棒，适合去户外旅行.";
    pa.audioFile = "syAudio.wav";
    pthread_t pthreadId;
    // 启动一个工作线程，用于识别。
    pthread_create(&pthreadId, NULL, &pthreadFunc, (void *)&pa);
    pthread_join(pthreadId, NULL);
    return 0;
}
// 合成多个文本数据
```

```
// 合成多个文本数据
// SDK多线程指一个文本数据对应一个线程，非一个文本数据对应多个线程。
// 示例代码为同时开启2个线程，合成2个文件。
#define AUDIO_TEXT_NUMS 2
#define AUDIO_TEXT_LENGTH 64
#define AUDIO_FILE_NAME_LENGTH 32
int speechLongSynthesizerMultFile(const char* appkey) {
    //获取当前系统时间戳，判断token是否过期。
    std::time_t curTime = std::time(0);
    if (g_expireTime - curTime < 10) {
        printf("the token will be expired, please generate new token by AccessKey-ID and AccessKey-Secret.\n");
        if (-1 == generateToken(g_akId, g_akSecret, &g_token, &g_expireTime)) {
            return -1;
        }
    }
    const char syAudioFiles[AUDIO_TEXT_NUMS][AUDIO_FILE_NAME_LENGTH] = {"syAudio0.wav", "syAudio1.wav"};
    const char texts[AUDIO_TEXT_NUMS][AUDIO_TEXT_LENGTH] = {"今日天气真不错，我想去操作踢足球.", "明天有大暴雨，还是宅在家里看电影吧."};
    ParamStruct pa[AUDIO_TEXT_NUMS];
    for (int i = 0; i < AUDIO_TEXT_NUMS; i ++) {
        pa[i].token = g_token;
        pa[i].appkey = appkey;
        pa[i].text = texts[i];
        pa[i].audioFile = syAudioFiles[i];
    }
    std::vector<pthread_t> pthreadId(AUDIO_TEXT_NUMS);
    // 启动工作线程，同时识别音频文件。
    for (int j = 0; j < AUDIO_TEXT_NUMS; j++) {
        pthread_create(&pthreadId[j], NULL, &pthreadFunc, (void *)&(pa[j]));
    }
    for (int j = 0; j < AUDIO_TEXT_NUMS; j++) {
        pthread_join(pthreadId[j], NULL);
    }
    return 0;
}
int main(int arc, char* argv[]) {
    if (arc < 4) {
        printf("params is not valid. Usage: ./demo <your appkey> <your AccessKey ID> <your AccessKey Secret>\n");
```

```
        return -1;
    }
    std::string appkey = argv[1];
    g_akId = argv[2];
    g_akSecret = argv[3];
    // 根据需要设置SDK输出日志，可选。此处表示SDK日志输出至log-Synthesizer.txt，LogDebug表示输出所有级别日志。
    int ret = NlsClient::getInstance()->setLogConfig("log-longsynthesizer", LogDebug);
    if (-1 == ret) {
        printf("set log failed\n");
        return -1;
    }
    //启动工作线程
    NlsClient::getInstance()->startWorkThread(4);
    /// 注意：长文本语音合成目前没有免费试用版，必须开通商业版长文本语音合成之后才能进行测试。
    // 合成单个文本
    speechLongSynthesizerFile(appkey.c_str());
    // 合成多个文本
    // speechLongSynthesizerMultFile(appkey.c_str());
    // 所有工作完成，进程退出前，释放nlsClient。请注意，releaseInstance()非线程安全。
    NlsClient::releaseInstance();
    return 0;
}
```

4.RESTful API

长文本语音合成RESTful API支持HTTP/HTTPS POST方式请求，将待合成的文本通过HTTP POST上传到服务端，服务端返回文本的语音合成结果。

功能介绍

支持如下设置：

- 合成音频的格式：PCM/WAV/MP3。
- 合成音频的采样率：8000Hz/16000Hz。
- 多种发音人。
- 语速/语调/音量。
- 数据获取方式：轮询方式/回调方式。

> 注意
>
> - 随着TTS合成效果不断提升，算法的复杂度也越来越高，对您而言，可能会遇到合成耗时变长的情况。因此我们建议您使用流式合成机制。本文及SDK示例中有相关流式处理示例代码可供参考。
> - 下载nls-restful-java-demo.zip。
>
>   新增RESTful语音合成Java示例代码。

前提条件

- 已获取项目appkey，详情请参见创建项目。
- 已获取Access Token，详情请参见获取Token。

服务地址

访问类型	说明	URL	Host
外网访问	所有服务器均可使用外网访问URL。	https://nls-gateway.cn-shanghai.aliyuncs.com/rest/v1/tts/async	nls-gateway.cn-shanghai.aliyuncs.com
阿里云上海ECS内网访问	使用阿里云上海ECS（即ECS地域为华东2（上海）），可使用内网访问URL。	http://nls-gateway.cn-shanghai-internal.aliyuncs.com/rest/v1/tts/async	nls-gateway.cn-shanghai-internal.aliyuncs.com

> **注意**
>
> 以下将以使用外网访问URL的方式进行介绍。如果您使用阿里云上海ECS，并需要通过内网访问URL，则使用HTTP协议，并替换外网访问的URL和Host。以下示例中除了Python，均支持HTTP和HTTPS协议，在使用Python示例时请阅读其注意事项。

交互流程

客户端向服务端发送携带文本内容的HTTPS POST方法的请求，服务端返回对应的处理。此后客户端有两种处理方式：

- 主动轮询合成状态，直至合成完成。
- 等待服务端全部完成语音合成后主动回调用户设置的回调地址，此时用户端程序可以继续进行后续处理。

> **说明**
>
> - 不同于语音合成的RESTful接口，长文本语音合成RESTful接口并不会把实际的合成数据直接返回给客户端，而是返回一个HTTP地址，您可以通过该HTTP地址下载录音文件或者播放。
> - 服务端的错误响应都会在返回信息中包含表示本次合成任务的task_id参数，请记录该值，如果出现错误，请将task_id和错误信息提交到工单。

请求参数

长文本语音合成的请求参数如下表所示。

发送HTTP/HTTPS请求时需要将这些参数设置到请求体（Body）中。

名称	类型	是否必选	描述
appkey	String	是	应用appkey。
token	String	否	服务鉴权Token。
text	String	是	待合成的文本，需要为 `UTF-8` 编码。
format	String	是	音频编码格式，支持PCM/WAV/MP3格式，默认是PCM。
sample_rate	Integer	是	音频采样率，支持16000Hz/8000Hz，默认是16000Hz。

名称	类型	是否必选	描述
voice	String	否	发音人，默认是xiaoyun。更多发音人请参见接口说明。
volume	Integer	否	音量，范围是0~100，默认50。
speech_rate	Integer	否	语速，范围是-500~500，默认是0。
pitch_rate	Integer	否	语调，范围是-500~500，默认是0。
enable_subtitle	Boolean	否	是否启用句级时间戳功能，默认值为false。
enable_notify	Boolean	是	是否启用回调功能，默认值为false。
notify_url	String	否	回调服务的地址。 当 `enable_notify` 取值为true时，本字段必填。 url支持HTTP/HTTPS协议，host不能使用IP地址。

POST方法上传文本

一个完整的语音合成RESTful API POST请求包含以下要素：

- URL

协议	URL	方法
HTTPS	https://nls-gateway.cn-shanghai.aliyuncs.com/rest/v1/tts/async	POST

- HTTPS POST请求体

  请求体是请求参数组成的JSON格式字符串，因此在POST请求头部中的 `Content-Type` 必须设置为”application/json”，示例如下。

```
{
  "payload":{
    "tts_request":{
      "voice":"xiaoyun",
      "sample_rate":16000,
      "format":"wav",
      "text":"今天天气好晴朗",
      "enable_subtitle": true
    },
    "enable_notify":false
  },
  "context":{
    "device_id":"my_device_id"
  },
  "header":{
    "appkey":"yourAppkey",
    "token":"yourToken"
  }
}
```

响应结果

发送请求后，不论是调用成功还是失败，服务端的响应消息都会通过HTTP的消息体返回给客户端。使用HTTPS GET方法和使用HTTPS POST方法请求的响应是相同的，响应的结果都包含在HTTPS的Body中。响应结果的成功或失败通过HTTPS Header的 `Content-Type` 字段来区分：

- 成功响应
  - HTTPS Headers的 `Content-Type` 字段内容为 `audio/mpeg` ，表示合成成功，合成的语音数据在Body中。
  - HTTPS Header的 `X-NLS-RequestId` 字段内容为请求任务的 `task_id` 。
  - Body内容为合成音频的二进制数据。
- 失败响应
  - HTTPS Headers没有 `Content-Type` 字段，或者 `Content-Type` 字段内容为 `application/json` ，表示合成失败，错误信息在Body中。
  - HTTPS Header的 `X-NLS-RequestId` 字段内容为请求任务的 `task_id` 。

- Body内容为错误信息，以JSON格式的字符串表示。

  错误信息字段如下表所示。

名称	类型	描述
task_id	String	32位请求任务ID，请记录该值，用于排查错误。
error_code	Integer	服务状态码。
error_message	String	服务状态描述。

- 发起请求后的成功响应

```
{
 "status":200,
 "error_code":20000000,
 "error_message":"SUCCESS",
 "request_id":"f0a9e2c49e9049e78730a3bf0b32****",
 "data":{
    "task_id":"35d9f813e00b11e9a2ce9ba0d6a2****"
 }
}
```

- 发起请求后的失败响应

```
{
 "error_message":"Meta:ACCESS_DENIED:The token 'fdf' is invalid!",
 "error_code":40000001,
 "request_id":"0d8c0eea55824aada9a374aec650****",
 "url":"/rest/v1/tts/async",
 "status":400
}
```

- 轮询服务端合成状态的成功响应

```
/// 轮询时服务端返回的中间状态。
{
   "status":200,
   "error_code":20000000,
   "error_message":"RUNNING",
   "request_id":"a3370c49a29148e78b39978f98ba****",
   "data":{
```

```
    "task_id":"35d9f813e00b11e9a2ce9ba0d6a2****",
    "audio_address":null,
    "notify_custom":null
  }
}
/// 获取到最终合成结果。
{
  "status":200,
  "error_code":20000000,
  "error_message":"SUCCESS",
  "request_id":"c541eae489af48d69dae2d2e203a****",
  "data":{
    "sentences":[
      {
        "text":"长文本语音合成接口",
        "begin_time":"0",
        "end_time":"2239"
      },
      {
        "text":"一次返回所有文本对应的音频.现在需要增加句级别的时间戳信息",
        "begin_time":"2239",
        "end_time":"8499"
      },
      {
        "text":"客户可利用该信息，实现播放控制功能",
        "begin_time":"8499",
        "end_time":"12058"
      }
    ],
    "task_id":"f4e9bf53cb1611eab327b15f61b4****",
    "audio_address":"此处为生成的url地址",
    "notify_custom":""
  }
}
```

> **说明**
>
> audio_address字段对应的HTTP下载地址有期限最多只有3天。

服务状态码

服务状态码	服务状态描述	解决办法
20000000	请求成功	无。
40000000	默认的客户端错误码	查看错误消息或提交工单。
40000001	身份认证失败	检查使用的令牌是否正确，是否过期。
40000002	无效的消息	检查发送的消息是否符合要求。
40000003	无效的参数	检查参数值设置是否合理。
40000004	空闲超时	确认是否长时间没有发送数据到服务端。
40000005	请求数量过多	检查是否超过了并发连接数或者每秒钟请求数。
50000000	默认的服务端错误	如果偶现可以忽略，重复出现请提交工单。
50000001	内部GRPC调用错误	如果偶现可以忽略，重复出现请提交工单。

Java示例

依赖文件内容如下。

```
<dependency>
    <groupId>com.squareup.okhttp3</groupId>
    <artifactId>okhttp</artifactId>
    <version>3.9.1</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.49</version>
</dependency>
```

示例代码如下。

```
package com.alibaba.nls.client;
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
/**
 * 此示例演示了长文本语音合成的使用方式。
 */
public class SpeechLongSynthesizerRestfulDemo {
    private static Logger logger = LoggerFactory.getLogger(SpeechLongSynthesizerRestfulDemo.class)
;
    private String accessToken;
    private String appkey;
    public SpeechLongSynthesizerRestfulDemo(String appkey, String token) {
        this.appkey = appkey;
        this.accessToken = token;
    }
    public void processPOSTRequest(String text, String format, int sampleRate, String voice) {
        String url = "https://nls-gateway.cn-shanghai.aliyuncs.com/rest/v1/tts/async";
        // 拼接HTTP Post请求的消息体内容。
        JSONObject context = new JSONObject();
        // device_id设置，可以设置为自定义字符串或者设备信息id。
        context.put("device_id", "my_device_id");
        JSONObject header = new JSONObject();
        // 设置你的appkey。
        header.put("appkey", appkey);
        // 设置你的token。
        header.put("token", accessToken);
        // voice 发音人，可选，默认是xiaoyun。
        // volume 音量，范围是0~100，可选，默认50。
        // speech_rate 语速，范围是-500~500，可选，默认是0。
        // pitch_rate 语调，范围是-500~500，可选，默认是0。
        JSONObject tts = new JSONObject();
        tts.put("text", text);
```

```
        // 设置发音人。
        tts.put("voice", voice);
        // 设置编码格式。
        tts.put("format", format);
        // 设置采样率。
        tts.put("sample_rate", sampleRate);
        // 设置声音大小，可选。
        //tts.put("volume", 100);
        // 设置语速，可选。
        //tts.put("speech_rate", 200);
        // 长文本tts restful接口支持句级时间戳，默认为false。
        tts.put("enable_subtitle", true);
        JSONObject payload = new JSONObject();
        // 可选，是否设置回调。如果设置，则服务端在完成长文本语音合成之后回调用户此处设置的回调接口，将请求状态推送给用户侧。
        payload.put("enable_notify", false);
        payload.put("notify_url", "http://123.com");
        payload.put("tts_request", tts);
        JSONObject json = new JSONObject();
        json.put("context", context);
        json.put("header", header);
        json.put("payload", payload);
        String bodyContent = json.toJSONString();
        logger.info("POST Body Content: " + bodyContent);
        // 发起请求
        RequestBody reqBody = RequestBody.create(MediaType.parse("application/json"), bodyContent);
        Request request = new Request.Builder()
            .url(url)
            .header("Content-Type", "application/json")
            .post(reqBody)
            .build();
        try {
            OkHttpClient client = new OkHttpClient();
            Response response = client.newCall(request).execute();
            String contentType = response.header("Content-Type");
            System.out.println("contentType = " + contentType);
            // 获取结果，并根据返回进一步进行处理。
            String result = response.body().string();
            response.close();
            System.out.println("result = " + result);
            JSONObject resultJson = JSON.parseObject(result);
```

```
        if(resultJson.containsKey("error_code") && resultJson.getIntValue("error_code") == 20000000) {
            logger.info("Request Success! task_id = " + resultJson.getJSONObject("data").getString("task_
id"));
            String task_id = resultJson.getJSONObject("data").getString("task_id");
            String request_id = resultJson.getString("request_id");
            /// 可选：轮询检查服务端的合成状态，该轮询操作非必须，如果设置了回调url，则服务端会在合成完成后
主动回调。
            waitLoop4Complete(url, appkey, accessToken, task_id, request_id);
        }else {
            logger.error("Request Error: status=" + resultJson.getIntValue("status")
                + ", error_code=" + resultJson.getIntValue("error_code")
                + ", error_message=" + resultJson.getString("error_message"));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
  /// 根据特定信息轮询检查某个请求在服务端的合成状态，轮询操作非必须，如果设置了回调url，则服务端会在合成
完成后主动回调。
  private void waitLoop4Complete(String url, String appkey, String token, String task_id, String reques
t_id) {
    String fullUrl = url + "?appkey=" + appkey + "&task_id=" + task_id + "&token=" + token + "&request
_id=" + request_id;
    System.out.println("query url = " + fullUrl);
    while(true) {
        Request request = new Request.Builder().url(fullUrl).get().build();
        try {
            OkHttpClient client = new OkHttpClient();
            Response response = client.newCall(request).execute();
            String result = response.body().string();
            response.close();
            System.out.println("waitLoop4Complete = " + result);
            JSONObject resultJson = JSON.parseObject(result);
            if(resultJson.containsKey("error_code")
                && resultJson.getIntValue("error_code") == 20000000
                && resultJson.containsKey("data")
                && resultJson.getJSONObject("data").getString("audio_address") != null) {
                logger.info("Tts Finished! task_id = " + resultJson.getJSONObject("data").getString("task_id
"));
                logger.info("Tts Finished! audio_address = " + resultJson.getJSONObject("data").getString("
audio_address"));
```

```
audio_address "));
                break;
            }else {
                logger.info("Tts Queuing...");
            }
            // 每隔3秒钟轮询一次状态。
            Thread.sleep(3000);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
  }
  public static void main(String[] args) {
    if (args.length < 2) {
      System.err.println("SpeechLongSynthesizerRestfulDemo need params: <token> <app-key>");
      System.exit(-1);
    }
    String token = args[0];
    String appkey = args[1];
    SpeechLongSynthesizerRestfulDemo demo = new SpeechLongSynthesizerRestfulDemo(appkey, to
ken);
    String text = "我家的后面有一个很大的园，相传叫作百草园。现在是早已并屋子一起卖给朱文公的子孙了，连那
最末次的相见也已经隔了七八年，其中似乎确凿只有一些野草；但那时却是我的乐园。";
    String format = "wav";
    int sampleRate = 16000;
    String voice = "siyue";
    demo.processPOSTRequest(text, format, sampleRate, voice);
  }
}
```

Python示例

> **说明**
>
> - Python 2.x请使用httplib模块；Python 3.x请使用http.client模块。
> - 如果使用阿里云上海ECS内网访问URL，请使用HTTP协议，需要替换如下函数，即将 `HTTPSConnection` 修改为 `HTTPConnection` 。

```
# -*- coding: UTF-8 -*-
# Python 2.x
import httplib
import urllib
```

```
import urllib
import urllib2
import json
import time
class TtsHeader:
    def __init__(self, appkey, token):
        self.appkey = appKey
        self.token = token
    def tojson(self, e):
        return {'appkey': e.appkey, 'token': e.token}
class TtsContext:
    def __init__(self, device_id):
        self.device_id = device_id
    #将序列化函数定义到类中。
    def tojson(self, e):
        return {'device_id': e.device_id}
class TtsRequest:
    def __init__(self, voice, sample_rate, format, enable_subtitle, text):
        self.voice = voice
        self.sample_rate = sample_rate
        self.format = format
        self.enable_subtitle = enable_subtitle
        self.text = text
    def tojson(self, e):
        return {'voice': e.voice, 'sample_rate': e.sample_rate, 'format': e.format, 'enable_subtitle': e.enable
_subtitle, 'text': e.text}
class TtsPayload:
    def __init__(self, enable_notify, notify_url, tts_request):
        self.enable_notify = enable_notify
        self.notify_url = notify_url
        self.tts_request = tts_request
    def tojson(self, e):
        return {'enable_notify': e.enable_notify, 'notify_url': e.notify_url, 'tts_request': e.tts_request.tojso
n(e.tts_request)}
class TtsBody:
    def __init__(self, tts_header, tts_context, tts_payload):
        self.tts_header = tts_header
        self.tts_context = tts_context
        self.tts_payload = tts_payload
    def tojson(self, e):
        return {'header': e.tts_header.tojson(e.tts_header), 'context': e.tts_context.tojson(e.tts_context),
```

```
'payload': e.tts_payload.tojson(e.tts_payload)}
# 根据特定信息轮询检查某个请求在服务端的合成状态，轮询操作非必须，如果设置了回调url，则服务端会在合成完成后主动回调。
def waitLoop4Complete(url, appkey, token, task_id, request_id):
    fullUrl = url + "?appkey=" + appkey + "&task_id=" + task_id + "&token=" + token + "&request_id=" + request_id
    print "fullUrl=", fullUrl
    host = {"Host":"nls-gateway.cn-shanghai.aliyuncs.com", "Accept":"*/*", "Connection":"keep-alive",'Content-Type': 'application/json'}
    while True:
        req = urllib2.Request(url=fullUrl)
        result = urllib2.urlopen(req).read()
        print "query result = ", result
        jsonData = json.loads(result)
        if jsonData.has_key("error_code") and jsonData["error_code"] == 20000000 and jsonData.has_key("data") and (jsonData["data"]["audio_address"] != ""):
            print "Tts Finished! task_id = " + jsonData["data"]["task_id"]
            print "Tts Finished! audio_address = " + jsonData["data"]["audio_address"]
            break
        else:
            print "Tts Running..."
            time.sleep(3)
# 长文本语音合成restful接口，支持post调用，不支持get请求。发出请求后，可以轮询状态或者等待服务端合成后自动回调（如果设置了回调参数）。
def requestLongTts4Post(tts_body, appkey, token):
    host = 'nls-gateway.cn-shanghai.aliyuncs.com'
    url = 'https://' + host + '/rest/v1/tts/async'
    # 设置HTTP Headers
    http_headers = {'Content-Type': 'application/json'}
    print('The POST request body content: ' + tts_body)
    conn = httplib.HTTPSConnection(host)
    #conn = httplib.HTTPConnection(host)
    conn.request(method='POST', url=url, body=tts_body, headers=http_headers)
    response = conn.getresponse()
    print('Response status and response reason:')
    print(response.status ,response.reason)
    contentType = response.getheader('Content-Type')
    print(contentType)
    body = response.read()
    if response.status == 200:
        jsonData = json.loads(body)
```

```
        print 'The request succeed : ', jsonData
        print 'error_code = ', jsonData['error_code']
        task_id = jsonData['data']['task_id']
        request_id = jsonData['request_id']
        print 'task_id = ', task_id
        print 'request_id = ', request_id
        # 说明：轮询检查服务端的合成状态，轮询操作非必须。如果设置了回调url，则服务端会在合成完成后主动回调
。
        waitLoop4Complete(url, appkey, token, task_id, request_id)
    else:
        print('The request failed: ' + str(body))
appKey = 'yourAppkey'
token = 'yourToken'
text = '今天是周一，天气挺好的。'
# 拼接HTTP Post请求的消息体内容。
th = TtsHeader(appKey, token)
tc = TtsContext("mydevice")
# TtsRequest对象内容为：发音人、采样率、语音格式、待合成文本内容。
tr = TtsRequest("xiaoyun", 16000, "wav", False, text)
# 是否设置回调url，回调url地址，TtsRequest对象。
tp = TtsPayload(True, "http://134.com", tr)
tb = TtsBody(th, tc, tp)
body = json.dumps(tb, default=tb.tojson)
requestLongTts4Post(str(body), appKey, token)
```

PHP示例

> **说明**
>
> PHP示例中使用了cURL函数，要求PHP的版本为4.0.2以上，并且确保已安装cURL扩展。

```
<?php
    // 根据特定信息轮询检查某个请求在服务端的合成状态，轮询操作非必须，如果设置了回调url，则服务端会在合成
完成后主动回调。
    function waitLoop4Complete($url, $appkey, $token, $task_id, $request_id) {
        $fullUrl = $url . "?appkey=" . $appkey . "&task_id=" . $task_id . "&token=" . $token . "&request_id=" .
$request_id;
        print "query url = " . $fullUrl . "\n";
        while(true) {
            $curl = curl_init();
            curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);
```

```
        curl_setopt($curl, CURLOPT_URL, $fullUrl);
        curl_setopt($curl, CURLOPT_HEADER, TRUE);
        curl_setopt($curl, CURLOPT_SSL_VERIFYPEER, FALSE);
        curl_setopt($curl, CURLOPT_SSL_VERIFYHOST, FALSE);
        $response = curl_exec($curl);
        if ($response == FALSE) {
            print "curl_exec failed!\n";
            curl_close($curl);
            return ;
        }
        // 处理服务端返回的响应。
        $headerSize = curl_getinfo($curl, CURLINFO_HEADER_SIZE);
        $headers = substr($response, 0, $headerSize);
        $bodyContent = substr($response, $headerSize);
        print $bodyContent."\n";
        $http_code = curl_getinfo($curl, CURLINFO_HTTP_CODE);
        if($http_code != 200) {    // 如果请求失败，需要检查调用是否正确。
            print "tts request failure, error code = " . $http_code . "\n";
            curl_close($curl);
            return ;
        }
        curl_close($curl);
        $data = json_decode($bodyContent, true);
        if(isset($data["error_code"]) && $data["error_code"] == 20000000
            && isset($data["data"]) && $data["data"]["audio_address"] != "") {
            print "Tts Finished! task_id = " . $data["data"]["task_id"] . "\n";
            print "Tts Finished! audio_address = " . $data["data"]["audio_address"] . "\n";
            break;
        } else {
            print "Tts Queuing..." . "\n";
        }
        // 每隔3秒钟轮询一次状态。
        sleep(3);
    }
}
// 长文本语音合成restful接口，支持post调用，不支持get请求。发出请求后，可以轮询状态或者等待服务端合成后自动回调（如果设置了回调参数）。
function requestLongTts4Post($appkey, $token, $text, $voice, $format, $sampleRate) {
    $url = "https://nls-gateway.cn-shanghai.aliyuncs.com/rest/v1/tts/async";
    // 拼接HTTP Post请求的消息体内容。
```

```
    $header = array("appkey" => $appkey, "token" => $token);
    $context = array("device_id" => "my_device_id");
    $tts_request = array("text" => $text, "format" => $format, "voice" => $voice, "sample_rate" => $sa
mpleRate, "enable_subtitle" => false);
    $payload = array("enable_notify" => true, "notify_url" => "http://123.com", "tts_request" => $tts_r
equest);
    $tts_body = array("context" => $context, "header" => $header, "payload" => $payload);
    $body = json_encode($tts_body);
    print "The POST request body content: " . $body . "\n";
    $httpHeaders = array("Content-Type: application/json");
    $curl = curl_init();
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);
    curl_setopt($curl, CURLOPT_URL, $url);
    curl_setopt($curl, CURLOPT_POST, TRUE);
    curl_setopt($curl, CURLOPT_HTTPHEADER, $httpHeaders);
    curl_setopt($curl, CURLOPT_POSTFIELDS, $body);
    curl_setopt($curl, CURLOPT_HEADER, TRUE);
    $response = curl_exec($curl);
    if ($response == FALSE) {
        print "curl_exec failed!\n";
        curl_close($curl);
        return ;
    }
    $headerSize = curl_getinfo($curl, CURLINFO_HEADER_SIZE);
    $headers = substr($response, 0, $headerSize);
    $bodyContent = substr($response, $headerSize);
    $http_code = curl_getinfo($curl, CURLINFO_HTTP_CODE);
    if($http_code != 200) {
        print "tts request failure, error code = " . $http_code . "\n";
        print "tts request failure, response = " . $bodyContent . "\n";
        return ;
    }
    curl_close($curl);
    print $bodyContent . "\n";
    $data = json_decode($bodyContent, true);
    if( isset($data["error_code"]) && $data["error_code"] == 20000000) {
        $task_id = $data["data"]["task_id"];
        $request_id = $data["request_id"];
        print "Request Success! task_id = " . $task_id . "\n";
        print "Request Success! request_id = " . $request_id . "\n";
        ///说明：轮询检查服务端的合成状态，轮询操作非必须，如果设置了回调url，则服务端会在合成完成后主动回
```

```
调。
        waitLoop4Complete($url, $appkey, $token, $task_id, $request_id);
    } else {
        print "Request Error: status=" . $data["status"] . "; error_code=" . $data["error_code"] . "; error_message=" . $data["error_message"] . "\n";
    }
  }
$appkey = "yourAppkey";
$token = "yourToken";
$text = "今天是周一，天气挺好的。";
$format = "wav";
$voice = "xiaogang";
$sampleRate = 16000;
requestLongTts4Post($appkey, $token, $text, $voice, $format, $sampleRate);
?>
```

Node.js示例

> **说明**
>
> 首先安装request依赖，请在您的示例文件所在目录执行如下命令：
>
> ```
> npm install request --save
> ```

```
const request = require('request');
const fs = require('fs');
  // 长文本语音合成restful接口，支持post调用，不支持get请求。发出请求后，可以轮询状态或者等待服务端合成后自动回调（如果设置了回调参数）。
  function requestLongTts4Post(textValue, appkeyValue, tokenValue, voiceValue, formatValue, sampleRateValue) {
    var url = "https://nls-gateway.cn-shanghai.aliyuncs.com/rest/v1/tts/async"
    console.log(url);
    // 请求参数，以JSON格式字符串填入HTTPS POST请求的Body中。
    var context = {
      device_id : "device_id",
    };
    var header = {
      appkey : appkeyValue,
      token : tokenValue,
    };
    var tts_request = {
```

```
var tts_request = {
    text : textValue,
    voice : voiceValue,
    format : formatValue,
    "sample_rate" : sampleRateValue,
    "enable_subtitle" : false
};
var payload = {
    "enable_notify" : false,
    "notify_url": "http://123.com",
    "tts_request" : tts_request,
};
var tts_body = {
    "context" : context,
    "header" : header,
    "payload" : payload
};
var bodyContent = JSON.stringify(tts_body);
console.log('The POST request body content: ' + bodyContent);
// 设置HTTPS POST请求头部。
var httpHeaders = {'Content-type' : 'application/json'};
// 设置HTTPS POST请求。
// encoding必须设置为null，HTTPS响应的Body为二进制Buffer类型。
var options = {
    url: url,
    method: 'POST',
    headers: httpHeaders,
    body: bodyContent,
    encoding: null
};
request(options, function (error, response, body) {
    // 处理服务端的响应。
    if (error != null) {
        console.log(error);
    } else {
        if(response.statusCode != 200) {
            console.log("Http Request Fail: " + response.statusCode + "; " + body.toString());
            return;
        }
        console.log("response result: " + body.toString());
        var code = 0;
```

```
            var task_id = "";
            var request_id = "";
            var json = JSON.parse(body.toString());
            console.info(json);
            for(var key in json){
              if(key=='error_code'){
                code = json[key]
              } else if(key=='request_id'){
                request_id = json[key]
              } else if(key == "data") {
                task_id = json[key]["task_id"];
              }
            }
            if(code == 20000000) {
              console.info("Request Success! task_id = " + task_id);
              console.info("Request Success! request_id = " + request_id);
              /// 说明：轮询检查服务端的合成状态，轮询操作非必须，如果设置了回调url，则服务端会在合成完成后主动回调。
              waitLoop4Complete(url, appkey, token, task_id, request_id);
            } else {
              console.info("Request Error: status=" + $data["status"] + "; error_code=" + $data["error_code"] + "; error_message=" + $data["error_message"]);
            }
          }
        });
    }
    // 根据特定信息轮询检查某个请求在服务端的合成状态，轮询操作非必须，如果设置了回调url，则服务端会在合成完成后主动回调。
    function waitLoop4Complete(urlValue, appkeyValue, tokenValue, task_id_value, request_id_value) {
      var fullUrl = urlValue + "?appkey=" + appkeyValue + "&task_id=" + task_id_value + "&token=" + tokenValue + "&request_id=" + request_id_value;
      console.info("query url = " + fullUrl);
      //while(true) {
      var timer = setInterval(() => {
        var options = {
          url: fullUrl,
          method: 'GET'
        };
        console.info("query url = " + fullUrl);
        request(options, function (error, response, body) {
          // 处理服务端的响应。
```

```
            if (error != null) {
                console.log(error);
            } else if(response.statusCode != 200) {
                console.log("Http Request Fail: " + response.statusCode + "; " + body.toString());
                return;
            } else {
                console.log("query result: " + body.toString());
                var code = 0;
                var task_id = "";
                var output_url = "";
                var json = JSON.parse(body.toString());
                console.info(json);
                for(var key in json){
                    if(key=='error_code'){
                        code = json[key]
                    } else if(key=='request_id'){
                        request_id = json[key]
                    } else if(key == "data" && json["data"] != null) {
                        task_id = json[key]["task_id"];
                        audio_address = json[key]["audio_address"];
                    }
                }
                if(code == 20000000 && audio_address != "") {
                    console.info("Tts Finished! task_id = " + task_id);
                    console.info("Tts Finished! audio_address = " + audio_address);
                    // 退出轮询。
                    clearInterval(timer);
                } else {
                    console.info("Tts Queuing...");
                }
            }
        }
        );
    }
  , 3000);
  }
var appkey = 'yourAppkey';
var token = 'yourToken';
var text = '今天是周一，天气挺好的。';
var voice = "xiaogang";
var format = 'wav';
```

```
var format = 'wav';
var sampleRate = 16000;
requestLongTts4Post(text, appkey, token, voice, format, sampleRate);
```

Go示例

```
package main
import (
    "fmt"
    "net/http"
    "io/ioutil"
    "encoding/json"
    "strconv"
    "bytes"
    "time"
)
// 长文本语音合成restful接口，支持post调用，不支持get请求。发出请求后，可以轮询状态或者等待服务端合成后自动回调（如果设置了回调参数）。
func requestLongTts4Post(text string, appkey string, token string) {
    // var url string= "http://pre-nls-gateway.aliyuncs.com:80/rest/v1/tts/async"
    var url string= "https://nls-gateway.cn-shanghai.aliyuncs.com/rest/v1/tts/async"
    // 拼接HTTP Post请求的消息体内容。
    context := make(map[string]interface{})
    context["device_id"] = "test-device"
    header := make(map[string]interface{})
    header["appkey"] = appkey
    header["token"] = token
    tts_request := make(map[string]interface{})
    tts_request["text"] = text
    tts_request["format"] = "wav"
    tts_request["voice"] = "xiaogang"
    tts_request["sample_rate"] = 16000
    tts_request["enable_subtitle"] = false
    payload := make(map[string]interface{})
    payload["enable_notify"] = false
    payload["notify_url"] = "http://123.com"
    payload["tts_request"] = tts_request
    ttsBody := make(map[string]interface{})
    ttsBody["context"] = context
    ttsBody["header"] = header
    ttsBody["payload"] = payload
```

```
    ttsBodyJson, err := json.Marshal(ttsBody)
    if err != nil {
        panic(nil)
    }
    fmt.Println(string(ttsBodyJson))
    // 发送HTTPS POST请求，处理服务端的响应。
    response, err := http.Post(url, "application/json;charset=utf-8", bytes.NewBuffer([]byte(ttsBodyJson
)))
    if err != nil {
        panic(err)
    }
    defer response.Body.Close()
    contentType := response.Header.Get("Content-Type")
    body, _ := ioutil.ReadAll(response.Body)
    fmt.Println(string(contentType))
    fmt.Println(string(body))
    statusCode := response.StatusCode
    if(statusCode == 200) {
        fmt.Println("The POST request succeed!")
        var f interface{}
        err := json.Unmarshal(body, &f)
        if err != nil {
            fmt.Println(err)
        }
        // 从消息体中解析出来task_id（重要）和request_id。
        var taskId string = ""
        var requestId string = ""
        m := f.(map[string]interface{})
        for k, v := range m {
            if(k == "error_code") {
                fmt.Println("error_code = ", v)
            } else if(k == "request_id") {
                fmt.Println("request_id = ", v)
                requestId = v.(string)
            } else if(k == "data") {
                fmt.Println("data = ", v)
                data := v.(map[string]interface{})
                fmt.Println(data)
                taskId = data["task_id"].(string)
            }
        }
```

```
        // 说明：轮询检查服务端的合成状态，轮询操作非必须，如果设置了回调url，则服务端会在合成完成后主动回调
。
        waitLoop4Complete(url, appkey, token, taskId, requestId)
    } else {
        fmt.Println("The POST request failed: " + string(body))
        fmt.Println("The HTTP statusCode: " + strconv.Itoa(statusCode))
    }
}
// 根据特定信息轮询检查某个请求在服务端的合成状态，轮询操作非必须，如果设置了回调url，则服务端会在合成完
成后主动回调。
func waitLoop4Complete(url string, appkey string, token string, task_id string, request_id string) {
    var fullUrl string = url + "?appkey=" + appkey + "&task_id=" + task_id + "&token=" + token + "&reques
t_id=" + request_id
    fmt.Println("fullUrl=" + fullUrl)
    for {
        response, err := http.Get(fullUrl)
        if err != nil {
            fmt.Println("The GET request failed!")
            panic(err)
        }
        defer response.Body.Close()
        body, _ := ioutil.ReadAll(response.Body)
        fmt.Println("waitLoop4Complete = ", string(body))
        var f interface{}
        json.Unmarshal(body, &f)
        if err != nil {
            fmt.Println(err)
        }
        // 从消息体中解析出来task_id（重要）和request_id。
        var code float64 = 0
        var taskId string = ""
        var audioAddress string = ""
        m := f.(map[string]interface{})
        for k, v := range m {
            if(k == "error_code") {
                code = v.(float64)
            } else if(k == "request_id") {
            } else if(k == "data") {
                if (v != nil) {
                    data := v.(map[string]interface{})
                    taskId = data["task_id"].(string)
```

```
                taskId = data["task_id"].(string)
                audioAddress = data["audio_address"].(string)
            }
        }
    }
    if (code == 20000000 && audioAddress != "") {
        fmt.Println("Tts Finished! task_id = " + taskId);
        fmt.Println("Tts Finished! audio_address = " + audioAddress);
        break
    } else {
        time.Sleep(time.Duration(3)*time.Second)
    }
  }
}
func main() {
  var appkey string = "yourAppkey"
  var token  string = "yourToken"
  var text string = "今天是周一，天气挺好的。"
  requestLongTts4Post(text, appkey, token)
}
```

5.SSML标记语言

5.1. 标记语言介绍

本文为您介绍SSML（Speech Synthesis Markup Language）标记语言的功能、使用方式及示例。

概述

SSML是一种基于XML的语音合成标记语言。 与纯文本的合成相比，使用SSML可以充实合成的内容，为最终合成效果带来更多变化。SSML不仅让人控制语音合成能读什么，更能控制语音合成可以怎么读，包括控制断句分词方式、发音、速度、停顿、声调和音量等特征，甚至加入背景音乐。

> 说明
>
> 阿里巴巴语音合成服务的SSML实现基于W3C的语音合成标记语言版本1.0。但并不支持W3C包含的所有的标记类型，而是从业务角度出发，将支持的标记类型最大程度与业务需求绑定。

使用方式

> 说明
>
> - 语音合成服务支持的SSML标签请参见标签。
> - 目前只有中文合成支持SSML功能。
> - 长文本语音合成的ssml功能支持多个<speak>标签闭合嵌套于文本之中，如：
>
> ```
> 你好吗？<speak>
> <say-as interpret-as="telephone">114</say-as>查询号码
> <say-as interpret-as="cardinal">123</say-as>开始干。
> 加起来为<say-as interpret-as="digits">1234</say-as>。
> <say-as interpret-as="name">薄斌</say-as>的快递。
> <say-as interpret-as="address">富路国际1号楼3单元304</say-as>
> <say-as interpret-as="nick">王永花6689</say-as>
> </speak>我很好。<speak>
> 车牌是<say-as interpret-as="verbatim">浙AX88888</say-as>
> </speak>哈哈哈
> ```

将带标签的文本作为text参数值，上传至语音合成服务，以Java SDK为例：

```
SpeechSynthesizer synthesizer = new SpeechSynthesizer(client, getSynthesizerListener());
String text = "<speak>请闭上眼睛休息一下<break time=\"500ms\"/>好了，请睁开眼睛。</speak>";
synthesizer.setText(text);
```

发送给语音合成服务的请求内容如下：

```
{
  "payload": {
    "volume": 50,
    "sample_rate": 16000,
    "format": "wav",
    "text": "<speak>请闭上眼睛休息一下<break time=\"500ms\"/>好了，请睁开眼睛。</speak>"
  },
  "context": {
    "sdk": {
      "name": "nls-sdk-java",
      "version": "2.0.4"
    }
  },
  "header": {
    "namespace": "SpeechSynthesizer",
    "name": "StartSynthesis",
    "message_id": "5fdf78c0dd574b6897f3cb204dd0****",
    "appkey": "f6OslY8nCPOa****",
    "task_id": "6e1be78ef5804c50a2c5a8b92de1****"
  }
}
```

标签

<speak>

- 描述

  <speak>标签是所有待支持SSML标签的根节点。一切需要调用SSML标签的文本都要包含在<speak></speak>中。

- 语法

  ```
  <speak>需要调用SSML标签的文本</speak>
  ```

- 属性

  <speak>标签支持如下属性。

属性名称	属性类型	属性值	是否必选	描述

属性名称	属性类型	属性值	是否必选	描述
voice	String	线上可调用的发音人的名称代号，为全小写的voice参数值， 如”siyue”。	否	阿里巴巴语音合成特有标签。在合成时，指定发音人，优先级高于接口请求参数 `voice` 指定的发音人。 **注意** “xiaoyue”暂不支持在voice中设置。
encodeType	String	PCM/WAV/MP3	否	阿里巴巴语音合成特有标签。在合成时，指定音频文件格式，优先级高于接口请求参数 `format` 指定的文件格式。
sampleRate	String	8000/16000	否	阿里巴巴语音合成特有标签。在合成时，指定音频的采样率，优先级高于接口请求参数 `sample_rate` 指定的音频采样率。
rate	String	[-500,500]之间整数。默认值为0。 ○ 大于0表示加快语速。 ○ 小于0表示减慢语速。	否	阿里巴巴语音合成特有标签。在合成时，指定音频的语速，优先级高于接口请求参数 `speech_rate` 指定的语速。
pitch	String	[-500,500]之间整数。默认值为0。 ○ 大于0表示升高音高。 ○ 小于0表示降低音高。	否	阿里巴巴语音合成特有标签。在合成时，指定音频的音高，优先级高于接口请求参数 `pitch_rate` 指定的音高。

属性名称	属性类型	属性值	是否必选	描述
volume	String	[0,100]之间整数。默认值为50。 ○ 大于50表示增大音量。 ○ 小于50表示减小音量。	否	阿里巴巴语音合成特有标签。在合成时，指定音频的音量，优先级高于接口请求参数 `volume` 指定的音量。
effect	String	robot/lolita/lowpass/echo/eq/lpfilter/hpfilter	否	阿里巴巴语音合成特有标签。使用该标签可以使合成的语音产生不同的声音效果。 ○ robot：机器人音效。 ○ lolita：萝莉音效。 ○ lowpass：低通音效。 ○ echo：回声音效。 ○ eq：均衡器。 ○ lpfilter：低通滤波器。 ○ hpfilter：高通滤波器。 **说明** ○ 其中eq、lpfilter、hpfilter是高级效果器，您可以通过 `effectValue` 自定义效果器的效果。 ○ 一个SSML只支持一种音效，不可以写多个effect属性。 ○ 选择使用音效功能会增加系统延时。

属性名称	属性类型	属性值	是否必选	描述
effectValue	String	当effect取值为eq/lpfilter/hpfilter时，使用该参数修改效果器的默认效果。	否	○ eq（均衡器）：系统默认使用8个等级。对应的频率为[“40Hz”，“100Hz”，“200Hz”，“400Hz”，“800Hz”，“1600Hz”，“4000Hz”，“12000Hz”]；对应的带宽为[“1.0q”，“1.0q”，“1.0q”，“1.0q”，“1.0q”，“1.0q”，“1.0q”，“1.0q”]。在使用过程中，需要输入8个等级对应的增益，其取值范围为[-20dB, 20dB]。例如，effectValue=”1 1 1 1 1 1 1 1”。是一个以空格分割的8个整数组成的字符串。数值为0表示不调整对应频率的gain。 ○ lpfilter：输入低通滤波器的频率值。取值为(0, 目标采样率/2]之间的整数。例如effectValue=”800”。 ○ hpfilter：输入高通滤波器的频率值。取值为(0, 目标采样率/2]之间的整数。例如effectValue=“1200”。
bgm	String	线上可调用的背景音乐的名称。参见bgm属性说明。	否	阿里巴巴语音合成特有标签。为合成的语音添加指定的背景音乐。
backgroundMusicVolume	String	[0,100]之间的整数。默认值为50。 ○ 大于50表示增大音量。 ○ 小于50表示减小音量。	否	阿里巴巴语音合成特有标签。控制背景音乐的音量。

其中，bgm属性说明如下。

服务内嵌URL	自定义背景音URL
目前阿里巴巴语音合成服务内嵌如下几款背景音乐供您体验： ○ http://nls.alicdn.com/bgm/1.wav ○ http://nls.alicdn.com/bgm/2.wav ○ http://nls.alicdn.com/bgm/3.wav	您可以根据需求，使用自定义的背景音。需要将背景音存放在阿里云的OSS上，并且所在的存储空间至少为公共读权限，请参见创建存储空间。使用HTTP/HTTPS协议生成文件访问链接，请参见管理Object章节中生成文件URL。 音频要求： ○ 采样率16 KHz、单声道WAV格式。 ○ 不超过2 MB。 说明 ○ 合成时长超出背景音时长时，背景音将随合成音频循环播放（如果背景音不是WAV格式，可使用ffmpeg将其转为WAV格式：`ffmpeg -i 输入音频 -acodec pcm_s16le -ac 1 -ar 16000 目标.wav`）。 ○ 标签内的URL如果包含XML的特殊字符，需要做字符转义。常用的共有5个：<、>、&、"和'。 注意 您需要对上传的音频版权承担相应的法律责任。

- 标签关系

  <speak>标签可以包含文本和以下标签：

  - <break>
  - <s>
  - <w>
  - <phoneme>
  - <say-as>
  - 示例
    - 空属性

      ```
      <speak>
       需要调用SSML标签的文本
      </speak>
      ```

      音频效果：SSML-speak1.mp3

- voice属性

```
<speak voice="xiaogang">
 我是男声。
</speak>
```

音频效果：SSML-speak2.mp3

- encodeType属性

```
<speak encodeType="mp3">
 我可以生成压缩格式的音频。
</speak>
```

音频效果：SSML-encode.mp3

- sampleRate属性

```
<speak sampleRate="8000">
 看看我的文件大小吧，是16000采样率音频的一半。
</speak>
```

音频效果：SSML-speak4.mp3

- rate属性

```
<speak rate="200">
 我的语速比正常人快。
</speak>
```

音频效果：SSML-speak5.mp3

- pitch属性

```
<speak pitch="-100">
 我的音高却比别人低。
</speak>
```

音频效果：SSML-speak6.mp3

- volume属性

```
<speak volume="80">
 我的音量也很大。
</speak>
```

音频效果：SSML-speak7.mp3

- 属性组合（空格分隔）

```
<speak rate="200" pitch="-100" volume="80">
 所以放在一起，我的声音是这样的。
</speak>
```

音频效果：SSML-speak8.mp3

- effect属性

```
<speak effect="robot">
 你喜欢机器人瓦力吗？
</speak>
```

音频效果：SSML-speak9.mp3

- bgm属性

```
<speak bgm="http://nls.alicdn.com/bgm/2.wav" backgroundMusicVolume="30" rate="-500" volu
me="40">
 <break time="2s"/>
 阴崖老木苍苍烟
 <break time="700ms"/>
 雨声犹在竹林间
 <break time="700ms"/>
 绵蕝固知裨国计
 <break time="700ms"/>
 绵州风物总堪怜
 <break time="2s"/>
</speak>
```

音频效果：SSML-speak10.mp3

<break>

- 描述

  用于在文本中插入停顿，该标签是可选标签。

- 语法

```
# 无属性
<break/>
# 带time属性
<break time="string"/>
```

- 属性

> 说明
> 使用无属性的<break>标签时，停顿时长为“1s”。

属性名称	属性类型	属性值	是否必选	描述
time	String	[number]s /[number] ms	否	以秒/毫秒为单位设置停顿的时长 (如“2s“、“50ms”)。 ■ [number]s：以秒为单位，[number]取值范围为[50, 10000]的整数。 ■ [number]ms：以毫秒为单位，[number]取值范围为[1, 10]的整数。

- 标签关系

  <break>是空标签，不能包含任何标签。如果SSML结构中存在<s>标签，请把<break>写在<s>里面，表示对当前段落或句子设置停顿。

- 示例

```
<speak>
  请闭上眼睛休息一下<break time="500ms"/>好了，请睁开眼睛。
</speak>
```

  音频效果：SSML-break.mp3

<s>

- 描述

  用于表示文本的句子结构，该标签是可选标签。

- 语法

```
<s>文本</s>
```

- 属性

  无。

- 标签关系

  <s>标签可以包含文本和以下标签：

  ■ <break>
  ■ <w>
  ■ <phoneme>
  ■ <say-as>

- 示例

```
<speak><s>这是第一句话</s><s>这是第二句话</s></speak>
```

音频效果：SSML-s.mp3

<sub>

- 描述

使用别名替换标签内文本。

- 语法

```
<sub alias="string"></sub>
```

- 属性

属性名称	属性类型	属性值	是否必选	描述
alias	String	替换后的内容	是	用于替换标签内的文本。

- 标签关系

<sub>标签仅包括文本。

- 示例

```
<speak><sub alias="网络协议标准">W3C</sub></speak>
```

音频效果：SSML-sub.mp3

<w>

- 描述

用于表示文本的词语结构，该标签是可选标签。

- 语法

```
<w>文本</w>
```

- 属性

无。

- 标签关系

<w>标签仅包括文本。

- 示例

```
<speak>南京市长<w>江大桥</w>今天发表了演讲。</speak>
```

音频效果：SSML-w.mp3

<phoneme>

- 描述

  用于控制标签内文本的读音，该标签是可选标签。

- 语法

```
<phoneme alphabet="string" ph="string">文本</phoneme>
```

- 属性

属性名称	属性类型	属性值	是否必选	描述
alphabet	String	py	是	“py”表示拼音。
ph	String	标签内文本对应的拼音串	是	拼音用法的赋值规范： ■ 字与字的拼音用空格分隔，拼音的数目必须与字数相等。 ■ 每个拼音由发音和音调组成，音调为1~5的数字编号，其中”5”表示轻声。

- 标签关系

  <phoneme>标签仅包括文本。

- 示例

```
<speak>
  去<phoneme alphabet="py" ph="dian3 dang4 hang2">典当行</phoneme>把这个玩意<phoneme al
phabet="py" ph="dang4 diao4">当掉</phoneme>
</speak>
```

  音频效果：SSML-phoneme.mp3

<soundEvent>

- 描述

  提示音标签，可以在SSML合成过程中，通过该标签在任意位置插入提示音。

- 语法

```
<soundEvent src="URL"/>
```

- 属性

属性名称	属性类型	属性值	是否必选	描述
src	String	URL提示音资源路径	是	您可以根据需求，使用自定义提示音。需要将提示音存放在阿里云OSS上，并且所在的存储空间至少为公共读权限，请参见创建存储空间，使用HTTP/HTTPS协议生成文件访问链接请参见管理Object章节中生成文件URL。 音频要求： ■ 采样率16K Hz、单声道WAV格式。 ■ 不超过2 MB。 注意 您需要对上传的音频版权承担相应的法律责任。

- 标签关系

  <soundEvent>是空标签，不可以包含任何标签。

- 示例

```
<speak>
  一匹马受了惊吓<soundEvent src="http://nls.alicdn.com/sound-event/horse-neigh.wav"/>人们四散躲避
</speak>
```

音频效果：SSML-sound-event.mp3

<say-as>

- 描述

  用于指示出标签内文本的信息类型，进而按照该类型的默认发音方式发音。

- 语法

```
<say-as interpret-as="string">文本</say-as>
```

- 属性

属性名称	属性类型	属性值	是否必选	描述
interpret-as	String	cardinal/digits/telephone/name/address/id/characters/punctuation/date/time/currency/measure	是	指示出标签内文本的信息类型： ■ cardinal：按整数或小数发音。 ■ digits：按数字发音。 ■ telephone：按电话号码常用方式发音。 ■ name：按人名发音。 ■ address：按地址发音。 ■ id：适用于账户名、昵称等。 ■ characters：将标签内的文本按字符一一读出。 ■ punctuation：将标签内的文本按标点符号的方式读出来。 • date：按日期发音。 • time：按时间发音。 • currency：按金额发音。 • measure：按计量单位发音。

- 各<say-as>类型支持范围

- cardinal

格式	示例	输出	说明
数字串	145	一百四十五	整数输入范围：20位以内的正负整数，[-99999999999999999999,9999999999999999999]。 小数输入范围：对小数点后小数的位数没有特殊限制，建议不超过10位。
负号+数字串	-145	负一百四十五	
以逗号分隔3位数字串	10,000	一万	
负号+以逗号分隔3位数字串	-10,124	负一万一百二十四	
数字串+小数点+2个零	10.00	十	
负号+数字串+小数点+2个零	-110.00	负一百一十	
数字串+小数点+数字串	79.090	七十九点零九零	
负号+数字串+小数点+数字串	-79.001	负七十九点零零一	

- digits

格式	示例	输出	说明
数字串	129090909	一二九零九零九零九	对数字串的长度没有特殊限制。 建议不超过20位，且当数字串超过10位时，每个数字后插入停顿。

- telephone

格式	示例	输出	说明

格式	示例	输出	说明
座机号	4930286	四九三 零二八六	支持7~8位座机号，支持空格和'-'作为分隔符。 其中：7位座机号支持“3-4”的数字分隔方式。8位座机号支持“4-4”的数字分隔方式。
	493 0286	四九三 零二八六	
	493-0286	四九三 零二八六	
	62552560	六二五五 二五六零	
	6255 2560	六二五五 二五六零	
	6255-2560	六二五五 二五六零	
座机号+分机号	4930286-109	四九三 零二八六 转幺零九	支持1~4位分机号。
	4930286转109	四九三 零二八六 转幺零九	
	4930286分机109	四九三 零二八六 分机幺零九	
	4930286分机号109	四九三 零二八六 分机号幺零九	
	01062552560	零幺零 六二五五 二五六零	
	010 62552560	零幺零 六二五五 二五六零	
	010 6255 2560	零幺零 六二五五 二五六零	
	010 6255-2560	零幺零 六二五五 二五六零	

格式	示例	输出	说明
区号+座机号	010-62552560	零幺零 六二五五 二五六零	支持区号：010、02x、03xx、04xx、05xx、07xx、08xx、09xx。
	010-6255-2560	零幺零 六二五五 二五六零	
	(010)62552560	零幺零 六二五五 二五六零	
	03198907098	零三幺九 八九零 七零九八	
	0319-8907098	三幺九 八九零 七零九八	
区号+座机号+分机号	010 62552560-109	零幺零 六二五五 二五六零 转幺零九	无
	010-62552560-109	零幺零 六二五五 二五六零 转幺零九	
	(010)62552560-109	零幺零 六二五五 二五六零 转幺零九	
	(010)62552560转109	零幺零 六二五五 二五六零 转幺零九	
	(010)62552560分机109	零幺零 六二五五 二五六零 分机幺零九	
	(010)62552560分机号109	零幺零 六二五五 二五六零 分机号幺零九	
	86-010-62791627	八六 零幺零 六二七九 幺六二七	
	(86)10-62791627	八六 幺零 六二七九 幺六二七	

格式 国家代码+区号+座机号	示例	输出	说明
	+86-010-62791627	八六 零幺零 六二七九 幺六二七	支持国家代码：86、(86)、+86、(+86)、0086。并统一读为“八六”。
	0086-10-62791627	八六 幺零 六二七九 幺六二七	
	(+86)-10-6279 1627	八六 幺零 六二七九 幺六二七	
国家代码+区号+座机号+分机号	(86)21-58118818-207	八六 二幺 五八幺幺 八八幺八 转二零七	无
	(86)021-5811-8818-207	八六 零二幺 五八幺幺 八八幺八 转二零七	
	(86)021-58118818转207	八六 零二幺 五八幺幺 八八幺八 转二零七	
	(86)21-5811-8818分机207	八六 二幺 五八幺幺 八八幺八 分机二零七	
	+86-021-58118818分机号207	八六 零二幺 五八幺幺 八八幺八分机号二零七	
手机号	151 9099 0987	幺五幺 九零九九 零九八七	支持11位手机号，支持3-3-5,3-4-4两种数字分隔方式
	151-909-90987	幺五幺 九零九 九零九八七	
	151 909 90987	幺五幺 九零九 九零九八七	
	+86-15190990987	八六 幺五幺 九零九九 零九八七	
	(+86)-151-9099-0987	八六 幺五幺 九零九九 零九八七	

格式	示例	输出	说明
国家代码+手机号	+8615190990987	八六 幺五幺 九零九九 零九八七	
	0086-151 909 90987	八六 幺五幺 九零九 九零九八七	
服务号	110	幺幺零	■ 支持常用的服务号如110。 ■ 支持以400/800开头的10位服务号，支持以“3-3-4”的数字分隔方式。 ■ 支持以12530/17951/12593开头的16位号码。
	95566	九五五六六	
	4008110510	四零零 八幺幺 零五幺零	
	800-810-8888	八零零 八幺零 八八八八	
	1253013520638377	幺二五三零 幺三五 二零六三 八三七七	
其他	(86)(21)9899-80800-0909	八六 二幺 九八九九 八零八零零 零九零九	支持“数字串+分隔符（左右括号、-）”方式。

- address

格式	示例	输出	说明
常用地址格式	元和镇嘉元30-9	元和镇嘉元三十杠九	支持常用地址格式。此处地址指标准的邮寄地址。
	市台路388弄1107-1108号	市台路三八八弄幺幺零七杠幺幺零八号	
	华润二十四城六期锦云府3-1-3205	华润二十四城六期锦云府三杠一杠三二零五	
	圣华名都大厦2幢2006室	圣华名都大厦二幢二零零六室	
	五常街道庭院5幢4单元201	五常街道庭院五幢四单元二零幺	
	芙蓉江路150弄19号	芙蓉江路幺五零弄十九号	

- id

格式	示例	输出	说明
字符串	dell0101	D E L L 零 一 零 一	大小写英文字符、阿拉伯数字0~9、下划线。 输出的空格表示每个字符之间插入停顿，即字符一个一个地读。
	myid_1998	M Y I D 下划线 一 九 九八	
	AiTest	A I T E S T	

- characters

格式	示例	输出	说明
字符串	ISBN 1-001-099098-1	I S B N 一 杠 零 零 一 杠 零 九 九 零 九 八 杠 一	支持中文汉字、大小写英文字符、阿拉伯数字0~9以及部分全角和半角字符。 输出的空格表示每个字符之间插入停顿，即字符一个一个地读。标签内的文本如果包含XML的特殊字符，需要做字符转义。常用的共有5个： < > & " ' ，分别对应<、>、& 、"、'。
	x10b2345_u	x 一 零 b 二 三 四 五 下划线 u	
	v1.0.1	v 一 点 零 点 一	
	版本号2.0	版本号二 点 零	
	苏M MA000	苏M M A 零 零 零	
	空中客车A330	空中客车A 三 三 零	
	型号s01 s02和s03	型号s 零 一 s 零二 和s 零 三	
	空中客车A330	空中客车A 三 三 零	
	αβγ	阿尔法 贝塔 伽玛	

- punctuation

格式	示例	输出	说明
标点符号	…	省略号	支持常见中英文标点。输出的空格表示每个字符之间插入停顿，即字符一个一个地读。 标签内的文本如果包含XML的特殊字符，需要做字符转义。常用的共有5个： < > & " ' ，分别对应<、>、& 、"、'。
	……	省略号	
	!"#$%&	叹号 双引号 井号 dollar 百分号 and	
	‘()*+	单引号 左括号 右括号 星号 加号	
	,-./:;	逗号 杠 点 斜杠 冒号 分号	
	<=>?@	小于 等号 大于 问号 at	
	[\]^_	左方括号 反斜线 右方括号 脱字符 下划线	

- date

格式	示例	输出	说明
xx年	71年	七一年	支持2位和4位年份。其中： ■ 2位年份支持60年~99年、00年~09年、10年~19年。 ■ 4位年份支持1000年~1999年、2000年~2099年。
	04年	零四年	
	19年	一九年	
	1011年	一零一一年	
	1998年	一九九八年	

格式	示例	输出	说明
	2008年	二零零八年	
xx年xx月	98年4月	九八年四月	当月份为1到9月时，支持开头带”0”和不带”0”两种写法。例如”1908年4月”和”1908年04月”。
	1998年04月	一九九八年四月	
	08年8月	零八年八月	
	2008年8月	二零零八年八月	
xx年xx月xx日xx年xx月xx号	98年4月23日	九八年四月二十三日	当日期为1到9日时，支持开头带”0”和不带”0”两种写法。例如”1908年4月8日”和”1908年04月08日”。
	1998年04月23日	一九九八年四月二十三日	
	08年8月8号	零八年八月八号	
	2008年08月08号	二零零八年八月八号	
xx年xx月xx日xx年xx月xx号	98年4月23日	九八年四月二十三日	当日期为1到9日时，支持开头带”0”和不带”0”两种写法。例如”1908年4月8日”和”1908年04月08日”。
	1998年04月23日	一九九八年四月二十三日	
	08年8月8号	零八年八月八号	
	2008年08月08号	二零零八年八月八号	
xx月xx号	3月20日	三月二十日	无
	08月07号	八月七号	

格式	示例	输出	说明
年月缩写	2018/08	二零一八年八月	支持”/“、”-“、”.”作为缩写的分隔符。
	2018-08	二零一八年八月	
	2018.08	二零一八年八月	
年月日缩写	2018/08/08	二零一八年八月八日	
	2018-8-8	二零一八年八月八日	
	2018.08.08	二零一八年八月八日	
xx年xx月xx日~xx年xx月xx日xx年xx月xx号~xx年xx月xx号	04年9月1日~30日	零四年九月一日至三十日	支持”~”、”-“作为”至”的缩写标志。
	2004年09月01号-2008年06月08号	二零零四年九月一号至二零零八年六月八号	
xx年xx月xx日~xx日xx年xx月xx号~xx号	04年9月1日~30日	零四年九月一日至三十日	
	2004年09月01号-2008年06月08号	二零零四年九月一号至二零零八年六月八号	
xx年xx月~xx年xx月	01年04月~10年04月	零一年四月至一零年四月	
	2001年04月~2010年04月	二零零一年四月至二零一零年四月	
	10月1日~10月7日	十月一日至十月七日	

格式	示例	输出	说明
xx月xx日~xx月xx日 xx月xx号~xx月xx号	10月01号~10月07号	十月一号至十月七号	
xx月xx日~xx日xx月xx号~xx号	10月1日~7日	十月一日至七日	
	10月01号~07号	十月一号至七号	
年月日缩写~年月日缩写	2018/03/03~2019/01/01	二零一八年三月三日至二零一九年一月一日	支持”/“、”.”作为缩写的分隔符，支持”~””-“作为”至”的缩写标志。
	1997.9.9~1998.9.9	一九九七年九月九日至一九九八年九月九日	
月日缩写~月日缩写	10/20~10/31	十月二十日至十月三十一日	
xx~xx月xx月~xx月	1~10月	一至十月	
	1月~10月	一月至十月	
月日年缩写	10/20/2018	二零一八年十月二十日	仅支持4位的年份，仅支持“/”作为日期的分隔符，仅支持”月/日/年”的书写方式。

- time

格式	示例	输出	说明
时刻	12:00	十二点	
	12:00:00点	十二点	
	10:20分	十点二十分	
	10:20:30	十点二十分三十秒	

格式	示例	输出	说明
			支持常用时间和时间范围格式。
	09:18:14	九点十八分十四秒	
时刻~时刻	11:00~12:00	十一点到十二点	
	09:00-14:00	九点到十四点	
	11:00~11:30	十一点到十一点三十分	
	11:00-12:18	十一点到十二点十八分	
	10:30~11:00	十点三十分到十一点	
	09:28-10:00	九点二十八分到十点	
	10:20~11:20	十点二十分到十一点二十分	
	06:00~08:00	六点到八点	
	上午10:20~下午13:30	上午十点二十分到下午十三点三十分	
	5:00am	凌晨五点整	
	5:30am	凌晨五点半	
	5:20:12am	凌晨五点二十分十二秒	
	7:00am	上午七点整	
	7:30AM	上午七点半	
	7:20:12a.m.	上午七点二十分十二秒	

格式	示例	输出	说明
时间缩写	07:08:12A.M.	上午七点零八分十二秒	
	5:00pm	下午五点整	
	5:30PM	下午五点半	
	5:20:12p.m.	下午五点二十分十二秒	
	05:09:12P.M.	下午五点零九分十二秒	
	9:00pm	晚上九点整	
	9:30pm	晚上九点半	
	9:20:12PM	晚上九点二十分十二秒	
	9:02:12P.M.	晚上九点零二分十二秒	
	12:00pm	中午十二点整	
	12:30p.m.	中午十二点半	
	12:20:12PM	中午十二点二十分十二秒	

- currency

格式	示例	输出	说明
	12.00RMB	十二人民币	
	12.50RMB	十二点五零人民币	

格式	示例	输出	说明
数字+金额标识符	12,000,000RMB	一千二百万人民币	支持AUD(澳元)、CAD（加元）、HKD（港币）、JPY（日元）、USD（美元）、CHF（瑞士法郎）、NOK（挪威克朗）、SEK（瑞典克朗）、GBP（英镑）、RMB（人民币）、CNY（元）和EUR（欧元）。 支持的数字格式包括：整数、小数以及以逗号分隔的国际写法。
	12,000,000.00RMB	一千二百万人民币	
	12,000.35RMB	一万两千点三五人民币	
金额标识符+数字	$12	十二美元	支持 CAD（加元）、$（美元）、$（美元）、Fr（法郎）、kr（丹麦克朗）、£（英镑）、¥（元）¥（元）和 €（欧元）。 支持的数字格式包括：整数、小数以及以逗号分隔的国际写法。
	$12.00	十二美元	
	$12.12	二点一二美元	
	$12,000	一万两千美元	
	$12,000.00	一万两千美元	
	$12,000.99	一万两千点九九美元	
其他默认读法	1213	一千二百一十三	无
	1213KML	一千二百一十三K M L	
	1213.00KML	一千二百一十三K M L	
	1213.9KML	一千二百一十三点九K M L	
	1,000KML	一千K M L	
	1,000.00KML	一千K M L	

格式	示例	输出	说明
	1,000.98KML	一千点九八KML	
	12,000	一万两千	

- measure

格式	示例	输出	说明
数字+中文单位	2片	两片	
	120公顷	一百二十公顷	
	100多毫克	一百多毫克	
	100来米	一百来米	
	100余人	一百余人	
	1厘米20毫米	一厘米二十毫米	
	120.00平方公里	一百二十平方公里	
数字+单位缩写	120.56cm²	一百二十点五六平方厘米	
	120㎡56cm²	一百二十平方米五十六平方厘米	
	100m12cm6mm	一百米十二厘米六毫米	
	10~15kg	十至十五千克	

格式	示例	输出	支持常见中文单位及单位缩写。
范围	10.24~789.82亩	十点二四至七百八十九点八二亩	
	10米~15米	十米至十五米	
	10.24cm~19.08cm	十点二四厘米至十九点零八厘米	
数字+单位+"/"+单位	10元/斤	十元每斤	
	199~299元/件	一百九十九至二百九十九元每件	
	299.99元/g~399.99元/g	二百九十九点九九元每克至三百九十九点九九元每克	
其他默认读法	12扎	十二扎	
	30rm	三十r m	
	4万万同胞	四万万同胞	
	12.897微克	十二点八九七微克	

其中<say-as>常见符号读法如下表所示。

符号	读法
!	叹号
“	双引号
#	井号

符号	读法
$	dollar
%	百分号
&	and
‘	单引号
(	左括号
)	右括号
*	星
+	加
,	逗号
-	杠
.	点
/	斜杠
:	零冒号
;	分号
<	小于
=	等号
>	大于

符号	读法
?	问号
@	at
[	左方括号
\	反斜线
]	右方括号
^	脱字符
_	下划线
`	反引号
{	左花括号
|	竖线
}	右花括号
~	波浪线
!	叹号
“	左双引号
”	右双引号
‘	左单引号
’	右单引号

符号	读法
（	左括号
）	右括号
，	逗号
。	句号
—	杠
：	冒号
；	分号
？	问号
、	顿号
…	省略号
……	省略号
《	左书名号
》	右书名号
￥	人民币符号
≥	大于等于
≤	小于等于
≠	不等于

符号	读法
≈	约等于
±	加减
×	乘
π	派
A	阿尔法
B	贝塔
Γ	伽玛
Δ	德尔塔
E	艾普西龙
Z	捷塔
E	依塔
Θ	西塔
I	艾欧塔
K	喀帕
Λ	拉姆达
M	缪
N	拗

符号	读法
Ξ	克西
Ο	欧麦克轮
Π	派
Ρ	柔
Σ	西格玛
Τ	套
Υ	宇普西龙
Φ	fai
Χ	器
Ψ	普赛
Ω	欧米伽
α	阿尔法
β	贝塔
γ	伽玛
δ	德尔塔
ε	艾普西龙
ζ	捷塔

符号	读法
η	依塔
θ	西塔
ι	艾欧塔
κ	喀帕
λ	拉姆达
μ	缪
ν	拗
ξ	克西
ο	欧麦克轮
π	派
ρ	柔
σ	西格玛
τ	套
υ	宇普西龙
φ	fai
χ	器
ψ	普赛

符号	读法
ω	欧米伽

<say-as>常见计量单位如下表所示。

格式	类别	示例
缩写	长度	nm（纳米）、μm（微米）、 mm（毫米）、cm（厘米）、m（米）、km（千米）、ft（英尺）、in（英寸）
	面积	cm^2（平方厘米）、m^2（平方米）、km2（平方千米）、SqFt（平方英尺）
	体积	cm^3（立方厘米）、m^3（立方米）、km3（立方千米）、mL（毫升）、L（升）、gallon（加仑）
	重量	μg（微克）、mg（毫克）、g（克）、kg（千克）
	时间	min（分）、sec（秒）、ms（毫秒）
	电磁	μA（微安）、mA（毫安）、Ω（欧姆）、Hz（赫兹）、KHz（千赫兹）、MHz（兆赫兹）、GHz（吉赫兹）、V（伏）、kV（千伏）、kWh（千瓦时）
	声音	dB（分贝）
	气压	Pa（帕）、kPa（千帕）、Mpa（兆帕）
中文单位	支持不限于上述类别的中文单位，例如“米”、“秒”、“美元”、“毫升每瓶”等。以及中文量词，例如“架”、“场”、“头”、“部”、“盆”等。	

- 标签关系

  <say-as>标签仅包括文本。

- 示例

  - cardinal

```
<speak>
 <say-as interpret-as="cardinal">12345</say-as>
</speak>
```

  音频效果：SSML-say-as_Cardinal.mp3

- digits

```
<speak>
 <say-as interpret-as="digits">12345</say-as>
</speak>
```

音频效果：SSML-say-as_digit.mp3

- telephone

```
<speak>
 <say-as interpret-as="telephone">12345</say-as>
</speak>
```

音频效果：SSML-say-as_Telephone.mp3

- name

```
<speak>
 她的曾用名是<say-as interpret-as="name">曾小凡</say-as>
</speak>
```

音频效果：SSML-say-as_Name.mp3

- address

```
<speak>
 <say-as interpret-as="address">富路国际1号楼3单元304</say-as>
</speak>
```

音频效果：SSML-say-as_Address.mp3

- id

```
<speak>
 <say-as interpret-as="id">myid_1998</say-as>
</speak>
```

音频效果：SSML-say-as_id.mp3

- characters

```
<speak>
 <say-as interpret-as="characters">希腊字母αβ</say-as>
</speak>
```

音频效果：SSML-say-as_characters.mp3

- punctuation

```
<speak>
 <say-as interpret-as="punctuation"> -./:;</say-as>
</speak>
```

音频效果：SSML-say-as_punctuation.mp3

- date

```
<speak>
 <say-as interpret-as="date">1000-10-10</say-as>
</speak>
```

音频效果：SSML-say-as_date.mp3

- time

```
<speak>
 <say-as interpret-as="time">5:00am</say-as>
</speak>
```

音频效果：SSML-say-as_time.mp3

- currency

```
<speak>
 <say-as interpret-as="currency">13,000,000.00RMB</say-as>
</speak>
```

音频效果：SSML-say-as_currency.mp3

- measure

```
<speak>
 <say-as interpret-as="measure">100m12cm6mm</say-as>
</speak>
```

音频效果：SSML-say-as_measure.mp3

综合示例

本示例将详细展示SSML的使用方法，您可以直接拷贝到管控台的项目功能配置中，测试效果。音频效果：综合示例.mp3。

> 说明
>
> - 示例样音中旁白发音人使用的是定制声音，暂没有开放。体验定制声音，请参见语音合成声音定制。
> - 语音合成每次合成请求的文本字数不能超过300字符，且只能使用一次<speak>标签，以下示例需要按照<speak>标签分多次进行合成测试。

```
<speak>
 相传北宋年间，
 <say-as interpret-as="date">1121-10-10</say-as>，
 <say-as interpret-as="address">开封城</say-as>
 郊外的早晨笼罩在一片
 <sub alias="双十一">11.11</sub>
 前买买买的欢乐海洋中。一支运货的骡队刚进入城门
 <soundEvent src="http://nls.alicdn.com/sound-event/bell.wav"/>
 一个肤白貌美
 <phoneme alphabet="py" ph="de5">地</phoneme>
 姑娘便拦下第一排的小哥<say-as interpret-as="name">阿发。</say-as>
</speak>

<speak voice="xiaomei">
 “亲，本店今日特惠，鞋履全场
 <say-as interpret-as="digits">199</say-as>
 减
 <say-as interpret-as="cardinal">100</say-as>，
 走过路过不要错过”。
</speak>

<speak voice="sicheng" rate="150">
 “不啦不啦，赶着上货，已经
 <say-as interpret-as="time">09:59:59</say-as>
 了，再晚就供应链断裂了”。
</speak>

<speak>
 <say-as interpret-as="name">阿发</say-as>
 擦了擦汗，带着运货队伍，径直穿过闹巷，耳边充斥着各种叫卖声：
</speak>

<speak voice="ninger" rate="200">
 最新花色现染布匹，买两尺送一尺；
```

```
</speak>

<speak voice="xiaobei">
 爆款纱帽头盔，7天无理由退货；
</speak>

<speak voice="sijia">
 专治大小方脉，调理男人妇人疑难杂症。
</speak>

<speak>
 突然，一匹马不知怎么受了惊，在路上嘶鸣狂奔
 <soundEvent src="http://nls.alicdn.com/sound-event/horse-neigh.wav"/>
 一个孩子也吓坏了，跌跌撞撞地扑向大人怀里
 <break time="50ms"/>大喊道：
</speak>

<speak voice="sitong" rate="150">
  “妈妈，妈妈！”
</speak>

<speak>
 这时，
 <say-as interpret-as="name">阿发</say-as>
 心想
</speak>

<speak effect="robot" pitch="-100">
  “吓死宝宝了！”
</speak>

<speak>
 于是他赶紧捂住了
 <phoneme alphabet="py" ph="he2 bao1">钱包</phoneme>，
 继续赶路送货。一路上，
 <say-as interpret-as="address">开封城</say-as>
 的繁荣景象给
 <say-as interpret-as="name">阿发</say-as>
 留下了深刻的印象。
</speak>
```

```
<speak bgm="http://nls.alicdn.com/bgm/2.wav" backgroundMusicVolume="30" rate="-200">
 物换星移，繁华落尽，于是他在购物狂欢之余握起画笔，勾勒出一幅长卷，并命名为《清明上河图》。
</speak>
```

5.2. FAQ

本文为您介绍使用SSML的相关问题。

如何在长文本接口中使用SSML？

- 如控制下面这段文本每句话之间的停顿，建议先基于标点符号（句号、感叹号、问号等）对文本分句，然后在句子上加SSML标签。

  文本内容如下：

  ```
  大考当前，不少考生容易出现不同程度的焦虑症状，如情绪烦躁、记忆力下降、注意力难以集中，甚至厌食、失眠等
  。
  如何应对考前"心慌慌"？
  新东方教育科技集团董事长俞敏洪特意通过网易教育录制视频，为高考学子送出6点忠告。
  俞敏洪：各位亲爱的、即将参加高考的同学，大家好。一年一度的高考，尽管今年因为疫情原因推迟了一个月，但是
  也马上即将来临。
  我相信凡是参加高考的同学，内心或多或少一定会有些紧张，因为咱们中国的高考几乎都是一考定终身，考的好与不
  好直接跟你未来能上什么样的大学有非常密切的关联。
  在高考快要到来的这个时候，我给同学们几个小小的建议。
  ```

  添加SSML标签后如下：

  ```
  <speak>大考当前，不少考生容易出现不同程度的焦虑症状，如情绪烦躁、记忆力下降、注意力难以集中，甚至厌
  食、失眠等<break time="700ms"/></speak>
  <speak>如何应对考前"心慌慌"？新东方教育科技集团董事长俞敏洪特意通过网易教育录制视频，为高考学子送出6
  点忠告<break time="700ms"/></speak>
  <speak>俞敏洪<break time="400ms"/>各位亲爱的、即将参加高考的同学，大家好<break time="700ms"/><
  /speak>
  <speak>一年一度的高考，尽管今年因为疫情原因推迟了一个月，但是也马上即将来临<break time="700ms"/><
  /speak>
  <speak>我相信凡是参加高考的同学，内心或多或少一定会有些紧张，因为咱们中国的高考几乎都是一考定终身，
  考的好与不好直接跟你未来能上什么样的大学有非常密切的关联<break time="700ms"/></speak>
  <speak>在高考快要到来的这个时候，我给同学们几个小小的建议<break time="700ms"/></speak>
  ```

- 如需要修改一个段落中某个字的读音，建议取多音字所在的语句片段（位于左右两个常用标点符号之间的部分），添加SSML标记。 文本内容如下：

  ```
  按照广州市国规委之前公示的规划，学校用地面积59593平方米，主入口在天坤二路。学校东部设置有5层的小学教
  学综合楼、5层的中学实验楼及教学楼、11层的宿舍楼和6层的行政楼。
  ```

添加SSML标签后如下：

```
按照广州市国规委之前公示的规划，学校用地面积59593平方米，主入口在天坤二路。学校东部设置有5层的小学教学综合楼、5层的中学实验楼及教学楼、
<speak>11层的宿舍楼和6层的<phoneme alphabet="py" ph="xing2">行</phoneme>政楼。</speak>
```