Notice

Nuance Cloud Services

HTTP Services for Nuance Cloud Services Clients

Copyright © 2011-2013 Nuance Communications, Inc. All rights reserved.

Published by Nuance Communications, Inc.

One Wayside Road, Burlington, Massachusetts 01803, U.S.A.

Last updated December 4, 2013.

Nuance Communications, Inc. provides this document without representation or warranty of any kind. The information in this document is subject to change without notice and does not represent a commitment by Nuance Communications, Inc. The software and/or databases described in this document are furnished under a license agreement and may be used or copied only in accordance with the terms of such license agreement. Without limiting the rights under copyright reserved herein, and except as permitted by such license agreement, no part of this document may be reproduced or transmitted in any form or by any means, including, without limitation, electronic, mechanical, photocopying, recording, or otherwise, or transferred to information storage and retrieval systems, without the prior written permission of Nuance Communications, Inc.

Nuance, the Nuance logo, Nuance Cloud Services, Nuance Voice Control, and Nuamnce Mobile Speech Platform are trademarks or registered trademarks of Nuance Communications, Inc. or its affiliates in the United States and/or other countries. All other trademarks referenced herein are the property of their respective owners.
# Table of Contents

1. **Nuance HTTP Interface basics** ........................................................................................................... 2  
   1.1 Building an HTTP application ........................................................................................................... 2

2. **Accessing the Automatic Speech Recognition (ASR) web service** ................................................. 2  
   2.1 Requirements ........................................................................................................................................ 2  
   2.1.1 Chunked transfer coding .................................................................................................................... 2  
   2.1.2 HTTP over TLS ................................................................................................................................. 3  
   2.1.3 Securing credentials ......................................................................................................................... 3  
   2.2 ASR web service .................................................................................................................................... 3  
   2.3 ASR HTTP request message ............................................................................................................... 4  
   2.3.1 Request query string ......................................................................................................................... 4  
   2.3.2 Request headers ................................................................................................................................ 5  
   2.3.3 Custom request headers (optional) ................................................................................................... 6  
   2.3.4 Request message body ..................................................................................................................... 7  
   2.4 ASR HTTP response message .............................................................................................................. 7  
   2.4.1 Response headers ........................................................................................................................... 7  
   2.4.2 Response message body ................................................................................................................ 8  
   2.4.3 Response HTTP status codes .......................................................................................................... 8

3. **Accessing the Text to Speech (TTS) web service** ............................................................................... 9  
   3.1 Requirements ....................................................................................................................................... 9  
   3.1.1 HTTP Over TLS .............................................................................................................................. 9  
   3.2 Text to Speech (TTS) ........................................................................................................................... 10  
   3.3 TTS HTTP request message ............................................................................................................... 11  
   3.3.1 Request query string ....................................................................................................................... 11  
   3.3.2 Request headers ................................................................................................................................ 12  
   3.3.3 Request message body .................................................................................................................... 13  
   3.4 TTS HTTP response message ............................................................................................................ 13  
   3.4.1 Response headers .......................................................................................................................... 13  
   3.4.2 Response message body ................................................................................................................ 13  
   3.4.3 Response HTTP status codes ......................................................................................................... 13

4. **Language support** .......................................................................................................................... 15  
   4.1 Languages and language codes ....................................................................................................... 15  
   4.2 Supported audio frequencies .......................................................................................................... 15

5. **Glossary and Definitions** ............................................................................................................... 16
1 Nuance HTTP Interface basics

This document describes how to use a HyperText Transfer Protocol (HTTP) client to access and request/receive speech services from Nuance® Cloud Services (NCS) web services. Available speech services are:

- Automatic Speech Recognition (ASR)
- Text to Speech (TTS) conversion

The HTTP interface allows you to add these services to your applications easily and quickly. This interface provides access to speech processing components hosted on a server using simple HTTP requests, minimizing overhead and resource consumption. The HTTP interface provides you with fast voice search, dictation, and high-quality, multilingual text-to-speech functionality in your application.

1.1 Building an HTTP application

When a client application uses HTTP-based speech services, it is expected to do the following:

- Handle audio recording
- Implement speech end-point detection (if desired)
- Handle audio playback
- Initiate the HTTP connection when necessary
- Construct and send the appropriate HTTP request
- Send or stream audio if necessary
- Receive the reply from the HTTP services
- Parse the reply
- Take action based on the reply

2 Accessing the Automatic Speech Recognition (ASR) web service

Use an HTTP client of your choice to access the ASR web service. Note that your HTTP client must be capable of sending requests that meet the requirements described below.

2.1 Requirements

Your HTTP request must meet the requirements concerning:

- Chunked transfer coding
- Secure HTTP
- Securing credentials

2.1.1 Chunked transfer coding

The ASR web service can support audio from various sources: audio files, audio from the device microphone, and streamed audio.
To minimize latency when handling streamed audio, the HTTP client must support HTTP 1.1 (RFC2616) and chunked transfer coding (www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.6.1).

Because the recorded audio stream length may not be known in advance, chunked encoding modifies the body of an HTTP message to transfer it as a series of chunks, each with its own size indicator.

For more information on chunked transfer coding, visit these links:
- four.livejournal.com/887211.html
- developers.sun.com/mobility/midp/questions/chunking/
- en.wikipedia.org/wiki/Chunked_transfer_encoding

### 2.1.2 HTTP over TLS

The HTTP client must support HTTP over TLS (Transport Layer Security), which is also known as HTTPS (Hypertext Transfer Protocol Secure). HTTPS provides encrypted communication and secure identification with the NCS ASR web service.

For more information on HTTP over TLS, visit these links:
- HTTP over TLS (www.ietf.org/rfc/rfc2818.txt)
- HTTP Secure (en.wikipedia.org/wiki/HTTP_Secure)

### 2.1.3 Securing credentials

It's very important to secure the credentials provided by Nuance—unique application ID (NMAID) and application key.

- Make sure that your application does not expose these credentials to end users.
- To secure the credentials and avoid theft, use a security tool (like amap).

### 2.2 ASR web service

The ASR web service receives an incoming voice stream in an HTTP POST from the HTTP client and provides the client with a transcription of what was spoken. Here’s an example POST and response:

**Request**

```plaintext
https://dictation.nuancemobility.net/NMDPAsrCmdServlet/dictation?appId=NMAID_FOO&appKey=525348e77144a9cee9a7471a8b67c50ea85b9e3eb377a3c2a3a23dc88f9150eefe76e6a339fdbc62b817595f53d72549d9ebe36438f8c2619846b963e9f43a93&id=57349abd2390 HTTP/1.1

Transfer-Encoding: chunked
Content-Type: audio/x-pcm;bit=16;rate=8000
Accept: text/plain
Accept-Language: en-US

... audio content ...
```

**Response**

```
HTTP/1.1 200 OK
```
Hello world

### 2.3 ASR HTTP request message

Use the information in this section to create your HTTP POST request to access the ASR web service.

#### 2.3.1 Request query string

A query string is the part of a Uniform Resource Locator (URL) that contains data to be passed to web applications. To access the ASR Web Service, you must set these three \(<\text{key}>\text{=}\text{<value>}\) pairs separated by ampersands (&):

- appId
- appKey
- id

For example: \text{appId=NMID\_FOO}&\text{appKey=323186E7}&\text{id=57349abd2390}

For more information on query strings, see these documents:


#### 2.3.1.1 appId and appKey

The unique application ID and application (customer) key are provided by Nuance and are used for authentication with the ASR Web Service. If the \text{appId} and \text{appKey} key-value pairs included with the HTTP POST request are missing or invalid, the ASR Web Service returns an HTTP response with status code \text{401 Unauthorized} (see \text{4xx} client error codes on page 8).

#### 2.3.1.2 id

A unique ID is required to identify the user or device. This parameter is used by the speaker-dependent acoustic model adaptation (SD-AMA) subsystem. SD-AMA creates adapted acoustic model profiles from audio collected from each user to improve recognition performance over time. Use caution regarding privacy regulations and laws when you assign an identifier to \text{id}. Assign \text{id} a randomly generated, persistent value that does not trace back to a specific person.

For example, do not use a telephone number, because it can potentially be traced back to an individual. The value of \text{id} must also be persistent across each use of the application by the same person or device. If it is not persistent, SD-AMA will not work correctly. If the same value is provided for all application users, then all users will share the same acoustic profile and recognition performance will be sub-optimal.

The value of \text{id} can contain any alphanumeric (0-9, a-z, A-Z) and underscore (_). This value cannot exceed 40 characters.
2.3.2 Request headers

Include these headers in the HTTP POST request for ASR services. All headers are mandatory except those identified as optional.

2.3.2.1 Content-Type

Specifies the audio format from the client incoming voice stream. The supported formats are:

<table>
<thead>
<tr>
<th>Header value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio/x-wav;codec=pcm;bit=16;rate=8000</td>
<td>8 kHz 16 bit PCM</td>
</tr>
<tr>
<td>audio/x-wav;codec=pcm;bit=16;rate=16000</td>
<td>16 kHz 16 bit PCM</td>
</tr>
<tr>
<td>audio/x-wav;codec=pcm;bit=16;rate=22000</td>
<td>22 kHz 16 bit PCM</td>
</tr>
<tr>
<td>audio/x-speex;rate=8000</td>
<td>Speex narrowband</td>
</tr>
<tr>
<td>audio/x-speex;rate=11025</td>
<td>Speex semi-wideband</td>
</tr>
<tr>
<td>audio/x-speex;rate=16000</td>
<td>Speex wideband</td>
</tr>
<tr>
<td>audio/amr</td>
<td>AMR</td>
</tr>
<tr>
<td>audio/qcelp</td>
<td>QCELP</td>
</tr>
<tr>
<td>audio/evrc</td>
<td>EVRC</td>
</tr>
</tbody>
</table>

Note 1: You can send only the raw audio without a header, except in the case of Speex, which is in the Ogg container format. If this header is missing, the default value is 8 kHz 16 bit PCM.

Note 2: Nuance’s acoustic data models are designed to perform best with audio from mobile devices. Audio from other sources, such as IVR, will not perform as well.

2.3.2.2 Accept

Specifies the ASR result types that are acceptable for the response. The supported formats are:

- text/plain
- application/xml

A text ASR result provides the top sentence and an n-best list of alternatives.

Note: XML will be supported in a future version. Currently, all results are returned as text. Do not specify application/xml, until the details of the XML format are made available. The XML format will support a list of alternatives in the ASR result.

If this header is missing or invalid, the default value is text/plain.

2.3.2.3 Accept-Language

Represents the language used to record the audio and restricts the natural language expected in the ASR result. For information on supported languages and the language codes that can be assigned to this header, see Language support on page 15. The full list of supported languages is available on the NDEV web site.

If this header is missing or the value assigned to it is invalid, the default value is enus (U.S. English).
2.3.2.4 Accept-Topic

Represents the type of topic used by the recognition engine.

The current supported values are: Dictation, WebSearch and DTV-Search (beta only). If this header is missing, the default value is Dictation. If the value assigned to the header is invalid, the status code 406 Not Acceptable is returned.

For example, you can specify the header as:

```
httppost.addHeader("Accept-Topic", "Dictation");
```

Dictation is typically used for longer phrases, like those used in SMS or email applications. WebSearch is typically used for shorter search-oriented phrases, like those often used in applications for searching products or music.

*Note:* The availability of topics depends on the end server deployment; not all topics are available on all deployments.

2.3.2.5 Transfer-Encoding

Unless the total length of the voice input stream is known, Transfer-Encoding: chunked is a mandatory header in the HTTP POST request.

2.3.2.6 Content-Length

Indicates the size, in bytes, of the voice input stream if it is known in advance. When Content-Length is specified, you cannot use the Transfer-Encoding header field.

2.3.2.7 X-Dictation-NBestListSize

Specifies the n-best list size (that is, the number of top-choice results) that should be returned to the client. Valid values are 1-10. The value defaults to 10 if a value outside the valid range is provided, if text is provided, or if no value is provided.

2.3.2.8 X-Dictation-EscapeNewLine

If set to "true", the X-Dictation-EscapeNewLine header replaces new lines within the text with an escape sequence: \\

Use this header if your application is designed for open dictation (for example, note taking) and there is a high probability the user will speak punctuation commands like "new line" or "new paragraph". This header helps to avoid confusion by distinguishing between new line characters separating n-best results and new lines that are part of the sentence within an n-best result.

2.3.3 Custom request headers (optional)

You can use custom request headers to send the "context" of the application’s text buffer within the same HTTP POST as the audio data. This information provides cross-utterance formatting and improves recognition accuracy.

Sending the context of a dictation or web search is optional. Note however, that if any one of X-Dictation-Buffer, X-Dictation-CursorStart, or X-Dictation-CursorEnd is included in the HTTP POST, you must also specify the other two. Otherwise, the dictation or web search may fail.

2.3.3.1 X-Dictation-Buffer

Provides the current content of the text buffer.

For example:

```
X-Dictation-Buffer: This is a test
```
2.3.3.2 X-Dictation-CursorStart
Indicates the cursor start position within the text specified in X-Dictation-Buffer.
For example:
X-Dictation-CursorStart: 4

2.3.3.3 X-Dictation-CursorEnd
Indicates the cursor end position within the text specified in X-Dictation-Buffer.
For example:
X-Dictation-CursorEnd: 4

2.3.3.4 X-Dictation-AudioSource
Indicates the source of the audio recording. The possible sources are:

- "SpeakerAndMicrophone"
- "HeadsetInOut"
- "HeadsetBT"
- "HeadPhone"
- "LineOut"

Properly specifying this header improves recognition accuracy. Nuance encourages you to pass this header whenever you can—and as accurately as possible.

2.3.4 Request message body
The message-body of the HTTP POST message carries the voice input stream associated with the request. The message-body differs from the entity-body only when Transfer-Encoding: chunked has been applied (see tools.ietf.org/html/rfc2616#section-3.6.1).

The presence of a message-body in a request is signaled by the inclusion of a Content-Length or Transfer-Encoding header field in the request message-headers.

Transfer-Encoding: chunked is the preferred way to send the voice input stream, which should be sent in chunks of \(~260\) ms of audio duration (that is, \(4160\) bytes of PCM 16 bits 8 kHz) when the user speaks.

When Transfer-Encoding: chunked is not used, the message body includes the complete recorded voice input stream with the Content-Length header.

2.4 ASR HTTP response message
The HTTP POST response message sent to the client contains status information about the completion of the request.

Note that confidence scores are not available.

2.4.1 Response headers
These headers are specified by the server for an ASR HTTP POST response message:

2.4.1.1 Content-Type
See Accept on page 5.
**Note:** Nuance supports only UTF-8 for the character set.

### 2.4.1.2 Content-Language
See [Accept-Language](#) on page 8.

### 2.4.1.3 x-nuance-sessionid
Used as a session ID from the server to identify the transaction with the ASR web service. Nuance requires the session ID if you report difficulties or failures in the application or service. Nuance uses `x-nuance-sessionid` to retrieve the call log associated with the ASR requests that experienced the problem. Once it retrieves the call logs, Nuance can investigate the reported problem.

### 2.4.2 Response message body
Carries the ASR result of what was spoken. The format is specified by the [Content-Type](#) header (page 5).

### 2.4.3 Response HTTP status codes
- 200 series—success
- 400 series—client error
- 500 series—server error

#### 2.4.3.1 2xx Success
Indicates that the action requested by the client was received, understood, accepted, and processed successfully.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 OK</td>
<td>Standard response for successful ASR HTTP POST request.</td>
</tr>
<tr>
<td>204 No Content</td>
<td>The server successfully processed the request but is not returning any content.</td>
</tr>
</tbody>
</table>

#### 2.4.3.2 4xx Client Error
Intended for cases in which the client seems to have erred.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 Bad Request</td>
<td>The request cannot be fulfilled due to bad syntax.</td>
</tr>
<tr>
<td>401 Unauthorized</td>
<td>Used when authentication is possible but has failed or not yet been provided.</td>
</tr>
<tr>
<td>403 Forbidden</td>
<td>The request was a legal request, but the server is refusing to respond to it.</td>
</tr>
<tr>
<td>404 Not Found</td>
<td>The ASR resource could not be found but may be available again in the future.</td>
</tr>
<tr>
<td>405 Method Not Allowed</td>
<td>A request was made for an ASR resource using a request method not supported; for example, using GET instead of POST.</td>
</tr>
<tr>
<td>406 Not Acceptable</td>
<td>The ASR resource is only capable of generating content not acceptable according to the Accept headers sent in the request.</td>
</tr>
</tbody>
</table>
### 408 Request Timeout
The server timed out waiting for the request.

### 410 Gone
The resource requested is no longer available and will not be available again.

### 413 Request Entity Too Large
The request is larger than the server is willing or able to process.

### 414 Request-URI Too Long
The URI provided was too long for the server to process.

### 415 Unsupported Media Type
The request entity has a media type that the server or resource does not support.

#### 2.4.3.3 5xx Server Error
Indicates that the server failed to fulfill an apparently valid request.

### 500 Internal Server Error
A generic error message, given when no more specific message is suitable.

### 501 Not Implemented
The server either does not recognize the request method, or it lacks the ability to fulfill the request.

### 503 Service Unavailable
The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary state.

### 504 Gateway Timeout
The server was acting as a proxy and did not receive a timely response from the upstream server.

### 505 HTTP Version Not Supported
The server does not support the HTTP protocol version used in the request.

### 3 Accessing the Text to Speech (TTS) web service
Use an HTTP client of your choice to access the Nuance Cloud Service web service to perform a Text to Speech (TTS) conversion. Note that your HTTP client must meet the requirements described below.

#### 3.1 Requirements
Your HTTP client must meet these requirements for a TTS web service.

##### 3.1.1 HTTP Over TLS
The HTTP client must support HTTP over TLS (Transport Layer Security), which is also known as HTTPS (Hypertext Transfer Protocol Secure). HTTPS provides encrypted communication and secure identification with the NCS TTS web service.

For more information on HTTP over TLS, visit these links:
- HTTP over TLS ([www.ietf.org/rfc/rfc2818.txt](http://www.ietf.org/rfc/rfc2818.txt))
3.2 Text to Speech (TTS)

The TTS web service receives a text message from an HTTP client and provides the client with an audio stream of the text sent. Here’s an example TTS HTTP POST request and the response from the server (note that the response content length is in bytes):

**Request:**
https://tts.nuancemobility.net:443/NMDesTCmdServlet/tts?appId=NMAID_FOO&appKey=25348e77144a9c3e9a7471a8b67c50ea85b9e3eb377a3c2a2a23dc88f9150eeef7e6a339fbd62b817595f53d72549d9ebe36438f8c2619846b963e9f43a9&id=57349abd2390&ttsLang=en_US
HTTP/1.1
Content-Type: text/plain
Accept: audio/x-wav
Hello world

**Response**
HTTP/1.1 200 OK
Date: Tue, 31 Aug 2010 22:50:35 GMT
x-nuance-sessionid: 97bd6505-b7d6-420a-8eb7-7583036f7a1

----------------------------------------
Response content length: 804
Chunked?: 0
----------------------------------------
... audio content ...
3.3 TTS HTTP request message

Use this information in this section to create your HTTP POST request to access the TTS web service.

3.3.1 Request query string

A query string is the part of a Uniform Resource Locator (URL) that contains data to be passed to web applications. To access the TTS Web Service, you must set the first three `<key>=<value>` pairs separated by ampersands (&).

- appId
- appKey
- id
- `{ttsLang}|voice`

Setting a key-value pair for either `ttsLang` or `voice` is optional. You cannot include both `ttsLang` and `voice` in the same query string. If you do not include a key-value pair for either `ttsLang` or `voice` in the query string, the default is `ttsLang=enus`:

Two examples of query strings to access the TTS web service:

- `appId=NMAID_FOO&appKey=323186E7&id=57349abd2390&ttsLang=enus`
- `appId=NMAID_FOO&appKey=323186E7&id=57349abd2390&voice=Samantha`

For more information on query strings, see these documents:

- Query strings: (en.wikipedia.org/wiki/Query_string)
- URI schemes: (en.wikipedia.org/wiki/URI_scheme)

3.3.1.1 appId and appKey

The unique application ID and application (customer) key are provided by Nuance and are used for authentication with the TTS Web Service. If the `appId` and `appKey` key-value pairs included with the HTTP POST request are missing or invalid, the TTS Web Service will return an HTTP response with status code 401 Unauthorized.

3.3.1.2 id

A unique ID is required to identify the user or device. Use caution regarding privacy regulations and laws when you assign an identifier to `id`. Assign `id` a randomly generated, persistent value that does not trace back to a specific person.

For example, do not use a telephone number, because it can potentially be traced back to an individual. The value of `id` must also be persistent across each use of the application by the same person or device, for reporting purposes.

The value of `id` can contain any alphanumeric (0-9, a-z, A-Z) and underscore (_). This value cannot exceed 40 characters.

3.3.1.3 ttsLang

This parameter defines the TTS language, and is optional. If you include the `ttsLang` parameter in the request query string, do not include the `voice` parameter; you will automatically get the default voice for the language that you specify.

If you specify a value for both `ttsLang` and for `voice`, the value assigned to `voice` takes precedence. For example, if you assign the value for U.S. English (`en_US`) to `ttsLang`, but specify the voice for U.S. Spanish, the result is that both the language and the voice will be U.S. Spanish.
For a list of supported languages and associated language codes that you can assign to the *ttsLang* parameter, see the NDEV website.

### 3.3.1.4 voice
The type (gender and personality) of voice used to generate the TTS audio response. For example, for North-American English there are a few voice types including:

- "Samantha"
- "Tom"

In addition to the voice type, the *voice* parameter identifies the language that’s used to generate the audio response. Therefore, if you include the *voice* parameter in the request query string, you must not include the *ttsLang* parameter.

For a list of currently supported voices (and their associated languages) that you can assign to the *voice* parameter, see the NDEV website.

### 3.3.2 Request headers
Include these headers in the HTTP POST request for TTS services. All headers are mandatory except those identified as optional.

#### 3.3.2.1 Content-Type
Specifies the type of text used for audio speech generation. The supported format is:

- text/plain—plain text

If this header is missing or invalid, the default value is *text/plain*.

#### 3.3.2.2 Accept
Can be used to specify the response audio format for the outgoing audio stream sent to the client. The supported formats are:

<table>
<thead>
<tr>
<th>Header value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio/x-wav;codec=pcm;bit=16;rate=8000</td>
<td>8 kHz 16 bit PCM</td>
</tr>
<tr>
<td>audio/x-wav;codec=pcm;bit=16;rate=16000</td>
<td>16 kHz 16 bit PCM</td>
</tr>
<tr>
<td>audio/x-wav;codec=pcm;bit=16;rate=22000</td>
<td>22 kHz 16 bit PCM</td>
</tr>
<tr>
<td>audio/x-speex;rate=8000</td>
<td>Speex narrowband</td>
</tr>
<tr>
<td>audio/x-speex;rate=11025</td>
<td>Speex semi-wideband</td>
</tr>
<tr>
<td>audio/x-speex;rate=16000</td>
<td>Speex wideband</td>
</tr>
<tr>
<td>audio/amr</td>
<td>AMR</td>
</tr>
<tr>
<td>audio/qcelp</td>
<td>QCELP</td>
</tr>
<tr>
<td>audio/evrc</td>
<td>EVRC</td>
</tr>
</tbody>
</table>

*Note:* Only the raw audio is returned, no header, except in the case of Speex, which is in the Ogg container format. If this header is missing, the default value is 8 kHz 16-bit PCM.
### 3.3.3 Request message body

The message-body of the HTTP POST message carries the text that is converted to speech, depending on the [Content-Type](#) (section 3.4.1.1).

### 3.4 TTS HTTP response message

The TTS response message is an HTTP POST that is sent to the client. The response message contains status information about the completion of the TTS request.

#### 3.4.1 Response headers

These headers specified by the server for a TTS HTTP POST response message:

**3.4.1.1 Content-Type**

See *Error! Reference source not found.* on page 12.

*Note:* Nuance supports only UTF-8 for the character set.

**3.4.1.2 x-nuance-sessionid**

Used as a session ID from the server to identify the transaction with the TTS web service. Nuance requires the session ID if you report difficulties or failures in the application or service. Nuance uses `x-nuance-sessionid` to retrieve the call log associated with the TTS requests that experienced the problem. After retrieving the call logs, Nuance can investigate the reported problem.

**3.4.1.3 Content-Length**

Indicates the size, in bytes, of the voice output stream.

**3.4.1.4 Chunked?**

Indicates if the TTS audio results are chunked.

#### 3.4.2 Response message body

Contains the TTS result (audio) of the text provided in the request.

#### 3.4.3 Response HTTP status codes

- 200 series—success
- 400 series—client error
- 500 series—server error

**3.4.3.1 2xx success codes**

Indicates that the action requested by the client was received, understood, accepted, and processed successfully.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 OK</td>
<td>Standard response for successful TTS HTTP POST request.</td>
</tr>
<tr>
<td>204 No Content</td>
<td>The server successfully processed the request, but is not returning any content.</td>
</tr>
</tbody>
</table>
### 3.4.3.2 4xx client error codes
Intended for cases in which the client seems to have erred.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 Bad Request</td>
<td>The request cannot be fulfilled due to bad syntax.</td>
</tr>
<tr>
<td>401 Unauthorized</td>
<td>Used when authentication is possible but has failed or not yet been provided.</td>
</tr>
<tr>
<td>403 Forbidden</td>
<td>The request was a legal request, but the server is refusing to respond to it.</td>
</tr>
<tr>
<td>404 Not Found</td>
<td>The TTS resource could not be found but may be available again in the future.</td>
</tr>
<tr>
<td>405 Method Not Allowed</td>
<td>A request was made of a TTS resource using a request method not supported; for example, using GET instead of POST.</td>
</tr>
<tr>
<td>406 Not Acceptable</td>
<td>The TTS resource is only capable of generating content not acceptable according to the Accept headers sent in the request.</td>
</tr>
<tr>
<td>408 Request Timeout</td>
<td>The server timed out waiting for the request.</td>
</tr>
<tr>
<td>410 Gone</td>
<td>Indicates that the resource requested is no longer available and will not be available again.</td>
</tr>
<tr>
<td>413 Request Entity Too Large</td>
<td>The request is larger than the server is willing or able to process.</td>
</tr>
<tr>
<td>414 Request-URI Too Long</td>
<td>The URI provided was too long for the server to process.</td>
</tr>
<tr>
<td>415 Unsupported Media Type</td>
<td>The request entity has a media type which the server or resource does not support.</td>
</tr>
</tbody>
</table>

### 3.4.3.3 5xx server error codes
Indicates that the server failed to fulfill an apparently valid request.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 Internal Server Error</td>
<td>A generic error message provided when no specific message is suitable.</td>
</tr>
<tr>
<td>501 Not Implemented</td>
<td>The server either does not recognize the request method, or it lacks the ability to fulfill the request.</td>
</tr>
<tr>
<td>503 Service Unavailable</td>
<td>The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary state.</td>
</tr>
<tr>
<td>Status Code</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>504 Gateway Timeout</td>
<td>The server was acting as a proxy and did not receive a timely response from the upstream server.</td>
</tr>
<tr>
<td>505 HTTP Version Not Supported</td>
<td>The server does not support the HTTP protocol version used in the request.</td>
</tr>
</tbody>
</table>

4 Language support

4.1 Languages and language codes

Information on the supported languages and language codes used by the ASR and TTS web services are available on the NDEV web site.

Use the codes to assign a value to these entities:

- **ASR web service**—The `Accept-Language` header in the HTTP POST request for ASR services (see [Accept-Language](#) on page 5)
- **TTS web service**—The `ttsLang` and `voice` parameters in the request query string to access text-to-speech (TTS) web services (see [Request query string](#) on page 11).

4.2 Supported audio frequencies

Note that 8 kHz and 16 kHz audio are supported for all languages.
5 Glossary and Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASR</td>
<td>automatic speech recognition</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>HTTP Over TLS (Transport Layer Security)</td>
</tr>
<tr>
<td>IMEI</td>
<td>International Mobile Equipment Identity</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organization</td>
</tr>
<tr>
<td>NCS</td>
<td>Nuance Cloud Services</td>
</tr>
<tr>
<td>NMAID</td>
<td>Nuance Mobile Application IDentifier</td>
</tr>
<tr>
<td>NVC</td>
<td>Nuance Voice Control</td>
</tr>
<tr>
<td>PCM</td>
<td>Pulse Code Modulation</td>
</tr>
<tr>
<td>SMS</td>
<td>Short Message Service (text messaging service)</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>TTS</td>
<td>Text to Speech</td>
</tr>
<tr>
<td>UUID</td>
<td>Universal(ly) Unique IDentifier</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Indicator</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UTF</td>
<td>Unicode Transformation Format</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>XML</td>
<td>eXtended Markup Language</td>
</tr>
</tbody>
</table>