ALIBABA CLOUD

阿里云

智能语音交互

语音合成

文档版本：20201022

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

语音合成为您提供将输入文本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。
- 支持设置不同类型的声音。

> 注意
>
> 字级别音素边界接口：语音合成服务在输出音频的同时，可输出每个字在音频中的时间位置，即时间戳。该时间信息可用于驱动虚拟人口型、做视频配音字幕等。详情请参见语音合成时间戳功能介绍。

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	支持字级别音素边界接口	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	否	无
小刚	Xiaogang	标准男声	通用场景	中文及中英文混合场景	8K/16K	否	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	否	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	是	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	否	无
思诚	Sicheng	标准男声	通用场景	中文及中英文混合场景	8K/16K/24K	是	无

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	支持字级别音素边界接口	备注
艾琪	Aiqi	温柔女声	通用场景	中文及中英文混合场景	8K/16K	是	无
艾佳	Aijia	标准女声	通用场景	中文及中英文混合场景	8K/16K	是	无
艾诚	Aicheng	标准男声	通用场景	中文及中英文混合场景	8K/16K	是	无
艾达	Aida	标准男声	通用场景	中文及中英文混合场景	8K/16K	是	无
宁儿	Ninger	标准女声	通用场景	纯中文场景	8K/16K/24K	否	无
瑞琳	Ruilin	标准女声	通用场景	纯中文场景	8K/16K/24K	否	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	否	无
艾雅	Aiya	严厉女声	客服场景	中文及中英文混合场景	8K/16K	是	无
艾夏	Aixia	亲和女声	客服场景	中文及中英文混合场景	8K/16K	是	无
艾美	Aimei	甜美女声	客服场景	中文及中英文混合场景	8K/16K	是	无

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	支持字级别音素边界接口	备注
艾雨	Aiyu	自然女声	客服场景	中文及中英文混合场景	8K/16K	是	无
艾悦	Aiyue	温柔女声	客服场景	中文及中英文混合场景	8K/16K	是	无
艾婧	Aijing	严厉女声	客服场景	中文及中英文混合场景	8K/16K	是	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	否	无
艾娜	Aina	浙普女声	客服场景	纯中文场景	8K/16K	是	无
伊娜	Yina	浙普女声	客服场景	纯中文场景	8K/16K/24K	否	无
思婧	Sijing	严厉女声	客服场景	纯中文场景	8K/16K/24K	是	无
思彤	Sitong	儿童音	童声场景	纯中文场景	8K/16K/24K	否	无
小北	Xiaobei	萝莉女声	童声场景	纯中文场景	8K/16K/24K	是	无
艾彤	Aitong	儿童音	童声场景	纯中文场景	8K/16K	是	无
艾薇	Aiwei	萝莉女声	童声场景	纯中文场景	8K/16K	是	无

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	支持字级别音素边界接口	备注
艾宝	Aibao	萝莉女声	童声场景	纯中文场景	8K/16K	是	无
Harry	Harry	英音男声	英文场景	英文场景	8K/16K	否	无
Abby	Abby	美音女声	英文场景	英文场景	8K/16K	否	无
Andy	Andy	美音男声	英文场景	英文场景	8K/16K	否	无
Eric	Eric	英音男声	英文场景	英文场景	8K/16K	否	无
Emily	Emily	英音女声	英文场景	英文场景	8K/16K	否	无
Luna	Luna	英音女声	英文场景	英文场景	8K/16K	否	无
Luca	Luca	英音男声	英文场景	英文场景	8K/16K	否	无
Wendy	Wendy	英音女声	英文场景	英文场景	8K/16K/24K	否	无
William	William	英音男声	英文场景	英文场景	8K/16K/24K	否	无
Olivia	Olivia	英音女声	英文场景	英文场景	8K/16K/24K	否	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	否	无
小玥	Xiaoyue	四川话女声	方言场景	中文及中英文混合场景	8K/16K	否	公测版

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	支持字级别音素边界接口	备注
Lydia	Lydia	英中双语女声	英文场景	英文场景	8K/16K	否	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	是	公测版
青青	Qingqing	台湾话女声	方言场景	中文场景	8K/16K	否	公测版
翠姐	Cuijie	东北话女声	方言场景	中文场景	8K/16K	否	公测版
小泽	Xiaoze	湖南重口音男声	方言场景	中文场景	8K/16K	是	公测版

调用限制

- 输入文本必须采用 `UTF-8` 编码。
- 输入文本不能超过300个字符，超过300字符的内容会被截断。

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）。	wss://nls-gateway.cn-shanghai.aliyuncs.com/ws/v1

访问类型	说明	URL
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。关于Token获取请参见获取Token。

2. 开始合成

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

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

参数	类型	是否必需	说明
text	String	是	待合成文本，文本内容必须采用 UTF-8 编码，长度不超过300个字符（英文字母之间需要添加空格）。
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PCM/WAV/MP3格式。
sample_rate	Integer	否	音频采样率，默认值：16000。
volume	Integer	否	音量，取值范围：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，举例如下。

```
{
  "header": {
    "message_id": "05450bf69c53413f8d88aed1ee60****",
    "task_id": "640bc797bb684bd6960185651307****",
    "namespace": "SpeechSynthesizer",
    "name": "SynthesisCompleted",
    "status": 20000000,
    "status_message": "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

服务的每一次响应都包含status字段，即服务状态码，各状态码含义如下。

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	首先检查使用的令牌是否过期，然后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）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。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	使用了不支持的接口，如果使用SDK请提交工单。
40010002	不支持的指令	使用了不支持的指令，如果使用SDK请提交工单。
40010003	无效的指令	指令格式错误，如果使用SDK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关闭了连接。
40010005	任务状态错误	发送了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。

下载安装

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

```
<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-tts-2.0.0-jar-with-dependencies.jar，将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-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-tts-2.0.0-jar-with-dependencies.jar
```

> 说明
>
> 自行压测超过2并发将产生费用。

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，通过该接口设置请求参数，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
 *
 * @param response
 */
abstract public void onComplete(SpeechSynthesizerResponse response);
```

更多介绍，请参见Java API接口说明。

> 注意
>
> 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", accesToken);
>   ```
>
> - 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。

```
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:
 *     语音合成API调用。
 *     动态获取token。
 *     流式合成TTS。
 *     首包延迟计算。
 */
public class SpeechSynthesizerDemo {
    private static final Logger logger = LoggerFactory.getLogger(SpeechSynthesizerDemo.class);
    private static long startTime;
    private String appKey;
    NlsClient client;
    public SpeechSynthesizerDemo(String appKey, String accessKeyId, String accessKeySecret) {
        this.appKey = appKey;
        //应用全局创建一个NlsClient实例，默认服务地址为阿里云线上服务地址。
        //获取token，使用时注意在accessToken.getExpireTime()过期前再次获取。
        AccessToken accessToken = new AccessToken(accessKeyId, accessKeySecret);
        try {
            accessToken.apply();
            System.out.println("get token: " + accessToken.getToken() + ", expire time: " + accessToken.get
ExpireTime());
            client = new NlsClient(accessToken.getToken());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public SpeechSynthesizerDemo(String appKey, String accessKeyId, String accessKeySecret, String ur
l) {
        this.appKey = appKey;
        AccessToken accessToken = new AccessToken(accessKeyId, accessKeySecret);
        try {
            accessToken.apply();
            System.out.println("get token: " + accessToken.getToken() + ", expire time: " + accessToken.get
```

```
ExpireTime());
        if(url.isEmpty()) {
            client = new NlsClient(accessToken.getToken());
        }else {
            client = new NlsClient(url, accessToken.getToken());
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
  private static SpeechSynthesizerListener getSynthesizerListener() {
    SpeechSynthesizerListener listener = null;
    try {
        listener = new SpeechSynthesizerListener() {
            File f=new File("tts_test.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) {
                        //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 - SpeechSynthesizerDemo.startTime) + " ms");
                    }
                    byte[] bytesArray = new byte[message.remaining()];
                    message.get(bytesArray, 0, 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task_id以便排查。
            System.out.println(
                "task_id: " + response.getTaskId() +
                    //状态码 20000000 表示识别成功
                    ", status: " + response.getStatus() +
                    //错误信息
                    ", status_text: " + response.getStatusText());
        }
    };
  } catch (Exception e) {
    e.printStackTrace();
  }
  return listener;
}
public void process() {
  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(100);
    //语速，范围是-500~500，默认是0。
    synthesizer.setSpeechRate(100);
    //设置用于语音合成的文本
    synthesizer.setText("欢迎使用阿里巴巴智能语音合成服务，您可以说北京明天天气怎么样啊");
    // 是否开启字幕功能（返回相应文本的时间戳），默认不开启，需要注意并非所有发音人都支持该参数。
    synthesizer.addCustomedParam("enable_subtitle", false);
```

```
            //此方法将以上参数设置序列化为JSON格式发送给服务端，并等待服务端确认。
            long start = System.currentTimeMillis();
            synthesizer.start();
            logger.info("tts start latency " + (System.currentTimeMillis() - start) + " ms");
            SpeechSynthesizerDemo.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 = "您的appkey";
        String id = "您的AccessKey Id";
        String secret = "您的AccessKey Secret";
        String url = ""; //默认值：wss://nls-gateway.cn-shanghai.aliyuncs.com/ws/v1
        if (args.length == 3) {
            appKey  = args[0];
            id      = args[1];
            secret  = args[2];
        } else if (args.length == 4) {
            appKey  = args[0];
            id      = args[1];
            secret  = args[2];
            url     = args[3];
        } else {
            System.err.println("run error, need params(url is optional): " + "<app-key> <AccessKeyId> <Acc
essKeySecret> [url]");
            System.exit(-1);
        }
        SpeechSynthesizerDemo demo = new SpeechSynthesizerDemo(appKey, id, secret, url);
        demo.process();
```

```
        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。
> - 使用SDK之前，请先阅读接口说明，详情请参见接口说明。
> - 该版本C++ SDK API和上一版本API（已下线）定义有区别，本文以当前版本为例进行介绍。

下载安装

SDK下载：

下载CppSdk，文件包含以下几个部分：

- 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	一句话识别
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：
   - Cmake 3.1
   - Glibc 2.5
   - 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）。
./stDemo <yourAppkey> <yourAccessKeyId> <yourAccessKeySecret> #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

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，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，并通过工单系统反馈给阿里云。
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阿里云。
10000009	HTTP: connect failed.	与云端连接失败，请检查网络后，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URL，则在创建SpeechRecognizerRequest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包中demo目录speechSynthesizerDemo.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token和其有效期时间戳。
//说明：只需在token即将过期时进行重新生成。所有的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 在接收到云端返回合成结束消息时，SDK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，方便定位问题，作为和服务端交互的唯一标识建议输出。
    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 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。
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事件上报给用户。
//@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 ) {
        // 重要提示：一旦获取到语音流，如第一次从服务端返回合成语音流，即可以进行播放或者其他处理。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() ： 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();
    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：PCM/WAV/MP3。
    request->setSampleRate(8000);       // 音频采样率，支持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，可选。注意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); // 调用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 speechSynthesizerFile(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;
        }
    }
    ParamStruct pa;
    pa.token = g_token;
    pa.appkey = appkey;
    // 说明：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。
// 免费用户并发连接不能超过2个。
#define AUDIO_TEXT_NUMS 2
#define AUDIO_TEXT_LENGTH 64
#define AUDIO_FILE_NAME_LENGTH 32
int speechSynthesizerMultFile(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 S
```

```
ecret>\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-synthesizer", LogDebug);
    if (-1 == ret) {
        printf("set log failed\n");
        return -1;
    }
    //启动工作线程
    NlsClient::getInstance()->startWorkThread(4);
    //合成单个文本
    speechSynthesizerFile(appkey.c_str());
    //合成多个文本
    //speechSynthesizerMultFile(appkey.c_str());
    //所有工作完成，进程退出前，释放nlsClient。releaseInstance()非线程安全。
    NlsClient::releaseInstance();
    return 0;
}
```

4.Python SDK

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

前提条件

使用SDK前，请先阅读接口说明，详情请参见接口说明。

下载安装

> 说明
>
> - SDK仅支持Python3.4及以上版本，暂不支持Python 2。
> - 确认已安装Python包管理工具setuptools，如果没有安装，请在Linux终端使用以下命令安装：
>
> ```
> pip install setuptools
> ```

1. 下载Python SDK。
2. 安装SDK。

   在Linux终端的SDK目录下，执行以下命令：

   ```
   # 打包
   python setup.py bdist_egg
   # 安装
   python setup.py install
   ```

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，通过该接口设置请求参数，发送请求。非线程安全。
  - start方法：建立与服务端的连接。默认参数ping_interval表示自动发送ping命令的间隔，ping_timeout表示等待ping命令响应的pong消息的超时时间，需要满足ping_interval 大于ping_timeout。
  - wait_completed方法：等待服务端合成完毕或者合成超时。
  - close方法：关闭与服务端的网络链接。
- SpeechSynthesizerCallback：回调事件函数集合的类，合成结果、异常等回调的统一入口。
  - on_binary_data_received方法：客户端接收到合成音频数据的回调。
  - on_completed方法：客户端接收到合成结束的回调。
  - on_completed方法：客户端接收到异常错误的回调。
  - on_channel_closed方法：客户端接收到断开网络链接成功的回调。

SDK调用注意事项

- NlsClient对象创建一次可以重复使用。
- 一个合成任务对应一个SpeechSynthesizer对象。例如有N个文本需要合成，则要进行N次语音合成任务，创建N个SpeechSynthesizer对象。
- SpeechSynthesizerCallback对象和SpeechSynthesizer对象是一一对应的，不能将一个实现的SpeechSynthesizerCallback对象设置到多个SpeechSynthesizer对象中，否则无法将识别任务区分开。

示例代码

> 说明
>
> - 示例使用SDK内置的默认语音合成服务的外网访问服务端URL，如果您使用上海阿里云ECS并想使用内网访问服务端URL，则在创建NlsClient对象时，设置内网访问的URL：
>
>   ```
>   synthesizer=client.create_synthesizer(callback,"ws://nls-gateway.cn-shanghai-internal.aliyuncs.com/ws/v1")ynthesizer=client.create_synthesizer(callback,"ws://nls-gateway.cn-shanghai-internal.aliyuncs.com/ws/v1")
>   ```
>
> - 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。

```
# -*- coding: utf-8 -*-
import threading
import ali_speech
from ali_speech.callbacks import SpeechSynthesizerCallback
from ali_speech.constant import TTSFormat
from ali_speech.constant import TTSSampleRate
class MyCallback(SpeechSynthesizerCallback):
    # 参数name用于指定保存音频的文件。
    def __init__(self, name):
        self._name = name
        self._fout = open(name, 'wb')
    def on_binary_data_received(self, raw):
        print('MyCallback.on_binary_data_received: %s' % len(raw))
        self._fout.write(raw)
    def on_completed(self, message):
        print('MyCallback.OnRecognitionCompleted: %s' % message)
        self._fout.close()
    def on_task_failed(self, message):
        print('MyCallback.OnRecognitionTaskFailed-task_id:%s, status_text:%s' % (
            message['header']['task_id'], message['header']['status_text']))
        self._fout.close()
    def on_channel_closed(self):
        print('MyCallback.OnRecognitionChannelClosed')
```

```
        print(f"MyCallback.OnSynthesisStarted: {message}")
def process(client, appkey, token, text, audio_name):
    callback = MyCallback(audio_name)
    synthesizer = client.create_synthesizer(callback)
    synthesizer.set_appkey(appkey)
    synthesizer.set_token(token)
    synthesizer.set_voice('xiaoyun')
    synthesizer.set_text(text)
    synthesizer.set_format(TTSFormat.WAV)
    synthesizer.set_sample_rate(TTSSampleRate.SAMPLE_RATE_16K)
    synthesizer.set_volume(50)
    synthesizer.set_speech_rate(0)
    synthesizer.set_pitch_rate(0)
    try:
        ret = synthesizer.start()
        if ret < 0:
            return ret
        synthesizer.wait_completed()
    except Exception as e:
        print(e)
    finally:
        synthesizer.close()
def process_multithread(client, appkey, token, number):
    thread_list = []
    for i in range(0, number):
        text = "这是线程" + str(i) + "的合成。"
        audio_name = "sy_audio_" + str(i) + ".wav"
        thread = threading.Thread(target=process, args=(client, appkey, token, text, audio_name))
        thread_list.append(thread)
        thread.start()
    for thread in thread_list:
        thread.join()
if __name__ == "__main__":
    client = ali_speech.NlsClient()
    # 设置输出日志信息的级别：DEBUG、INFO、WARNING、ERROR。
    client.set_log_level('INFO')
    appkey = '您的appkey'
    token = '您的token'
    text = "今天是周一，天气挺好的。"
    audio_name = 'sy_audio.wav'
    process(client, appkey, token, text, audio_name)
```

```
    # 多线程示例
    # process_multithread(client, appkey, token, 2)
```

5.Android SDK

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

> 说明
>
> 推荐您使用新版本Android SDK，本版本后续将不再更新。详情请参见Android SDK。

前提条件

- 阅读接口说明，详情请参见接口说明。
- 已获取项目appkey，详情请参见创建项目。
- 已获取智能语音服务访问令牌，详情请参见获取Token。

下载安装

1. 下载SDK和示例代码。
2. 解压ZIP文件，在app/libs目录下获取AAR格式的SDK包。
3. 使用Android Studio打开此工程。

   语音合成示例代码为SpeechSynthesizerActivity.java文件。

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：代表一次语音合成请求。
- SpeechSynthesizerCallback：语音合成回调接口，在获得合成音频数据、发生错误等事件发生时会触发回调。您需要实现此接口，在回调方法中加入自己的处理逻辑。

调用顺序

1. 创建NlsClient实例。
2. 定义SpeechSynthesizerCallback实现类，按业务需求处理识别结果或错误情况。
3. 调用NlsClient.createSynthesizerRequest()方法得到SpeechSynthesizer实例。
4. 设置SpeechSynthesizer参数。

   包括access Token、appkey、语音文本（text）、发音人（voice）和语速（speechRate）等。
5. 调用SpeechSynthesizer.start()方法开始与云端服务连接。
6. 在回调中获得合成的音频数据并播放，或者处理错误。
7. 调用SpeechSynthesizer.stop()方法结束语音合成。

   > 说明
   >
   > 如果需要发起新的请求，请重复步骤3~7。
8. 调用NlsClient.release()方法释放客户端实例。

Proguard配置

如果代码使用了混淆，请在proguard-rules.pro中配置：

```
-keep class com.alibaba.idst.util.*{*;}
```

代码示例

- 创建识别请求

```
// 创建语音合成对象
speechSynthesizer = client.createSynthesizerRequest(callback);
speechSynthesizer.setToken("");
speechSynthesizer.setAppkey("");
// 设置语音编码，PCM编码可以直接用audioTrack播放，其他编码不行。
speechSynthesizer.setFormat(SpeechSynthesizer.FORMAT_PCM);
// 以下选项都会改变最终合成的语音效果。
// 设置语音数据采样率
speechSynthesizer.setSampleRate(SpeechSynthesizer.SAMPLE_RATE_16K);
// 设置人声
speechSynthesizer.setVoice(SpeechSynthesizer.VOICE_XIAOGANG);
// 设置语音合成方法
speechSynthesizer.setMethod(SpeechSynthesizer.METHOD_RUS);
// 设置语速
speechSynthesizer.setSpeechRate(100);
// 设置是否返回语音对应的时间戳信息
speechSynthesizer.setEnableSubtitle(true);
// 设置要转为语音的文本
speechSynthesizer.setText("欢迎使用智能语音！");
speechSynthesizer.start()
```

- 获取合成语音并播放

```
// 获取音频数据的回调，在这里将音频写入播放器。
@Override
public void OnBinaryReceived(byte[] data, int code)
{
   Log.d(TAG, "binary received length: " + data.length);
   if (!playing) {
      playing = true;
      audioTrack.play();
   }
   audioTrack.write(data, 0, data.length);
}
```

- 返回语音时间戳信息

```
// 调用onMetaInfo，需要设置：SpeechSynthesizer.setEnableSubtitle(true)。
@Override
public void onMetaInfo(String message, int code) {
   Log.d(TAG,"onMetaInfo " + message + ": " + String.valueOf(code));
}
```

6.iOS SDK

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

> 说明
>
> 推荐您使用新版本iOS SDK，本版本后续将不再更新。详情请参见iOS SDK。

前提条件

- 首先阅读接口说明，详情请参见接口说明。
- 已在智能语音管控台创建项目并获取appkey，详情请参见创建项目。
- 已获取智能语音服务访问令牌，详情请参见获取Token。

语音合成

语音合成即将文本转化为语音。我们支持多个说话人声音，支持PCM/WAV/MP3格式输出，示例实现了基于PCM的语音合成和播放。

下载安装

1. 下载iOS SDK。

   解压后，NlsDemo目录即为示例工程目录。

2. 双击NlsDemo.xcodeproj，使用Xcode打开示例工程。
   - 导入的iOS SDK即NlsDemo/AliyunNlsSdk.framework。SDK支持 x86_64/armv7/arm64架构。
   - 如果您要将应用程序提交发布至苹果应用商店，请您使用该版本SDK：NlsDemo-iOS/dynamic-verison-framework/AliyunNlsSdk.framework。

调用步骤

> 说明
>
> 请使用Embedded Binaries方式导入SDK到工程中。

1. 导入NlsSdk中的AliyunNlsClientAdaptor.h、NlsSpeechSynthesizerRequest.h以及SynthesizerRequestParam.h头文件。
2. 实现NlsSpeechSynthesizerRequest的NlsSpeechSynthesizerDelegate回调方法。
3. 创建AliyunNlsClientAdaptor对象nlsClient。

   该对象全局只需创建一次且可以重复使用。

4. 通过调用nlsClient对象的createSynthesizerRequest方法获得NlsSpeechSynthesizerRequest对象。

   该NlsSpeechSynthesizerRequest对象不可重复使用，一个请求创建一个对象。

5. 通过SynthesizerRequestParam设置参数，如access Token、appkey等。
6. 通过NlsSpeechSynthesizerRequest的setSynthesizerParams 传入步骤5中设置的SynthesizerRequestParam对象。

7. 调用NlsSpeechSynthesizerRequest对象的start方法，开始语音合成。
8. 通过NlsSpeechSynthesizerDelegate回调的 `-(void)OnBinaryDataReceived:(NlsDelegateEvent)event voiceData:(Byte *)data length:(NSInteger)length;` 方法传入合成语音数据。
9. 使用NLSPlayAudio工具播放步骤8接收的语音数据。

关键接口

- AliyunNlsClientAdaptor：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。
- NlsSpeechSynthesizerRequest：语音合成处理的请求对象，线程安全。
- SynthesizerRequestParam：语音合成相关参数。
- NlsSpeechSynthesizerDelegate：语音合成相关回调函数。在获得结果、遇到错误等事件发生时会触发回调。

代码示例

```
#import <Foundation/Foundation.h>
#import "Synthesizer.h"
@interface Synthesizer()<NlsSpeechSynthesizerDelegate>{
    IBOutlet UITextView *textViewSynthesizer;
}
@end
@implementation Synthesizer
-(void)viewDidLoad{
    [super viewDidLoad];
    //1. 全局参数初始化操作
    //1.1 初始化语音合成客户端
    _nlsClient = [[NlsClientAdaptor alloc]init];
    //1.2 初始化语音播放工具类
    _nlsAudioPlayer = [[NLSPlayAudio alloc]init];
    //1.3 初始化合成参数类
    _requestParam = [[SynthesizerRequestParam alloc]init];
    //1.4 设置log级别
    [_nlsClient setLog:NULL logLevel:1];
}
-(IBAction)startSynthesizer{
    //2. 创建请求对象和开始语音合成
    if(_synthesizerRequest!= NULL){
        _synthesizerRequest = NULL;
    }
    //2.1 初始化语音播放类
```

```
    [_nlsAudioPlayer cleanup];
    _nlsAudioPlayer = [[NLSPlayAudio alloc]init];
    //2.2 创建请求对象，设置NlsSpeechSynthesizerRequest回调。
    _synthesizerRequest = [_nlsClient createSynthesizerRequest];
    _synthesizerRequest.delegate = self;
    //2.3 获取页面合成文本
    NSString *inputText = [textViewSynthesizer text];
    //2.4 设置SynthesizerRequestParam请求参数
    [_requestParam setFormat:@"pcm"];
    [_requestParam setText:inputText];
    [_requestParam setToken:@""];
    [_requestParam setAppkey:@""];
    //2.5 传入请求参数
    [_synthesizerRequest setSynthesizerParams:_requestParam];
    //2.6 开始语音合成
    [_synthesizerRequest start];
}
/**
 *3. NlsSpeechSynthesizerDelegate接口回调
 */
//3.1 本次请求失败
- (void)OnTaskFailed:(NlsDelegateEvent)event statusCode:(NSString *)statusCode errorMessage:(NS
String *)eMsg {
    NSLog(@"OnTaskFailed, statusCode is: %@ error message： %@",statusCode,eMsg);
}
//3.2 服务端连接关闭
- (void)OnChannelClosed:(NlsDelegateEvent)event statusCode:(NSString *)statusCode errorMessage:
(NSString *)eMsg {
    NSLog(@"OnChannelClosed, statusCode is: %@",statusCode);
}
//3.3 回调合成语音数据，通过NlsAudioPlayer工具播放。
- (void)OnBinaryDataReceived:(NlsDelegateEvent)event voiceData:(Byte *)data length:(NSInteger)len
gth{
    NSLog(@"Received voice data length %lu", length);
    [_nlsAudioPlayer process:data length:length];
}
//3.4 合成结束
- (void)OnSynthesizerCompleted:(NlsDelegateEvent)event result:(NSString *)result statusCode:(NSSt
ring *)statusCode errorMessage:(NSString *)eMsg {
}
//3.5 合成开始
```

```
- (void)OnSynthesizerStarted:(NlsDelegateEvent)event result:(NSString *)result statusCode:(NSStrin
g *)statusCode errorMessage:(NSString *)eMsg {
}
@end
```

7.RESTful API

语音合成RESTful API支持HTTPS GET和POST两种方法的请求，将待合成的文本上传到服务端，服务端返回文本的语音合成结果，开发者需要保证在语音合成结果返回之前连接不中断。

功能介绍

支持如下设置：

- PCM/WAV/MP3音频格式
- 8000Hz/16000Hz采样率
- 多种发音人
- 语速/语调/音量

> 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。本文档及SDK示例中有相关流式处理示例代码可供参考。
> - 单次调用传入文本不能超过300个字符，超过的字符会被截断。对于更长文本的合成，请参考SDK中的长文本切分及拼接示例。
> - 下载nls-restful-java-demo.zip。
>
>   新增RESTful语音合成Java示例代码，增加关于RESTful、HTTP Chunked、长文本切分分段合成等演示程序。

前提条件

- 已准备项目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。	nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts	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。	nls-gateway.cn-shanghai-internal.aliyuncs.com/stream/v1/tts	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。如果您使用的是阿里云上海ECS，并需要使用内网访问URL，则需要使用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 GET方法或POST方法的请求，服务端返回携带合成语音数据的HTTP响应。

> **说明**
>
> 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

语音合成的请求参数如下表所示。

- 如果使用HTTPS GET方法的请求，需要在HTTPS的URL请求参数中设置这些参数。
- 如果使用HTTPS POST方法的请求，需要在HTTPS的请求体（Body）中设置这些参数。

名称	类型	是否必选	描述
appkey	String	是	项目appkey。
text	String	是	待合成的文本，需要为 `UTF-8` 编码。使用GET方法，需要再采用RFC 3986规范进行urlencode编码；使用POST方法不需要urlencode编码。
token	String	否	若不设置token参数，需要在HTTP Headers中设置 `X-NLS-Token` 字段来指定Token。
format	String	否	音频编码格式，支持PCM/WAV/MP3格式。默认值：PCM。

名称	类型	是否必选	描述
sample_rate	Integer	否	音频采样率，支持16000Hz/8000Hz，默认值：16000Hz。
voice	String	否	发音人，默认值：xiaoyun。更多发音人请参见接口说明。
volume	Integer	否	音量，取值范围：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。

GET方法上传文本

一个完整的语音合成RESTful API GET方法的请求包含以下要素：

- URL

协议	URL	方法
HTTPS	nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts	GET

- 请求参数

  参见上述请求参数。由URL和请求参数组成的完整请求链接如下所示，在浏览器中打开该链接可直接获取语音合成结果：

```
# text文本："今天是周一，天气挺好的。"
https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts?appkey=${您的appkey}&token=${您的token}&text=%E4%BB%8A%E5%A4%A9%E6%98%AF%E5%91%A8%E4%B8%80%EF%BC%8C%E5%A4%A9%E6%B0%94%E6%8C%BA%E5%A5%BD%E7%9A%84%E3%80%82&format=wav&sample_rate=16000
```

- HTTPS GET请求头部

名称	类型	是否必选	描述
X-NLS-Token	String	否	服务鉴权Token。

> 注意
>
> - 服务鉴权Token参数有如下两种设置方式：
>   - （推荐）在请求参数 `token` 中设置
>   - 在HTTPS Headers的 `X-NLS-Token` 字段设置
> - 参数 `text` 必须采用 `UTF-8` 编码，再采用RFC 3986规范进行urlencode编码。如加号 `+` 编码为 `%2B` ，星号 `*` 编码为 `%2A` ， `%7E` 编码为 `~` 。

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	nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts	POST

- **HTTPS POST请求头部**

名称	类型	是否必选	描述
Content-Type	String	是	必须为“application/json”。表明HTTP请求体的内容为JSON格式字符串。
X-NLS-Token	String	否	服务鉴权Token。
Content-Length	long	否	HTTP请求体中内容的长度。

- **HTTPS POST请求体**

HTTPS POST请求体由请求参数组成JSON格式的字符串组成，因此在HTTPS POST请求头部中的Content-Type必须设置为"application/json"。示例如下：

```
{
 "appkey":"31f932fb",
 "text":"今天是周一，天气挺好的。",
 "token":"450343c793aaaaaa****",
 "format":"wav"
}
```

> 注意
>
> - 服务鉴权Token参数有如下两种设置方式：
>   - 推荐在请求体通过参数 `token` 设置
>   - 在HTTPS Headers的 `X-NLS-Token` 字段设置
> - 使用POST方法的请求，请求体中的请求参数 `text` 必须采用 `UTF-8` 编码，但是不进行urlencode编码，注意与GET方法请求的区别。

响应结果

使用HTTPS GET方法和HTTPS POST方法请求的响应是相同的，响应结果都包含在HTTPS的响应体中。响应结果的成功/失败通过HTTPS Headers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，合成的语音数据在响应体中。
  - HTTPS Header的 `X-NLS-RequestId` 字段内容为请求任务的task_id。
  - 响应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，错误信息在响应体中。
  - HTTPS Headers的 `X-NLS-RequestId` 字段内容为请求任务的task_id。
  - 响应体内容为错误信息，以JSON格式的字符串表示。如下所示：

```
{
  "task_id":"8f95d0b9b6e948bc98e8d0ce64b0****",
  "result":"",
  "status":40000000,
  "message":"Gateway:CLIENT_ERROR:in post data, json format illegal"
}
```

- 错误信息字段如下表

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。
result	String	服务结果
status	Integer	服务状态码
message	String	服务状态描述

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>
<!-- http://mvnrepository.com/artifact/com.alibaba/fastjson -->
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.42</version>
</dependency>
<dependency>
    <groupId>org.asynchttpclient</groupId>
    <artifactId>async-http-client</artifactId>
    <version>2.5.4</version>
</dependency>
```

示例代码如下：

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import com.alibaba.fastjson.JSONObject;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;
public class SpeechSynthesizerRestfulDemo {
    private String accessToken;
    private String appkey;
    public SpeechSynthesizerRestfulDemo(String appkey, String token) {
        this.appkey = appkey;
        this.accessToken = token;
    }
    /**
     * HTTPS GET请求
```

```
    */
    public void processGETRequet(String text, String audioSaveFile, String format, int sampleRate, Strin
g voice) {
        /**
         * 设置HTTPS GET请求：
         * 1.使用HTTPS协议
         * 2.语音识别服务域名：nls-gateway.cn-shanghai.aliyuncs.com
         * 3.语音识别接口请求路径：/stream/v1/tts
         * 4.设置必须请求参数：appkey、token、text、format、sample_rate
         * 5.设置可选请求参数：voice、volume、speech_rate、pitch_rate
         */
        String url = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts";
        url = url + "?appkey=" + appkey;
        url = url + "&token=" + accessToken;
        url = url + "&text=" + text;
        url = url + "&format=" + format;
        url = url + "&voice=" + voice;
        url = url + "&sample_rate=" + String.valueOf(sampleRate);
        // voice 发音人，可选，默认是xiaoyun。
        // url = url + "&voice=" + "xiaoyun";
        // volume 音量，范围是0~100，可选，默认50。
        // url = url + "&volume=" + String.valueOf(50);
        // speech_rate 语速，范围是-500~500，可选，默认是0。
        // url = url + "&speech_rate=" + String.valueOf(0);
        // pitch_rate 语调，范围是-500~500，可选，默认是0。
        // url = url + "&pitch_rate=" + String.valueOf(0);
        System.out.println("URL: " + url);
        /**
         * 发送HTTPS GET请求，处理服务端的响应。
         */
        Request request = new Request.Builder().url(url).get().build();
        try {
            long start = System.currentTimeMillis();
            OkHttpClient client = new OkHttpClient();
            Response response = client.newCall(request).execute();
            System.out.println("total latency :" + (System.currentTimeMillis() - start) + " ms");
            System.out.println(response.headers().toString());
            String contentType = response.header("Content-Type");
            if ("audio/mpeg".equals(contentType)) {
                File f = new File(audioSaveFile);
                FileOutputStream fout = new FileOutputStream(f);
```

```
            fout.write(response.body().bytes());
            fout.close();
            System.out.println("The GET request succeed!");
        }
        else {
            // ContentType 为 null 或者为 "application/json"
            String errorMessage = response.body().string();
            System.out.println("The GET request failed: " + errorMessage);
        }
        response.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
  /**
   * HTTPS POST请求
   */
  public void processPOSTRequest(String text, String audioSaveFile, String format, int sampleRate, Str
ing voice) {
    /**
     * 设置HTTPS POST请求：
     * 1.使用HTTPS协议
     * 2.语音合成服务域名：nls-gateway.cn-shanghai.aliyuncs.com
     * 3.语音合成接口请求路径：/stream/v1/tts
     * 4.设置必须请求参数：appkey、token、text、format、sample_rate
     * 5.设置可选请求参数：voice、volume、speech_rate、pitch_rate
     */
    String url = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts";
    JSONObject taskObject = new JSONObject();
    taskObject.put("appkey", appkey);
    taskObject.put("token", accessToken);
    taskObject.put("text", text);
    taskObject.put("format", format);
    taskObject.put("voice", voice);
    taskObject.put("sample_rate", sampleRate);
    // voice 发音人，可选，默认是xiaoyun。
    // taskObject.put("voice", "xiaoyun");
    // volume 音量，范围是0~100，可选，默认50。
    // taskObject.put("volume", 50);
    // speech_rate 语速，范围是-500~500，可选，默认是0。
    // taskObject.put("speech_rate", 0);
```

```
        // taskObject.put("speech_rate", 0);
        // pitch_rate 语调，范围是-500~500，可选，默认是0。
        // taskObject.put("pitch_rate", 0);
        String bodyContent = taskObject.toJSONString();
        System.out.println("POST Body Content: " + bodyContent);
        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");
            if ("audio/mpeg".equals(contentType)) {
                File f = new File(audioSaveFile);
                FileOutputStream fout = new FileOutputStream(f);
                fout.write(response.body().bytes());
                fout.close();
                System.out.println("The POST request succeed!");
            }
            else {
                // ContentType 为 null 或者为 "application/json"
                String errorMessage = response.body().string();
                System.out.println("The POST request failed: " + errorMessage);
            }
            response.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        if (args.length < 2) {
            System.err.println("SpeechSynthesizerRestfulDemo need params: <token> <app-key>");
            System.exit(-1);
        }
        String token = args[0];
        String appkey = args[1];
        SpeechSynthesizerRestfulDemo demo = new SpeechSynthesizerRestfulDemo(appkey, token);
        String text = "今天是周一，天气挺好的。";
```

```
        // 采用RFC 3986规范进行urlencode编码。
        String textUrlEncode = text;
        try {
            textUrlEncode = URLEncoder.encode(textUrlEncode, "UTF-8")
                .replace("+", "%20")
                .replace("*", "%2A")
                .replace("%7E", "~");
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        System.out.println(textUrlEncode);
        String audioSaveFile = "syAudio.wav";
        String format = "wav";
        int sampleRate = 16000;
        demo.processGETRequet(textUrlEncode, audioSaveFile, format, sampleRate, "siyue");
        //demo.processPOSTRequest(text, audioSaveFile, format, sampleRate, "siyue");
        System.out.println("### Game Over ###");
    }
}
```

Java（流式合成）示例代码如下：

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.concurrent.CountDownLatch;
import io.netty.handler.codec.http.HttpHeaders;
import org.asynchttpclient.AsyncHandler;
import org.asynchttpclient.AsyncHttpClient;
import org.asynchttpclient.AsyncHttpClientConfig;
import org.asynchttpclient.DefaultAsyncHttpClient;
import org.asynchttpclient.DefaultAsyncHttpClientConfig;
import org.asynchttpclient.HttpResponseBodyPart;
import org.asynchttpclient.HttpResponseStatus;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
/**
 * 此示例演示了
 * 1. TTS的RESTFul接口调用。
 * 2. 启用HTTP chunked机制的处理方式（流式返回）。
```

```
 */
public class SpeechSynthesizerRestfulChunkedDemo {
    private static Logger logger = LoggerFactory.getLogger(SpeechSynthesizerRestfulChunkedDemo.cla
ss);
    private String accessToken;
    private String appkey;
    public SpeechSynthesizerRestfulChunkedDemo(String appkey, String token) {
        this.appkey = appkey;
        this.accessToken = token;
    }
    public void processGETRequet(String text, String audioSaveFile, String format, int sampleRate, Strin
g voice, boolean chunked) {
        /**
         * 设置HTTPS GET请求：
         * 1.使用HTTPS协议
         * 2.语音识别服务域名：nls-gateway.cn-shanghai.aliyuncs.com
         * 3.语音识别接口请求路径：/stream/v1/tts
         * 4.设置必须请求参数：appkey、token、text、format、sample_rate
         * 5.设置可选请求参数：voice、volume、speech_rate、pitch_rate
         * 6.设置参数chunk，启用http流式返回
         */
        String url = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts";
        url = url + "?appkey=" + appkey;
        url = url + "&token=" + accessToken;
        url = url + "&text=" + text;
        url = url + "&format=" + format;
        url = url + "&voice=" + voice;
        url = url + "&sample_rate=" + String.valueOf(sampleRate);
        url = url + "&chunk=" + String.valueOf(chunked);
        System.out.println("URL: " + url);
        try {
            AsyncHttpClientConfig config = new DefaultAsyncHttpClientConfig.Builder()
                .setConnectTimeout(3000)
                .setKeepAlive(true)
                .setReadTimeout(10000)
                .setRequestTimeout(50000)
                .setMaxConnections(1000)
                .setMaxConnectionsPerHost(200)
                .setPooledConnectionIdleTimeout(-1)
                .build();
            AsyncHttpClient httpClient = new DefaultAsyncHttpClient(config);
```

```
        CountDownLatch latch = new CountDownLatch(1);
        AsyncHandler<org.asynchttpclient.Response> handler = new AsyncHandler<org.asynchttpclient
.Response>() {
            FileOutputStream outs;
            boolean firstRecvBinary = true;
            long startTime = System.currentTimeMillis();
            int httpCode = 200;
            @Override
            public State onStatusReceived(HttpResponseStatus httpResponseStatus) throws Exception {
              logger.info("onStatusReceived status {}", httpResponseStatus);
              httpCode = httpResponseStatus.getStatusCode();
              if (httpResponseStatus.getStatusCode() != 200) {
                logger.error("request error " +  httpResponseStatus.toString());
              }
              return null;
            }
            @Override
            public State onHeadersReceived(HttpHeaders httpHeaders) throws Exception {
              outs = new FileOutputStream(new File("tts.wav"));
              return null;
            }
            @Override
            public State onBodyPartReceived(HttpResponseBodyPart httpResponseBodyPart) throws Exc
eption {
              // 注意：此处一旦接收到数据流，即可向用户播放或者用于其他处理，以提升响应速度。
              // 注意：请不要在此回调接口中执行耗时操作，可以以异步或者队列形式将二进制TTS语音流推送到另一
线程中。
              logger.info("onBodyPartReceived " + httpResponseBodyPart.getBodyPartBytes().toString()
);
              if(httpCode != 200) {
                System.err.write(httpResponseBodyPart.getBodyPartBytes());
              }
              if (firstRecvBinary) {
                firstRecvBinary = false;
                // 统计第一包数据的接收延迟。实际上接收到第一包数据后就可以进行业务处理了，比如播放或者发送
给调用方。注意：这里的首包延迟也包括了网络建立链接的时间。
                logger.info("tts first latency " + (System.currentTimeMillis() - startTime) + " ms");
              }
              // 此处以将语音流保存到文件为例。
              outs.write(httpResponseBodyPart.getBodyPartBytes());
              return null;
```

```
                return null;
            }
            @Override
            public void onThrowable(Throwable throwable) {
                logger.error("throwable {}", throwable);
                latch.countDown();
            }
            @Override
            public org.asynchttpclient.Response onCompleted() throws Exception {
                logger.info("completed");
                logger.info("tts total latency " + (System.currentTimeMillis() - startTime) + " ms");
                outs.close();
                latch.countDown();
                return null;
            }
        };
        httpClient.prepareGet(url).execute(handler);
        // 等待合成完成
        latch.await();
        httpClient.close();
    }catch (Exception e) {
    }
  }
  public static void main(String[] args) {
    if (args.length < 2) {
        System.err.println("SpeechSynthesizerRestfulDemo need params: <token> <app-key>");
        System.exit(-1);
    }
    String token = args[0];
    String appkey = args[1];
    SpeechSynthesizerRestfulChunkedDemo demo = new SpeechSynthesizerRestfulChunkedDemo(ap
pkey, token);
    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。";
    // 采用RFC 3986规范进行urlencode编码。
    String textUrlEncode = text;
    try {
        textUrlEncode = URLEncoder.encode(textUrlEncode, "UTF-8")
            .replace("+", "%20")
            .replace("*", "%2A")
            .replace("%7E", "~");
```

```
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        System.out.println(textUrlEncode);
        String audioSaveFile = "syAudio.wav";
        String format = "wav";
        int sampleRate = 16000;
        // 最后一个参数为true表示使用http chunked机制。
        demo.processGETRequet(textUrlEncode, audioSaveFile, format, sampleRate, "aixia", true);
        System.out.println("### Game Over ###");
    }
}
```

C++示例

> 说明
>
> - 下载C++示例。
>
> C++示例使用第三方函数库curl处理HTTPS的请求和响应，使用jsoncpp处理POST请求体的JSON字符串。
> - Linux环境下，运行环境最低要求：Glibc 2.5及以上， Gcc4或Ccc5。
> - Windows下需解压lib目录下的windows.zip库编译使用。

示例目录说明如下：

- CMakeLists.txt：示例工程的CMakeList文件。
- demo：示例文件。

文件名	描述
restfulTtsDemo.cpp	语音合成RESTful API示例。

- include

目录名	描述
curl	curl库头文件目录。
json	jsoncpp库头文件目录。

- lib：包含curl、jsoncpp动态库。

根据平台不同，使用如下版本软件加载库文件：

- linux（Glibc：2.5及以上，Gcc4或Gcc5）
- windows（VS2013、VS2015）

- readme.txt：说明文件。
- release.log：更新记录。
- version：版本号。
- build.sh：示例编译脚本。

编译运行操作步骤：

假设示例文件已解压至 `path/to` 路径下，在Linux终端依次执行如下命令编译运行程序。

- 支持Cmake：
  i. 确认本地系统已安装Cmake 2.4及以上版本。
  ii. `cd path/to/sdk/lib` 。
  iii. `tar -zxvpf linux.tar.gz` 。
  iv. `cd path/to/sdk` 。
  v. `./build.sh` 。
  vi. `cd path/to/sdk/demo` 。
  vii. `./restfulTtsDemo <your-token> <your-appkey>` 。
- 不支持Cmake：
  i. `cd path/to/sdk/lib` 。
  ii. `tar -zxvpf linux.tar.gz` 。
  iii. `cd path/to/sdk/demo` 。
  iv. `g++ -o restfulTtsDemo restfulTtsDemo.cpp -I path/to/sdk/include -L path/to/sdk/lib/linux -ljsoncpp -lssl -lcrypto -lcurl -D_GLIBCXX_USE_CXX11_ABI=0` 。
  v. `export LD_LIBRARY_PATH=path/to/sdk/lib/linux/` 。
  vi. `./restfulTtsDemo <your-token> <your-appkey>` 。

示例代码如下：

```
#ifdef _WIN32
#include <Windows.h>
#endif
#include <iostream>
#include <string>
#include <map>
#include <fstream>
#include <sstream>
```

```
#include "curl/curl.h"
#include "json/json.h"
using namespace std;
#ifdef _WIN32
string GBKToUTF8(const string &strGBK) {
    string strOutUTF8 = "";
    WCHAR * str1;
    int n = MultiByteToWideChar(CP_ACP, 0, strGBK.c_str(), -1, NULL, 0);
    str1 = new WCHAR[n];
    MultiByteToWideChar(CP_ACP, 0, strGBK.c_str(), -1, str1, n);
    n = WideCharToMultiByte(CP_UTF8, 0, str1, -1, NULL, 0, NULL, NULL);
    char * str2 = new char[n];
    WideCharToMultiByte(CP_UTF8, 0, str1, -1, str2, n, NULL, NULL);
    strOutUTF8 = str2;
    delete[] str1;
    str1 = NULL;
    delete[] str2;
    str2 = NULL;
    return strOutUTF8;
}
#endif
void stringReplace(string& src, const string& s1, const string& s2) {
    string::size_type pos = 0;
    while ((pos = src.find(s1, pos)) != string::npos) {
        src.replace(pos, s1.length(), s2);
        pos += s2.length();
    }
}
string urlEncode(const string& src) {
    CURL* curl = curl_easy_init();
    char* output = curl_easy_escape(curl, src.c_str(), src.size());
    string result(output);
    curl_free(output);
    curl_easy_cleanup(curl);
    return result;
}
size_t responseHeadersCallback(void* ptr, size_t size, size_t nmemb, void* userdata)
{
    map<string, string> *headers = (map<string, string>*)userdata;
    string line((char*)ptr);
    string::size_type pos = line.find(':');
```

```
    string::size_type pos = line.find(':');
    if (pos != line.npos)
    {
        string name = line.substr(0, pos);
        string value = line.substr(pos + 2);
        size_t p = 0;
        if ((p = value.rfind('\r')) != value.npos) {
            value = value.substr(0, p);
        }
        headers->insert(make_pair(name, value));
    }
    return size * nmemb;
}
size_t responseBodyCallback(void* ptr, size_t size, size_t nmemb, void* userData) {
    size_t len = size * nmemb;
    char* pBuf = (char*)ptr;
    string* bodyContent = (string*)userData;
    (*bodyContent).append(string(pBuf, pBuf + len));
    return len;
}
int processGETRequest(string appKey, string token, string text,
                      string audioSaveFile, string format, int sampleRate) {
    CURL* curl = NULL;
    CURLcode res;
    curl = curl_easy_init();
    if (curl == NULL) {
        return -1;
    }
    string url = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts";
    /**
     * 设置HTTPS URL请求参数
     */
    ostringstream oss;
    oss << url;
    oss << "?appkey=" << appKey;
    oss << "&token=" << token;
    oss << "&text=" << text;
    oss << "&format=" << format;
    oss << "&sample_rate=" << sampleRate;
    // voice 发音人，可选，默认是xiaoyun。
    // oss << "&voice=" << "xiaoyun";
```

```
// volume 音量，范围是0~100，可选，默认50。
// oss << "&volume=" << 50;
// speech_rate 语速，范围是-500~500，可选，默认是0。
// oss << "&speech_rate=" << 0;
// pitch_rate 语调，范围是-500~500，可选，默认是0。
// oss << "&pitch_rate=" << 0;
string request = oss.str();
cout << request << endl;
curl_easy_setopt(curl, CURLOPT_URL, request.c_str());
/**
 * 设置获取响应的HTTPS Headers回调函数
 */
map<string, string> responseHeaders;
curl_easy_setopt(curl, CURLOPT_HEADERFUNCTION, responseHeadersCallback);
curl_easy_setopt(curl, CURLOPT_HEADERDATA, &responseHeaders);
/**
 * 设置获取响应的HTTPS Body回调函数
 */
string bodyContent = "";
curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, responseBodyCallback);
curl_easy_setopt(curl, CURLOPT_WRITEDATA, &bodyContent);
/**
 * 发送HTTPS GET请求
 */
res = curl_easy_perform(curl);
/**
 * 释放资源
 */
curl_easy_cleanup(curl);
if (res != CURLE_OK) {
    cerr << "curl_easy_perform failed: " << curl_easy_strerror(res) << endl;
    return -1;
}
/**
 * 处理服务端返回的响应
 */
map<string, string>::iterator it = responseHeaders.find("Content-Type");
if (it != responseHeaders.end() && it->second.compare("audio/mpeg") == 0) {
    ofstream fs;
    fs.open(audioSaveFile.c_str(), ios::out | ios::binary);
    if (!fs.is_open()) {
```

```
            cout << "The audio save file can not open!";
            return -1;
        }
        fs.write(bodyContent.c_str(), bodyContent.size());
        fs.close();
        cout << "The GET request succeed!" << endl;
    }
    else {
        cout << "The GET request failed: " + bodyContent << endl;
        return -1;
    }
    return 0;
}
int processPOSTRequest(string appKey, string token, string text,
                       string audioSaveFile, string format, int sampleRate) {
    CURL* curl = NULL;
    CURLcode res;
    curl = curl_easy_init();
    if (curl == NULL) {
        return -1;
    }
    string url = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts";
    /**
     * 设置HTTPS POST URL
     */
    curl_easy_setopt(curl, CURLOPT_URL, url.c_str());
    curl_easy_setopt(curl, CURLOPT_POST, 1L);
    /**
     * 设置HTTPS POST请求头部
     */
    struct curl_slist* headers = NULL;
    // Content-Type
    headers = curl_slist_append(headers, "Content-Type:application/json");
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);
    /**
     * 设置HTTPS POST请求体
     */
    Json::Value root;
    Json::FastWriter writer;
    root["appkey"] = appKey;
    root["token"] = token;
```

```
root["token"] = token;
root["text"] = text;
root["format"] = format;
root["sample_rate"] = sampleRate;
// voice 发音人，可选，默认是xiaoyun。
// root["voice"] = "xiaoyun";
// volume 音量，范围是0~100，可选，默认50。
// root["volume"] = 50;
// speech_rate 语速，范围是-500~500，可选，默认是0。
// root["speech_rate"] = 0;
// pitch_rate 语调，范围是-500~500，可选，默认是0。
// root["pitch_rate"] = 0;
string task = writer.write(root);
cout << "POST request Body: " << task << endl;
curl_easy_setopt(curl, CURLOPT_POSTFIELDS, task.c_str());
curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, task.length());
/**
 * 设置获取响应的HTTPS Headers回调函数
 */
map<string, string> responseHeaders;
curl_easy_setopt(curl, CURLOPT_HEADERFUNCTION, responseHeadersCallback);
curl_easy_setopt(curl, CURLOPT_HEADERDATA, &responseHeaders);
/**
 * 设置获取响应的HTTPS Body回调函数
 */
string bodyContent = "";
curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, responseBodyCallback);
curl_easy_setopt(curl, CURLOPT_WRITEDATA, &bodyContent);
/**
 * 发送HTTPS POST请求
 */
res = curl_easy_perform(curl);
/**
 * 释放资源
 */
curl_slist_free_all(headers);
curl_easy_cleanup(curl);
if (res != CURLE_OK) {
    cerr << "curl_easy_perform failed: " << curl_easy_strerror(res) << endl;
    return -1;
}
```

```
    /**
     * 处理服务端返回的响应
     */
    map<string, string>::iterator it = responseHeaders.find("Content-Type");
    if (it != responseHeaders.end() && it->second.compare("audio/mpeg") == 0) {
        ofstream fs;
        fs.open(audioSaveFile.c_str(), ios::out | ios::binary);
        if (!fs.is_open()) {
            cout << "The audio save file can not open!";
            return -1;
        }
        fs.write(bodyContent.c_str(), bodyContent.size());
        fs.close();
        cout << "The POST request succeed!" << endl;
    }
    else {
        cout << "The POST request failed: " + bodyContent << endl;
        return -1;
    }
    return 0;
}
int main(int argc, char* argv[]) {
    if (argc < 3) {
        cerr << "params is not valid. Usage: ./demo your_token your_appkey" << endl;
        return -1;
    }
    string token = argv[1];
    string appKey = argv[2];
    string text = "今天是周一，天气挺好的。";
#ifdef _WIN32
    text = GBKToUTF8(text);
#endif
    string textUrlEncode = urlEncode(text);
    stringReplace(textUrlEncode, "+", "%20");
    stringReplace(textUrlEncode, "*", "%2A");
    stringReplace(textUrlEncode, "%7E", "~");
    string audioSaveFile = "syAudio.wav";
    string format = "wav";
    int sampleRate = 16000;
    // 全局只初始化一次。
    curl_global_init(CURL_GLOBAL_ALL);
```

```
    processGETRequest(appKey, token, textUrlEncode, audioSaveFile, format, sampleRate);
    //processPOSTRequest(appKey, token, text, audioSaveFile, format, sampleRate);
    curl_global_cleanup();
    return 0;
}
```

Python示例

> 说明
>
> - Python 2.x请使用httplib模块；Python 3.x请使用http.client模块。
> - 采用RFC 3986规范进行urlencode编码，Python 2.x请使用urllib模块的urllib.quote，Python 3.x请使用urllib.parse模块的urllib.parse.quote_plus。
> - 如果使用内网访问URL，请使用HTTP协议，需要替换如下函数，即将 `HTTPSConnection` 修改为 `HTTPConnection` ：
>
>   ```
>   # Python 2.x 请使用httplib
>   # conn = httplib.HTTPConnection(host)
>
>   # Python 3.x 请使用http.client
>   conn = http.client.HTTPConnection(host)
>   ```

```
# -*- coding: UTF-8 -*-
# Python 2.x引入httplib模块。
# import httplib
# Python 3.x引入http.client模块。
import http.client
# Python 2.x引入urllib模块。
# import urllib
# Python 3.x引入urllib.parse模块。
import urllib.parse
import json
def processGETRequest(appKey, token, text, audioSaveFile, format, sampleRate) :
    host = 'nls-gateway.cn-shanghai.aliyuncs.com'
    url = 'https://' + host + '/stream/v1/tts'
    # 设置URL请求参数
    url = url + '?appkey=' + appKey
    url = url + '&token=' + token
    url = url + '&text=' + text
    url = url + '&format=' + format
    url = url + '&sample_rate=' + str(sampleRate)
```

```
    url = url + '&sample_rate=' + str(sampleRate)
    # voice 发音人，可选，默认是xiaoyun。
    # url = url + '&voice=' + 'xiaoyun'
    # volume 音量，范围是0~100，可选，默认50。
    # url = url + '&volume=' + str(50)
    # speech_rate 语速，范围是-500~500，可选，默认是0。
    # url = url + '&speech_rate=' + str(0)
    # pitch_rate 语调，范围是-500~500，可选，默认是0。
    # url = url + '&pitch_rate=' + str(0)
    print(url)
    # Python 2.x请使用httplib。
    # conn = httplib.HTTPSConnection(host)
    # Python 3.x请使用http.client。
    conn = http.client.HTTPSConnection(host)
    conn.request(method='GET', url=url)
    # 处理服务端返回的响应。
    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 'audio/mpeg' == contentType :
        with open(audioSaveFile, mode='wb') as f:
            f.write(body)
        print('The GET request succeed!')
    else :
        print('The GET request failed: ' + str(body))
    conn.close()
def processPOSTRequest(appKey, token, text, audioSaveFile, format, sampleRate) :
    host = 'nls-gateway.cn-shanghai.aliyuncs.com'
    url = 'https://' + host + '/stream/v1/tts'
    # 设置HTTPS Headers。
    httpHeaders = {
        'Content-Type': 'application/json'
        }
    # 设置HTTPS Body。
    body = {'appkey': appKey, 'token': token, 'text': text, 'format': format, 'sample_rate': sampleRate}
    body = json.dumps(body)
    print('The POST request body content: ' + body)
    # Python 2.x请使用httplib。
```

```
    # conn = httplib.HTTPSConnection(host)
    # Python 3.x请使用http.client。
    conn = http.client.HTTPSConnection(host)
    conn.request(method='POST', url=url, body=body, headers=httpHeaders)
    # 处理服务端返回的响应。
    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 'audio/mpeg' == contentType :
        with open(audioSaveFile, mode='wb') as f:
            f.write(body)
        print('The POST request succeed!')
    else :
        print('The POST request failed: ' + str(body))
    conn.close()
appKey = '您的appkey'
token = '您的token'
text = '今天是周一，天气挺好的。'
# 采用RFC 3986规范进行urlencode编码。
textUrlencode = text
# Python 2.x请使用urllib.quote。
# textUrlencode = urllib.quote(textUrlencode, '')
# Python 3.x请使用urllib.parse.quote_plus。
textUrlencode = urllib.parse.quote_plus(textUrlencode)
textUrlencode = textUrlencode.replace("+", "%20")
textUrlencode = textUrlencode.replace("*", "%2A")
textUrlencode = textUrlencode.replace("%7E", "~")
print('text: ' + textUrlencode)
audioSaveFile = 'syAudio.wav'
format = 'wav'
sampleRate = 16000
# GET请求方式
processGETRequest(appKey, token, textUrlencode, audioSaveFile, format, sampleRate)
# POST请求方式
# processPOSTRequest(appKey, token, text, audioSaveFile, format, sampleRate)
```

PHP示例

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

```
<?php
function processGETRequest($appkey, $token, $text, $audioSaveFile, $format, $sampleRate) {
    $url = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts";
    $url = $url . "?appkey=" . $appkey;
    $url = $url . "&token=" . $token;
    $url = $url . "&text=" . $text;
    $url = $url . "&format=" . $format;
    $url = $url . "&sample_rate=" . strval($sampleRate);
    // voice 发音人，可选，默认是xiaoyun。
    // $url = $url . "&voice=" . "xiaoyun";
    // volume 音量，范围是0~100，可选，默认50。
    // $url = $url . "&volume=" . strval(50);
    // speech_rate 语速，范围是-500~500，可选，默认是0。
    // $url = $url . "&speech_rate=" . strval(0);
    // pitch_rate 语调，范围是-500~500，可选，默认是0。
    // $url = $url . "&pitch_rate=" . strval(0);
    print $url . "\n";
    $curl = curl_init();
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);
    /**
     * 设置HTTPS GET URL。
     */
    curl_setopt($curl, CURLOPT_URL, $url);
    /**
     * 设置返回的响应包含HTTPS头部信息。
     */
    curl_setopt($curl, CURLOPT_HEADER, TRUE);
    /**
     * 发送HTTPS GET请求。
     */
    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, $headerSize);
    curl_close($curl);
    if (stripos($headers, "Content-Type: audio/mpeg") != FALSE || stripos($headers, "Content-Type:aud
io/mpeg") != FALSE) {
        file_put_contents($audioSaveFile, $bodyContent);
        print "The GET request succeed!\n";
    }
    else {
        print "The GET request failed: " . $bodyContent . "\n";
    }
}
function processPOSTRequest($appkey, $token, $text, $audioSaveFile, $format, $sampleRate) {
    $url = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts";
    /**
     * 请求参数，以JSON格式字符串填入HTTPS POST请求的Body中。
     */
    $taskArr = array(
        "appkey" => $appkey,
        "token" => $token,
        "text" => $text,
        "format" => $format,
        "sample_rate" => $sampleRate
        // voice 发音人，可选，默认是xiaoyun。
        // "voice" => "xiaoyun",
        // volume 音量，范围是0~100，可选，默认50。
        // "volume" => 50,
        // speech_rate 语速，范围是-500~500，可选，默认是0。
        // "speech_rate" => 0,
        // pitch_rate 语调，范围是-500~500，可选，默认是0。
        // "pitch_rate" => 0
    );
    $body = json_encode($taskArr);
    print "The POST request body content: " . $body . "\n";
    $curl = curl_init();
```

```
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);
    /**
     * 设置HTTPS POST URL。
     */
    curl_setopt($curl, CURLOPT_URL, $url);
    curl_setopt($curl, CURLOPT_POST, TRUE);
    /**
     * 设置HTTPS POST请求头部。
     * */
    $httpHeaders = array(
        "Content-Type: application/json"
    );
    curl_setopt($curl, CURLOPT_HTTPHEADER, $httpHeaders);
    /**
     * 设置HTTPS POST请求体。
     */
    curl_setopt($curl, CURLOPT_POSTFIELDS, $body);
    /**
     * 设置返回的响应包含HTTPS头部信息。
     */
    curl_setopt($curl, CURLOPT_HEADER, TRUE);
    /**
     * 发送HTTPS POST请求。
     */
    $response = curl_exec($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, $headerSize);
    curl_close($curl);
    if (stripos($headers, "Content-Type: audio/mpeg") != FALSE || stripos($headers, "Content-Type:aud
io/mpeg") != FALSE) {
        file_put_contents($audioSaveFile, $bodyContent);
        print "The POST request succeed!\n";
```

```
    }
    else {
        print "The POST request failed: " . $bodyContent . "\n";
    }
}
$appkey = "您的appkey";
$token = "您的token";
$text = "今天是周一，天气挺好的。";
$textUrlEncode = urlencode($text);
$textUrlEncode = preg_replace('/\+/', '%20', $textUrlEncode);
$textUrlEncode = preg_replace('/\*/', '%2A', $textUrlEncode);
$textUrlEncode = preg_replace('/%7E/', '~', $textUrlEncode);
$audioSaveFile = "syAudio.wav";
$format = "wav";
$sampleRate = 16000;
processGETRequest($appkey, $token, $textUrlEncode, $audioSaveFile, $format, $sampleRate);
// processPOSTRequest($appkey, $token, $text, $audioSaveFile, $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');
function processGETRequest(appkey, token, text, audioSaveFile, format, sampleRate) {
    var url = 'https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts';
    /**
     * 设置URL请求参数。
     */
    url = url + '?appkey=' + appkey;
    url = url + '&token=' + token;
    url = url + '&text=' + text;
    url = url + '&format=' + format;
    url = url + '&sample_rate=' + sampleRate;
    // voice 发音人，可选，默认是xiaoyun。
    // url = url + "&voice=" + "xiaoyun";
```

```
    // url = url + "&voice=" + "xiaoyun";
    // volume 音量，范围是0~100，可选，默认50。
    // url = url + "&volume=" + 50;
    // speech_rate 语速，范围是-500~500，可选，默认是0。
    // url = url + "&speech_rate=" + 0;
    // pitch_rate 语调，范围是-500~500，可选，默认是0。
    // url = url + "&pitch_rate=" + 0;
    console.log(url);
    /**
     * 设置HTTPS GET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: 'GET',
        encoding: null
    };
    request(options, function (error, response, body) {
        /**
         * 处理服务端的响应。
         */
        if (error != null) {
            console.log(error);
        }
        else {
            var contentType = response.headers['content-type'];
            if (contentType === undefined || contentType != 'audio/mpeg') {
                console.log(body.toString());
                console.log('The GET request failed!');
            }
            else {
                fs.writeFileSync(audioSaveFile, body);
                console.log('The GET request is succeed!');
            }
        }
    });
}
function processPOSTRequest(appkeyValue, tokenValue, textValue, audioSaveFile, formatValue, samp
leRateValue) {
    var url = 'https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts';
    /**
```

```
 * 请求参数，以JSON格式字符串填入HTTPS POST请求的Body中。
*/
var task = {
    appkey : appkeyValue,
    token : tokenValue,
    text : textValue,
    format : formatValue,
    sample_rate : sampleRateValue
    // voice 发音人，可选，默认是xiaoyun。
    // voice : 'xiaoyun',
    // volume 音量，范围是0~100，可选，默认50。
    // volume : 50,
    // speech_rate 语速，范围是-500~500，可选，默认是0。
    // speech_rate : 0,
    // pitch_rate 语调，范围是-500~500，可选，默认是0。
    // pitch_rate : 0
};
var bodyContent = JSON.stringify(task);
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 {
        var contentType = response.headers['content-type'];
        if (contentType === undefined || contentType != 'audio/mpeg') {
          console.log(body.toString());
          console.log('The POST request failed!');
        }
        else {
          fs.writeFileSync(audioSaveFile, body);
          console.log('The POST request is succeed!');
        }
      }
    });
}
var appkey = '您的appkey';
var token = '您的token';
var text = '今天是周一，天气挺好的。';
var textUrlEncode = encodeURIComponent(text)
                .replace(/[!'()*]/g, function(c) {
                  return '%' + c.charCodeAt(0).toString(16);
                });
console.log(textUrlEncode);
var audioSaveFile = 'syAudio.wav';
var format = 'wav';
var sampleRate = 16000;
processGETRequest(appkey, token, textUrlEncode, audioSaveFile, format, sampleRate);
// processPOSTRequest(appkey, token, text, audioSaveFile, format, sampleRate);
```

.Net示例

> 说明
>
> 示例使用依赖System.Net.Http、System.Web、Newtonsoft.Json.Linq。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Net.Http;
```

```
using System.Web;
using Newtonsoft.Json.Linq;
namespace RESTfulAPI
{
    class SpeechSynthesizerRESTfulDemo
    {
        private string appkey;
        private string token;
        public SpeechSynthesizerRESTfulDemo(string appkey, string token)
        {
            this.appkey = appkey;
            this.token = token;
        }
        public void processGETRequest(string text, string audioSaveFile, string format, int sampleRate)
        {
            /**
             * 设置HTTPS GET请求：
             * 1.使用HTTPS协议
             * 2.语音识别服务域名：nls-gateway.cn-shanghai.aliyuncs.com
             * 3.语音识别接口请求路径：/stream/v1/tts
             * 4.设置必须请求参数：appkey、token、text、format、sample_rate
             * 5.设置可选请求参数：voice、volume、speech_rate、pitch_rate
             */
            string url = "http://nls-gateway.aliyuncs.com/stream/v1/tts";
            url = url + "?appkey=" + appkey;
            url = url + "&token=" + token;
            url = url + "&text=" + text;
            url = url + "&format=" + format;
            url = url + "&sample_rate=" + sampleRate.ToString();
            // voice 发音人，可选，默认是xiaoyun。
            // url = url + "&voice=" + "xiaoyun";
            // volume 音量，范围是0~100，可选，默认50。
            // url = url + "&volume=" + 50;
            // speech_rate 语速，范围是-500~500，可选，默认是0。
            // url = url + "&speech_rate=" + 0;
            // pitch_rate 语调，范围是-500~500，可选，默认是0。
            // url = url + "&pitch_rate=" + 0;
            System.Console.WriteLine(url);
            /**
             * 发送HTTPS GET请求，处理服务端的响应。
             */
```

```
        */
        HttpClient client = new HttpClient();
        HttpResponseMessage response = null;
        response = client.GetAsync(url).Result;
        string contentType = null;
        if (response.IsSuccessStatusCode)
        {
            string[] typesArray = response.Content.Headers.GetValues("Content-Type").ToArray();
            if (typesArray.Length > 0)
            {
                contentType = typesArray.First();
            }
        }
        if ("audio/mpeg".Equals(contentType))
        {
            byte[] audioBuff = response.Content.ReadAsByteArrayAsync().Result;
            FileStream fs = new FileStream(audioSaveFile, FileMode.Create);
            fs.Write(audioBuff, 0, audioBuff.Length);
            fs.Flush();
            fs.Close();
            System.Console.WriteLine("The GET request succeed!");
        }
        else
        {
            // ContentType 为 null 或者为 "application/json"
            System.Console.WriteLine("Response status code and reason phrase: " +
                response.StatusCode + " " + response.ReasonPhrase);
            string responseBodyAsText = response.Content.ReadAsStringAsync().Result;
            System.Console.WriteLine("The GET request failed: " + responseBodyAsText);
        }
    }
    public void processPOSTRequest(string text, string audioSaveFile, string format, int sampleRate)
    {
        /**
         * 设置HTTPS POST请求：
         * 1.使用HTTPS协议
         * 2.语音合成服务域名：nls-gateway.cn-shanghai.aliyuncs.com
         * 3.语音合成接口请求路径：/stream/v1/tts
         * 4.设置必须请求参数：appkey、token、text、format、sample_rate
         * 5.设置可选请求参数：voice、volume、speech_rate、pitch_rate
         */
```

```
string url = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts";
JObject obj = new JObject();
obj["appkey"] = appkey;
obj["token"] = token;
obj["text"] = text;
obj["format"] = format;
obj["sample_rate"] = sampleRate;
// voice 发音人，可选，默认是xiaoyun。
// obj["voice"] = "xiaoyun";
// volume 音量，范围是0~100，可选，默认50。
// obj["volume"] = 50;
// speech_rate 语速，范围是-500~500，可选，默认是0。
// obj["speech_rate"] = 0;
// pitch_rate 语调，范围是-500~500，可选，默认是0。
// obj["pitch_rate"] = 0;
String bodyContent = obj.ToString();
StringContent content = new StringContent(bodyContent, Encoding.UTF8, "application/json");
/**
 * 发送HTTPS POST请求，处理服务端的响应。
 */
HttpClient client = new HttpClient();
HttpResponseMessage response = client.PostAsync(url, content).Result;
string contentType = null;
if (response.IsSuccessStatusCode)
{
   string[] typesArray = response.Content.Headers.GetValues("Content-Type").ToArray();
   if (typesArray.Length > 0)
   {
      contentType = typesArray.First();
   }
}
if ("audio/mpeg".Equals(contentType))
{
   byte[] audioBuff = response.Content.ReadAsByteArrayAsync().Result;
   FileStream fs = new FileStream(audioSaveFile, FileMode.Create);
   fs.Write(audioBuff, 0, audioBuff.Length);
   fs.Flush();
   fs.Close();
   System.Console.WriteLine("The POST request succeed!");
}
else
```

```
        {
            System.Console.WriteLine("Response status code and reason phrase: " +
                response.StatusCode + " " + response.ReasonPhrase);
            string responseBodyAsText = response.Content.ReadAsStringAsync().Result;
            System.Console.WriteLine("The POST request failed: " + responseBodyAsText);
        }
    }
    static void Main(string[] args)
    {
        if (args.Length < 2)
        {
            System.Console.WriteLine("SpeechSynthesizerRESTfulDemo need params: <token> <app-key
>");
            return;
        }
        string token = args[0];
        string appkey = args[1];
        SpeechSynthesizerRESTfulDemo demo = new SpeechSynthesizerRESTfulDemo(appkey, token);
        string text = "今天是周一，天气挺好的。";
        // 采用RFC 3986规范进行urlencode编码。
        string textUrlEncode = text;
        textUrlEncode = HttpUtility.UrlEncode(textUrlEncode, Encoding.UTF8)
            .Replace("+", "%20")
            .Replace("*", "%2A")
            .Replace("%7E", "~");
        System.Console.WriteLine(textUrlEncode);
        string audioSaveFile = "syAudio.wav";
        string format = "wav";
        int sampleRate = 16000;
        demo.processGETRequest(textUrlEncode, audioSaveFile, format, sampleRate);
        //demo.processPOSTRequest(text, audioSaveFile, format, sampleRate);
    }
  }
}
```

GO示例

```
package main
import (
  "fmt"
  "net/url"
```

```
    "net/url"
    "net/http"
    "io/ioutil"
    "encoding/json"
    "strconv"
    "os"
    "bytes"
    "strings"
)
func processGETRequest(appkey string, token string, text string, audioSaveFile string, format string, s
ampleRate int) {
    /**
     * 设置HTTPS GET请求：
     * 1.使用HTTPS协议
     * 2.语音识别服务域名：nls-gateway.cn-shanghai.aliyuncs.com
     * 3.语音识别接口请求路径：/stream/v1/tts
     * 4.设置必须请求参数：appkey、token、text、format、sample_rate
     * 5.设置可选请求参数：voice、volume、speech_rate、pitch_rate
     */
    var url string = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts"
    url = url + "?appkey=" + appkey
    url = url + "&token=" + token
    url = url + "&text=" + text
    url = url + "&format=" + format
    url = url + "&sample_rate=" + strconv.Itoa(sampleRate)
    // voice 发音人，可选，默认是xiaoyun。
    // url = url + "&voice=" + "xiaoyun"
    // volume 音量，范围是0~100，可选，默认50。
    // url = url + "&volume=" + strconv.Itoa(50)
    // speech_rate 语速，范围是-500~500，可选，默认是0。
    // url = url + "&speech_rate=" + strconv.Itoa(0)
    // pitch_rate 语调，范围是-500~500，可选，默认是0。
    // url = url + "&pitch_rate=" + strconv.Itoa(0)
    fmt.Println(url)
    /**
     * 发送HTTPS GET请求，处理服务端的响应。
     */
    response, err := http.Get(url)
    if err != nil {
        fmt.Println("The GET request failed!")
        panic(err)
```

```
    }
    defer response.Body.Close()
    contentType := response.Header.Get("Content-Type")
    body, _ := ioutil.ReadAll(response.Body)
    if ("audio/mpeg" == contentType) {
        file, _ := os.Create(audioSaveFile)
        defer file.Close()
        file.Write([]byte(body))
        fmt.Println("The GET request succeed!")
    } else {
        // ContentType 为 null 或者为 "application/json"
        statusCode := response.StatusCode
        fmt.Println("The HTTP statusCode: " + strconv.Itoa(statusCode))
        fmt.Println("The GET request failed: " + string(body))
    }
}
func processPOSTRequest(appkey string, token string, text string, audioSaveFile string, format string,
sampleRate int) {
    /**
     * 设置HTTPS POST请求：
     * 1.使用HTTPS协议
     * 2.语音合成服务域名：nls-gateway.cn-shanghai.aliyuncs.com
     * 3.语音合成接口请求路径：/stream/v1/tts
     * 4.设置必须请求参数：appkey、token、text、format、sample_rate
     * 5.设置可选请求参数：voice、volume、speech_rate、pitch_rate
     */
    var url string = "https://nls-gateway.cn-shanghai.aliyuncs.com/stream/v1/tts"
    bodyContent := make(map[string]interface{})
    bodyContent["appkey"] = appkey
    bodyContent["text"] = text
    bodyContent["token"] = token
    bodyContent["format"] = format
    bodyContent["sample_rate"] = sampleRate
    // voice 发音人，可选，默认是xiaoyun。
    // bodyContent["voice"] = "xiaoyun"
    // volume 音量，范围是0~100，可选，默认50。
    // bodyContent["volume"] = 50
    // speech_rate 语速，范围是-500~500，可选，默认是0。
    // bodyContent["speech_rate"] = 0
    // pitch_rate 语调，范围是-500~500，可选，默认是0。
    // bodyContent["pitch_rate"] = 0
```

```
    bodyJson, err := json.Marshal(bodyContent)
    if err != nil {
        panic(nil)
    }
    fmt.Println(string(bodyJson))
    /**
     * 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(bodyJson)))
    if err != nil {
        panic(err)
    }
    defer response.Body.Close()
    contentType := response.Header.Get("Content-Type")
    body, _ := ioutil.ReadAll(response.Body)
    if ("audio/mpeg" == contentType) {
        file, _ := os.Create(audioSaveFile)
        defer file.Close()
        file.Write([]byte(body))
        fmt.Println("The POST request succeed!")
    } else {
        // ContentType 为 null 或者为 "application/json"
        statusCode := response.StatusCode
        fmt.Println("The HTTP statusCode: " + strconv.Itoa(statusCode))
        fmt.Println("The POST request failed: " + string(body))
    }
}
func main() {
    var appkey string = "您的appkey"
    var token  string = "您的token"
    var text string = "今天是周一，天气挺好的。"
    var textUrlEncode = text
    textUrlEncode = url.QueryEscape(textUrlEncode)
    textUrlEncode = strings.Replace(textUrlEncode, "+", "%20", -1)
    textUrlEncode = strings.Replace(textUrlEncode, "*", "%2A", -1)
    textUrlEncode = strings.Replace(textUrlEncode, "%7E", "~", -1)
    fmt.Println(textUrlEncode)
    var audioSaveFile string = "syAudio.wav"
    var format string = "wav"
    var sampleRate int = 16000
    processGETRequest(appkey, token, textUrlEncode, audioSaveFile, format, sampleRate)
```

```
    processGETRequest(appkey, token, textUrlEncode, audioSaveFile, format, sampleRate)
    // processPOSTRequest(appkey, token, text, audioSaveFile, format, sampleRate)
}
```

8.SSML标记语言介绍

本文为您介绍SSML（Speech Synthesis Markup Language）标记语言的功能、标签使用及示例。

概述

SSML是一种基于XML的标记语言。 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，更让人可以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。但并不支持所有的W3C包含的标记类型，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
>
> - 目前只有中文合成支持SSML功能。
> - 所有文本需放在<speak></speak>标签之内，且每个语音合成任务只能包含一个<speak></speak>标签。

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": "fd4er4aa****",
    "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

```
# 空属性
<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16KHz、单声道WAV格式。 ■ 不超过2MB。 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：
			`<`
			`>`
			`&amp;`
			`"`
			`'`
			，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
O	欧麦克轮
Π	派
P	柔
Σ	西格玛
T	套
Y	宇普西龙
Φ	fai
X	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>
```

9.语音合成时间戳功能介绍

语音实时合成服务在输出音频流的同时，可输出每个字在音频中的时间位置，即时间戳，时间戳功能又叫字级别音素边界接口。该时间信息可用于驱动虚拟人口型、做视频配音字幕等。

> 注意
>
> 只有支持字级别音素边界接口的发音人才有此功能。

参数设置

在客户端设置请求参数 `enable_subtitle` 为 `true` ，开启时间戳功能。

以Java SDK为例，其设置方式如下。

```
// 是否开启字幕功能（返回对应文本的相应时间戳），默认不开启。
synthesizer.addCustomedParam("enable_subtitle", true);
```

服务端响应

服务端返回的带字幕信息的响应MetaInfo事件。

参数	类型	说明
subtitles	List	时间戳信息。

其中，SubtitleItem格式如下。

参数	类型	说明
text	String	文本信息。
begin_time	Integer	文本对应tts语音开始时间戳，单位ms。
end_time	Integer	文本对应tts语音结束时间戳，单位ms。

示例

```
{
  "header": {
    "message_id": "05450bf69c53413f8d88aed1ee60****",
    "task_id": "640bc797bb684bd6960185651307****",
    "namespace": "SpeechSynthesizer",
    "name": "MetaInfo",
    "status": 20000000,
    "status_message": "GATEWAY|SUCCESS|Success."
  },
  "payload": {
    "subtitles": [
      {
        "text": "xx",
        "begin_time": 130,
        "end_time": 260
      },
      {
        "text": "xx",
        "begin_time": 260,
        "end_time": 370
      }
    ]
  }
}
```


注意事项

- TTS服务返回的字幕是基于发音的，所以不能直接用于上屏，需要使用您的原始文本。
- 如果用于上屏，可以基于返回的结果，定位每个句子的句首和句尾时间戳。

10.移动端NUI SDK

10.1. 接口说明

语音合成提供将输入文本合成为语音二进制数据的功能。

功能简介

NUI SDK提供更小的工具包和更完善的状态管理。为满足不同用户需求，NUI SDK既能够提供全链路的语音能力，同时可做原子能力SDK进行使用，并保持接口的统一。

语音合成功能支持如下能力：

- 支持输出PCM、MP3编码格式数据。
- 支持设置语速、语调、音量。
- 支持设置声音类型，如下表所示。

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	支持字级别音素边界接口	备注
小云	Xiaoyun	标准女声	通用场景	中文及中英文混合	8K/16K	否	无
小刚	Xiaogang	标准男声	通用场景	中文及中英文混合	8K/16K	否	无
若兮	Ruoxi	温柔女声	通用场景	中文及中英文混合	8K/16K/24K	否	无
思琪	Siqi	温柔女声	通用场景	中文及中英文混合	8K/16K/24K	是	无
思佳	Sijia	标准女声	通用场景	中文及中英文混合	8K/16K/24K	否	无
思诚	Sicheng	标准男声	通用场景	中文及中英文混合	8K/16K/24K	是	无
艾琪	Aiqi	温柔女声	通用场景	中文及中英文混合	8K/16K	是	无

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	支持字级别音素边界接口	备注
艾佳	Aijia	标准女声	通用场景	中文及中英文混合	8K/16K	是	无
艾诚	Aicheng	标准男声	通用场景	中文及中英文混合	8K/16K	是	无
艾达	Aida	标准男声	通用场景	中文及中英文混合	8K/16K	是	无
宁儿	Ninger	标准女声	通用场景	中文	8K/16K/24K	否	无
瑞琳	Ruilin	标准女声	通用场景	中文	8K/16K/24K	否	无
思悦	Siyue	温柔女声	客服场景	中文及中英文混合	8K/16K/24K	否	无
艾雅	Aiya	严厉女声	客服场景	中文及中英文混合	8K/16K	是	无
艾夏	Aixia	亲和女声	客服场景	中文及中英文混合	8K/16K	是	无
艾美	Aimei	甜美女声	客服场景	中文及中英文混合	8K/16K	是	无
艾雨	Aiyu	自然女声	客服场景	中文及中英文混合	8K/16K	是	无
艾悦	Aiyue	温柔女声	客服场景	中文及中英文混合	8K/16K	是	无
艾婧	Aijing	严厉女声	客服场景	中文及中英文混合	8K/16K	是	无

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

名称	voice参数值	类型	适用场景	支持语言	支持采样率(Hz)	支持字级别音素边界接口	备注
Luna	Luna	英音女声	英文场景	英文	8K/16K	否	无
Luca	Luca	英音男声	英文场景	英文	8K/16K	否	无
Wendy	Wendy	英音女声	英文场景	英文	8K/16K/24K	否	无
William	William	英音男声	英文场景	英文	8K/16K/24K	否	无
Olivia	Olivia	英音女声	英文场景	英文	8K/16K/24K	否	无
姗姗	Shanshan	粤语女声	方言场景	标准粤文（简体）及粤英文混合	8K/16K/24K	否	无
小玥	Xiaoyue	四川话女声	方言场景	中文及中英文混合	8K/16K	否	公测版
Lydia	Lydia	英中双语女声	英文场景	英文	8K/16K	否	公测版
艾硕	Aishuo	自然男声	客服场景	中文及中英文混合	8K/16K	是	公测版
青青	Qingqing	台湾话女声	方言场景	中文	8K/16K	否	公测版
翠姐	Cuijie	东北话女声	方言场景	中文	8K/16K	否	公测版
小泽	Xiaoze	湖南重口音男声	方言场景	中文	8K/16K	是	公测版

限制条件

- 传入的文本必须采用 `UTF-8` 编码。
- 传入的文本不能超过300个字符，超过300个字符的内容会被截断。

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）。	wss://nls-gateway.cn-shanghai.aliyuncs.com/ws/v1
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

服务端响应除了音频流之外，均会在返回信息的header包含表示本次识别任务的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。关于Token获取请参见获取Token。

初始化参数如下。

参数	类型	是否必选	说明
workspace	String	是	工作目录路径，SDK从该路径读取配置文件。

参数	类型	是否必选	说明
app_key	String	是	管控台创建项目的appkey。
token	String	是	请确保该token可以使用并在有效期内。 Token可以在初始化时设置，也可通过参数设置进行更新。
device_id	String	是	设备标识，能够唯一表示一台设备（如Mac地址/SN/UniquePsuedoID）。

2. 开始合成

客户端发起语音合成请求，在请求消息中进行参数设置，各参数通过SDK中setparamTts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。
token	String	否	如果需要更新，则进行设置。
direct_host	String	否	支持客户端自行DNS解析后传入IP进行访问。
font_name	String	否	发音人，默认是xiaoyun。
encode_type	String	否	音频编码格式，默认值：PCM。支持格式：PCM、WAV、MP3。
sample_rate	String	否	音频采样率，默认值：16000。

参数	类型	是否必选	说明
volume	String	否	音量，取值范围：0～2，默认值：1.0。
speed_level	String	否	语速，取值范围：0～2，默认值：1.0，值越大语速越快。
pitch_level	String	否	语调，取值范围：-500～500，默认值：0，值越大声音越尖锐。
enable_subtitle	String	否	字级别音素边界功能开关，该参数只对支持字级别音素边界接口的发音人有效。“1”表示打开，“0”表示关闭。

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。

错误码

如果语音合成发生错误，SDK将上报TTS_EVENT_ERROR事件，并提供错误信息，如下表所示。

错误码	错误消息	描述
0	TTS_SUCCESS	成功
140000	TTS_CREATE_FAILED	引擎初始化失败
140001	TTS_ENGINE_INVALID	引擎未初始化
140002	TTS_TEXT_ERROR	文本非法，如文本为空。
140003	TTS_MALLOC_FAILED	内存申请失败

错误码	错误消息	描述
140004	TTS_TEXT_QUEUE_FULL	任务队列已满
140005	TTS_ASSETPATH_INVALID	资源路径为空
140006	TTS_HANLDE_INVALID	处理线程不存在
140007	TTS_CREATE_HANLDE_FAILED	创建处理线程失败
140008	TTS_AUTH_FAILED	鉴权失败，无法继续使用SDK。
140009	TTS_TEXT_QUEUE_EMPTY	合成队列已空
140010	TTS_MODE_INVALID	合成模式无效
140012	TTS_OPEN_FILE_FAILED	打开文件失败
140013	TTS_STATE_INVALID	SDK的状态机校验失败
140014	TTS_SYNTHESIZER_INIT_ERROR	合成器初始化失败
140015	TTS_SYNTHESIZER_RELEASE_ERROR	合成器释放失败
140016	TTS_SYNTHESIZER_FAILED	合成失败
140017	TTS_WAIT_TIMEOUT	超时退出
140018	TTS_CLOSED	没有编译语音合成部分代码
140100	TTS_PARAM_INVALID	参数无效
140101	TTS_PARAM_VALUE_INVALID	参数值无效
140102	TTS_CFG_OPEN_FAILED	配置文件打开失败

错误码	错误消息	描述
140103	TTS_CFG_WRONG_FORMAT	配置文件格式错误
140150	TTS_LOG_OPEN_FAILED	创建日志文件失败
140200	TTS_AM_CREATE_FAILED	播放器创建失败
140201	TTS_AM_OPEN_FAILED	播放器打开失败
140210	TTS_DECODER_INIT_FAILED	音频解码器初始化失败
140211	TTS_DECODER_MALLOC_FAILED	音频解码器申请内存失败
140212	TTS_DECODER_INPUT_TOO_MANY	单次输入数据过多，将被丢弃。
140213	TTS_DECODER_OUTPUT_TOO_MANY	输出数据过多，超过缓存，会丢失。
140220	TTS_AP_INIT_FAILED	音频处理单元打开失败
140221	TTS_AP_START_FAILED	音频处理单元启动出错
140222	TTS_AP_MALLOC_FAILED	音频处理单元内存申请失败
140230	TTS_BGM_START_FAILED	背景音乐开始失败
140231	TTS_BGM_DECODE_INVALID	解码器初始化失败
140232	TTS_BGM_ADD_FAILED	本句加背景音乐失败
140233	TTS_BGM_MALLOC_FAILED	内存申请失败
140234	TTS_BGM_OPEN_FILE_FAILED	背景音乐文件打开失败

错误码	错误消息	描述
140235	TTS_BGM_FILE_FORMAT_ERROR	背景音乐格式错误
140300	TTS_CACHE_INIT_FAILED	初始化cache失败
140301	TTS_CACHE_MGR_INVALID	cache管理器未初始化
140303	TTS_CACHE_CALLBACK_INVALID	回调函数未初始化
140304	TTS_CACHE_START_READ_FAILED	开始打开cache文件失败
140305	TTS_CACHE_READ_FAILED	读取cache文件数据失败
140306	TTS_CACHE_MALLOC_FAILED	cache模块内存申请失败
140307	TTS_CACHE_DELETE_FAILED	删除失败
140308	TTS_CACHE_PATH_INVALID	无法创建缓存路径
140309	TTS_CACHE_LIST_CREATE_FAILED	cache列表创建失败
140310	TTS_CACHE_FAILED	缓存失败
140311	TTS_CACHE_TOO_MANY	缓存过多
140312	TTS_CACHE_PARAM_INVALID	缓存功能的参数设置错误
140313	TTS_CACHE_RECORDING_OPEN_FAILED	打开本地文件错误
140352	TTS_FONT_INITLIST_INVALID	fontlist管理器未初始化
140353	TTS_FONT_CMD_INVALID	命令格式错误

错误码	错误消息	描述
140354	TTS_FONT_RESPONSE_ERROR	服务端返回格式错误
140355	TTS_FONT_RESPONSELIST_ERROR	fontlist请求服务端返回格式错误
140356	TTS_FONT_GET_FONTLIST_FAILED	获取fontlist失败
140357	TTS_FONT_REQUEST_CMD_ERROR	创建请求指令错误
140358	TTS_FONT_LOCALMSG_ERROR	本地list文件解析失败
140360	TTS_FONT_CLOUDMSG_ERROR	云端list解析失败
140900	TTS_LOCAL_CRE_ENGINE_ERROR	本地引擎初始化失败
140901	TTS_LOCAL_ENGINE_INVALID	本地引擎未初始化
140902	TTS_LOCAL_ASSET_ERROR	本地资源校验失败
140903	TTS_LOCAL_CRE_TASK_ERROR	创建本地task失败
140904	TTS_LOCAL_TASK_INVALID	task无效
140905	TTS_LOCAL_START_FAILED	本地开始合成失败
141000	TTS_CLOUD_CREATE_FAILED	云端引擎初始化失败
141001	TTS_CLOUD_ENGINE_INVALID	云端引擎未初始化
141002	TTS_CLOUD_TASK_FAILED	创建云端task失败
141003	TTS_CLOUD_TASK_INVALID	task无效

错误码	错误消息	描述
141004	TTS_CLOUD_START_FAILED	云端服务请求失败
141005	TTS_CLOUD_CANCEL_FAILED	取消任务失败
141006	TTS_CLOUD_NETWORK_BROKEN	网络较差
144001	TTS_CLOUD_AUTH_FAILED	身份认证失败
144002	TTS_CLOUD_INVALID_MESSAGE	消息无效
144003	TTS_CLOUD_INVALID_TOKEN	令牌过期或无效
144004	TTS_CLOUD_WAIT_TIMEOUT	空闲超时
144005	TTS_CLOUD_EXCEED_CONCURRENCY	请求数量过多
144100	TTS_CLOUD_INVALID_INTERFACE	接口不支持
144101	TTS_CLOUD_UNSUPPORTED_ORDER	指令不支持
144102	TTS_CLOUD_INVALID_ORDER	指令无效
144103	TTS_CLOUD_CLIENT_DISCONNECT	客户端提前断开连接
144200	TTS_CLOUD_INVALID_APPKEY	应用不存在
144300	TTS_CLOUD_INVALID_PARAM	访问云端服务参数错误
144400	TTS_CLOUD_SERVER_ERROR	语音合成服务端出错

10.2. Android SDK

本文介绍了如何使用阿里云智能语音服务提供的Android NUI SDK，包括SDK下载安装、关键接口及代码示例。

前提条件

- 阅读接口说明，详情请参见接口说明。
- 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。

下载安装

1. 下载SDK和示例代码。
2. 解压ZIP包，在app/libs目录下获取AAR格式的SDK包。
3. 使用Android Studio打开此工程。

   其中语音合成示例代码为TtsBasicActivity.java文件。

SDK关键接口

- tts_initialize：初始化SDK。

```
/**
    * 初始化SDK，SDK为单例，请先释放后再次进行初始化。请勿在UI线程调用，可能会引起阻塞。
    * @param callback：事件监听回调，参见下文具体回调。
    * @param ticket：初始化参数，参见接口说明。
    * @param level：log打印级别，值越小打印越多。
    * @param save_log：是否保存log为文件，存储目录为parameter中的debug_path字段值。
    * @return：参见错误码。
    */
   public synchronized int tts_initialize(INativeTtsCallback callback,
                                          String ticket,
                                          final Constants.LogLevel level,
                                          boolean save_log)
```

其中，INativeTtsCallback类型包含如下回调。

- onTtsEventCallback：SDK事件回调。

```
/**
    * 事件回调
    * @param event：回调事件，参见如下事件列表。
    * @param task_id：请求的任务ID。
    * @param ret_code：参见错误码，出现TTS_EVENT_ERROR事件时有效。
    */
   void onTtsEventCallback(TtsEvent event, String task_id, int ret_code);
```

事件列表：

名称	说明
TTS_EVENT_START	语音合成开始，准备播放。
TTS_EVENT_END	语音合成播放结束
TTS_EVENT_CANCEL	取消语音合成
TTS_EVENT_PAUSE	语音合成暂停
TTS_EVENT_RESUME	语音合成恢复
TTS_EVENT_ERROR	语音合成发生错误

- onTtsDataCallback：合成数据回调。

```
/**
    * 合成数据回调
    * @param info：使用时间戳功能时，返回JSON格式的时间戳结果。
    * @param info_len：info字段的数据长度，暂不使用。
    * @param data：合成的音频数据，写入播放器。
    */
   void onTtsDataCallback(String info, int info_len, byte[] data);
```

- setparamTts：设置参数。

```
/**
    * 以键值对形式设置参数
    * @param param：参数名，参见接口说明。
    * @param value：参数值，参见接口说明。
    * @return：参见错误码。
    */
   public synchronized int setparamTts(String param, String value);
```

- **getparamTts**：获取参数。

```
/**
    * 获取参数值
    * @param param：参数名，参考接口说明。
    * @return：参数值。
    */
   public String getparamTts(String param);
```

- **startTts**：开始播放。

```
/**
    * 开始播放
    * @param priority：任务优先级，请使用"1"。
    * @param taskid：任务ID，可传入32个字节的uuid，或传入空内容由SDK自动生成。
    * @param text：播放的文本内容。
    * @return：参见错误码。
    */
   public synchronized int startTts(String priority, String taskid, String text)
```

- **cancelTts**：取消播放。

```
/**
    * 取消合成任务
    * @param taskid：传入想要停止的任务ID，如果为空则取消所有任务。
    * @return：参见错误码。
    */
   public synchronized int cancelTts(String taskid)
```

- **pauseTts**：暂停播放。

```
/**
    * 暂停
    * @return：参见错误码。
    */
    public synchronized int pauseTts()
```

- **resumeTts**：恢复播放。

```
/**
    * 恢复暂停的任务
    * @return：参见错误码。
    */
    public synchronized int resumeTts()
```

- **tts_release**：释放SDK资源。

```
/**
    * 释放SDK
    * @return：参见错误码。
    */
    public synchronized int tts_release()
```

调用步骤

1. 初始化SDK和播放组件。
2. 根据业务需求设置参数。
3. 调用startTts进行播放。
4. 在合成数据回调中，将数据写入播放器进行播放，建议使用流式播放。
5. 收到语音合成结束的回调。

代码示例

语音合成初始化

```
//拷贝资源
CommonUtils.copyAssetsData(this);
//SDK初始化
int ret = NativeNui.GetInstance().tts_initialize(new INativeTtsCallback() {}, genTicket(path), Constants.
LogLevel.LOG_LEVEL_VERBOSE, true);
```

其中，genTicket生成为String JSON字符串，包含资源目录和用户信息。其中用户信息包含如下字段。

```
private String genTicket(String workpath) {
    String str = "";
    try {
        JSONObject object = Auth.getAliYunTicket();
        object.put("workspace", workpath);
        str = object.toString();
    } catch (JSONException e) {
        e.printStackTrace();
    }
    Log.i(TAG, "UserContext:" + str);
    return str;
  }
```

根据需求设置参数

```
NativeNui.GetInstance().setparamTts("font_name", "xiaoyun");
```

启动语音合成

```
NativeNui.GetInstance().startTts("1", "", ttsText);
```

回调处理

- onTtsEventCallback：语音合成事件回调，根据语音合成状态控制播放器。

```
public void onTtsEventCallback(INativeTtsCallback.TtsEvent event) {
            Log.i(TAG, "tts event:" + event);
            if (event == INativeTtsCallback.TtsEvent.TTS_EVENT_START) {
                mAudioTrack.play();
                Log.i(TAG, "start play");
            } else if (event == INativeTtsCallback.TtsEvent.TTS_EVENT_END) {
                Log.i(TAG, "play end");
            } else if (event == TtsEvent.TTS_EVENT_PAUSE) {
                mAudioTrack.pause();
                Log.i(TAG, "play pause");
            } else if (event == TtsEvent.TTS_EVENT_RESUME) {
                mAudioTrack.play();
            }
        }
```

- onTtsDataCallback：语音合成数据回调，将回调中的合成数据写入播放器进行播放。

```
public void onTtsDataCallback(String info, int info_len, byte[] data) {
            if (data.length > 0) {
                mAudioTrack.setAudioData(data);
            }
        }
```

结束语音合成

```
NativeNui.GetInstance().cancelTts("");
```

10.3. iOS SDK

本文介绍了如何使用阿里云智能语音服务提供的iOS NUI SDK，包括SDK下载安装、关键接口及代码示例。

前提条件

- 阅读接口说明，详情请参见接口说明。
- 已准备项目appkey，详情请参见创建项目。
- 已获取Access Token，详情请参见获取Token。

下载安装

1. 下载SDK和示例代码。
2. 解压ZIP包，使用nuisdk.framework进行集成。

   > **说明**
   >
   > 代码采用Ojective-C和C++混合编写，请您调用.mm扩展名文件。

3. 使用Xcode打开此工程。

   其中语音合成示例代码在TTSViewController.mm文件。

SDK关键接口

- **nui_tts_initialize**：初始化SDK。

```
/**
    * 初始化SDK，SDK为单例，请先释放后再次进行初始化。请勿在UI线程调用，可能会引起阻塞。
    * @param parameters：初始化参数，参见接口说明。
    * @param listener：事件监听回调，参见下文具体回调。
    * @param async_listener：异步回调，设置nullptr采用同步方式调用。
    * @param level：log打印级别，值越小打印越多。
    * @param save_log：是否保存log为文件，存储目录为parameter中的debug_path字段值。
    * @return：参见错误码。
    */
   -(int) nui_tts_initialize:(const char *)parameters
            ttsListener:(const NuiTtsSdkListener *)listener
          asyncCallback:(const NuiAsyncCallback *)async_listener
               logLevel:(NuiSdkLogLevel)level
                saveLog:(bool)save_log;
```

其中，NuiTtsSdkListener类型如下表。

名称	类型	说明
tts_event_callback	FuncTtsListenerOnEvent	语音合成事件回调
tts_user_data_callback	FuncTtsUserProvideData	语音合成数据回调
tts_user_volume_callback	FuncTtsUserProvideVolume	音量回调
user_data	void *	用户数据，上述回调中第一个参数。

- FuncTtsListenerOnEvent：SDK事件回调。

```
/**
    * SDK主要事件回调
    * @param user_data：暂不使用。
    * @param event：回调事件，参见如下事件列表。
    * @param taskid：本次合成的任务ID。
    * @param code：参见错误码，出现TTS_EVENT_ERROR事件时有效。
    */
   typedef void (*FuncTtsListenerOnEvent) (void *user_data, NuiSdkTtsEvent event, char *taskid, int code);
```

事件列表：

名称	说明
TTS_EVENT_START	语音合成开始，准备播放。
TTS_EVENT_END	语音合成播放结束
TTS_EVENT_CANCEL	取消语音合成
TTS_EVENT_PAUSE	语音合成暂停
TTS_EVENT_RESUME	语音合成恢复
TTS_EVENT_ERROR	语音合成发生错误

- FuncTtsUserProvideData：回调中输出合成的音频数据。

```
/**
    * 开始合成后，此回调输出合成的音频数据，App可进行播放。
    * @param user_data：暂不使用。
    * @param info：当使用时间戳功能时，将返回JSON格式的时间戳结果。
    * @param info_len：info字段的数据长度。
    * @param buffer：合成的语音数据。
    * @param len：合成的语音长度。
    * @param taskid：本次合成的任务ID。
    */
   typedef void (*FuncTtsUserProvideData) (void *user_data, char *info, int info_len, char *buffer, i
nt len, char *taskid);
```

- FuncTtsUserProvideVolume：合成数据音量回调。

```
/**
    * SDK主要事件回调
    * @param user_data：暂不使用。
    * @param volume：合成语音的音量，可用于音量的动态显示。
    * @param taskid：本次合成的任务ID。
    */
   typedef void (*FuncTtsUserProvideVolume) (void *user_data, int volume, char *taskid);
```

- nui_tts_play：开始播放。

```
/**
    * 开始播放
    * @param priority：任务优先级，请使用"1"。
    * @param taskid：任务ID，可传入32个字节的uuid，或传入空内容由SDK自动生成。
    * @param text：播放的文本内容。
    * @param async_listener：异步回调，设置nullptr采用同步方式调用。
    * @return：参见错误码。
    */
   -(int) nui_tts_play:(const char *)priority
         taskId:(const char *)taskid
           text:(const char *)text
    asyncCallback:(const NuiAsyncCallback *)listener;
```

- nui_tts_cancel：取消播放。

```
/**
    * 取消合成任务
    * @param taskid：传入想要停止的任务ID，如果为空则取消所有任务。
    * @param async_listener：异步回调，设置nullptr采用同步方式调用。
    * @return：参见错误码。
    */
   -(int) nui_tts_cancel:(const char *)taskid
       asyncCallback:(const NuiAsyncCallback *)listener;
```

- nui_tts_pause：暂停播放。

```
/**
    * 暂停
    * @param async_listener：异步回调，设置nullptr采用同步方式调用。
    * @return：参见错误码。
    */
   -(int) nui_tts_pause:(const NuiAsyncCallback *)listener;
```

- nui_tts_resume：恢复播放。

```
/**
    * 恢复暂停的任务
    * @param async_listener：异步回调，设置nullptr采用同步方式调用。
    * @return：参见错误码。
    */
   -(int) nui_tts_resume:(const NuiAsyncCallback *)listener;
```

- nui_tts_set_param：设置语音合成参数。

```
/**
    * 以键值对形式设置参数
    * @param param：参数名，参见接口说明。
    * @param value：参数值，参见接口说明。
    * @param async_listener：异步回调，设置nullptr采用同步方式调用。
    * @return：参见错误码。
    */
   -(int) nui_tts_set_param:(const char *)param
                  value:(const char *)value
           asyncCallback:(const NuiAsyncCallback *)listener;
```

- nui_tts_get_param：获取参数。

```
/**
    * 获取参数值
    * @param param：参数名，参见接口说明。
    * @param async_listener：异步回调，设置nullptr采用同步方式调用。
    * @return：参数值。
    */
   -(const char *) nui_tts_get_param:(const char *)param
                asyncCallback:(const NuiAsyncCallback *)listener;
```

- nui_tts_release：释放SDK资源。

```
/**
    * 释放SDK
    * @param async_listener：异步回调，设置nullptr采用同步方式调用。
    * @return：参见错误码。
    */
   -(int) nui_tts_release:(const NuiAsyncCallback *)async_listener;
```

调用步骤

1. 初始化SDK和播放组件。
2. 根据业务需要设置参数。
3. 调用nui_tts_play进行播放。
4. 在合成数据回调中，将数据写入播放器进行播放，建议使用流式播放。
5. 收到语音合成结束的回调。

代码示例

1. 语音合成初始化

```
NSString * initParam = [self genInitParams];
    //nui listener
    NuiTtsSdkListener ttsListener;
    ttsListener.tts_event_callback = ttsListenerOnEvent;
    ttsListener.tts_font_progress_callback = nullptr;
    ttsListener.tts_user_volume_callback = ttsUserProvideVolume;
    ttsListener.tts_user_data_callback = ttsListenerUserdata;
    ttsListener.user_data = nullptr;
    [_nui nui_tts_initialize:[initParam UTF8String] ttsListener:&ttsListener asyncCallback:nullptr log
Level:LOG_LEVEL_VERBOSE saveLog:true];
```

其中，genInitParams生成为String JSON字符串，包含资源目录和用户信息。其中用户信息包含如下字段。

```
[dictM setObject:id_string forKey:@"device_id"];
[dictM setObject:@"" forKey:@"url"];
[dictM setObject:@"" forKey:@"app_key"];
[dictM setObject:@"" forKey:@"token"];
```

2. 根据需求设置参数

```
//可以设置发音人、语调和语速等参数，参数列表参见接口说明。
[self.nui nui_tts_set_param:"font_name" value:"xiaoyun" asyncCallback:nullptr];
```

3. 启动语音合成

```
[self.nui nui_tts_play:"1" taskId:"" text:[content UTF8String] asyncCallback:nullptr];
```

4. 回调处理
   - onNuiTtsEventCallback：语音合成事件回调，根据语音合成状态控制播放器。

```
- (void)onNuiTtsEventCallback:(nuisdk::NuiSdkTtsEvent)event taskId:(char*)taskid code:(int)co
de {
    TLog(@"onNuiTtsEventCallback event[%d]", event);
    if (event == TTS_EVENT_START) {
        loop_in = TTS_EVENT_START;
        dispatch_async(dispatch_get_main_queue(), ^{
            [self->_voicePlayer play];
        });
    } else if (event == TTS_EVENT_END) {
        loop_in = TTS_EVENT_END;
        dispatch_async(dispatch_get_main_queue(), ^{
            [self->_voicePlayer drain];
        });
    }
}
```

   - onNuiTtsUserdataCallback：语音合成数据回调，将回调中的合成数据写入播放器进行播放。

```
- (void)onNuiTtsUserdataCallback:(char*)info wordIdx:(int)info_len buffer:(char*)buffer len:(in
t)len taskId:(char*)task_id {
    TLog(@"onnuiTtsUserdataCallback played info %s info_len %d", info, info_len);
    [_voicePlayer write:(char*)buffer Length:(unsigned int)len];
}
```

5. 结束语音合成

```
[self.nui nui_tts_cancel:nullptr asyncCallback:nullptr];
```