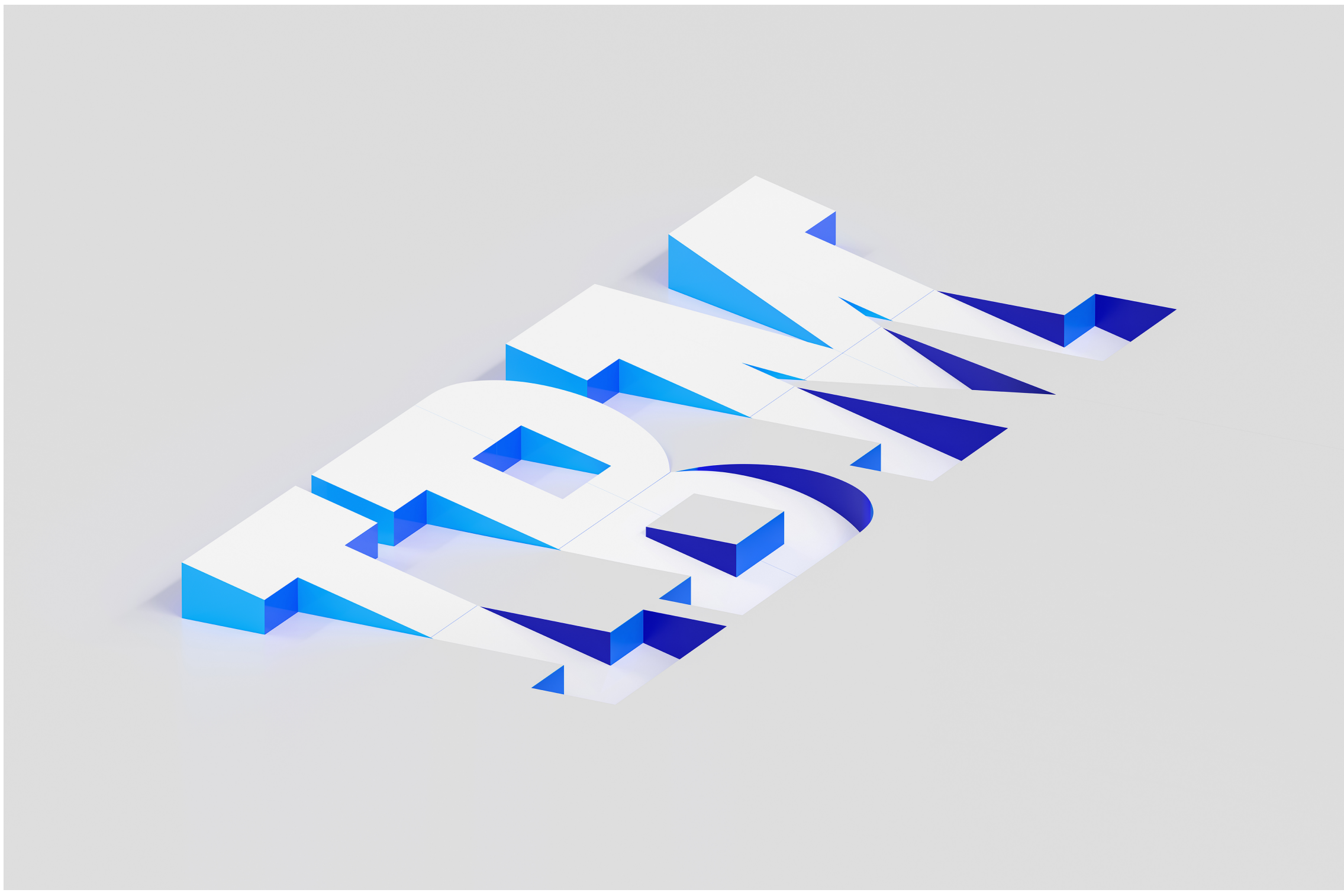
IBM Cloud

Text to Speech

Product guide



Edition notices

This PDF was created on 2026-05-19 as a supplement to *Text to Speech* in the IBM Cloud docs. It might not be a complete set of information or the latest version. For the latest information, see the IBM Cloud documentation at https://cloud.ibm.com/docs/text-to-speech.

Getting started with Text to Speech

The IBM Watson® Text to Speech service converts written text to natural-sounding speech to provide speech-synthesis capabilities for applications. This `curl`-based tutorial can help you get started quickly with the service. The examples show you how to call the service's `POST` and `GET /v1/synthesize` methods to request an audio stream.

Note: The tutorial uses the `curl` command-line utility to demonstrate REST API calls. For more information about `curl`, see Using curl with Watson examples.

IBM Cloud Watch the following video for a visual summary of getting started with the Text to Speech service.

View video: Getting started with the Text to Speech service

Before you begin

IBM Cloud

IBM Cloud

- Create an instance of the service:
  1. Go to the Text to Speech page in the IBM Cloud catalog.
  2. Sign up for a free IBM Cloud account or log in.
  3. Read and agree to the terms of the license agreement.
  4. Click **Create**.
- Copy the credentials to authenticate to your service instance:
  1. View the **Manage** page for the service instance:
     - If you are on the **Getting started** page for your service instance, click the **Manage** entry in the list of topics.
     - If you are on the **Resource list** page, expand the **AI / Machine Learning** grouping in the **Name** column, and click the name of your service instance.
  2. On the **Manage** page, click **Show Credentials** in the **Credentials** box.
  3. Copy the `API Key` and `URL` values for the service instance.

Tip: This tutorial uses an API key to authenticate. In production, use an IAM token. For more information see Authenticating to IBM Cloud.

IBM Cloud Pak for Data

IBM Cloud Pak for Data

The Text to Speech service must be installed and configured before beginning this tutorial. For more information, see Watson Speech services on Cloud Pak for Data.

1. Create an instance of the service by using the web client, the API, or the command-line interface. For more information about creating a service instance on IBM Cloud Pak for Data, see Creating a service instance for Watson Speech services.
2. Follow the instructions in *Creating a Watson Speech services instance* to obtain a Bearer token for the instance. This tutorial uses a Bearer token to authenticate to the service.

Step 1: Synthesize text in US English

The following command use the `POST /v1/synthesize` method to synthesize US English input to audio. The request uses the voice `en-US_MichaelV3Voice`. It produces audio in the WAV format.

Tip: You can use a browser or other tools to play the audio files that are produced by the examples in this tutorial. For more information, see Playing an audio file.

1. Issue the following command to synthesize the string "hello world". The request produces a WAV file that is named `hello_world.wav`.

   IBM Cloud

   - Replace `{apikey}` and `{url}` with your API key and URL.

   ```
   $ curl -X POST -u "apikey:{apikey}" \
   --header "Content-Type: application/json" \
   --header "Accept: audio/wav" \
   --data "{\"text\":\"hello world\"}" \
   --output hello_world.wav \
   "{url}/v1/synthesize?voice=en-US_MichaelV3Voice"
   ```

   IBM Cloud Pak for Data IBM Software Hub

   - Replace `{token}` and `{url}` with the access token and URL for your service instance.

   ```
   $ curl -X POST \
   --header "Authorization: Bearer {token}" \
   --header "Content-Type: application/json" \
   --header "Accept: audio/wav" \
   --data "{\"text\":\"hello world\"}" \
   --output hello_world.wav \
   "{url}/v1/synthesize?voice=en-US_MichaelV3Voice"
   ```

Step 2: Use a different voice and audio format

The following command again uses the `POST /v1/synthesize` method to synthesize the same US English input to audio. But this request uses the voice `en-US_AllisonV3Voice` and explicitly requests audio in the default Ogg format.

1. Issue the following command to synthesize the string "hello world" but with a different voice. The request produces an Ogg file that is named `hello_world.ogg`.

   IBM Cloud

   - Replace `{apikey}` and `{url}` with your API key and URL.

   ```
   $ curl -X POST -u "apikey:{apikey}" \
   --header "Content-Type: application/json" \
   --data "{\"text\":\"hello world\"}" \
   --output hello_world.ogg \
   "{url}/v1/synthesize?voice=en-US_AllisonV3Voice"
   ```

   IBM Cloud Pak for Data IBM Software Hub

   - Replace `{token}` and `{url}` with the access token and URL for your service instance.

   ```
   $ curl -X POST \
   --header "Authorization: Bearer {token}" \
   --header "Content-Type: application/json" \
   --header "Accept: audio/wav" \
   --data "{\"text\":\"hello world\"}" \
   --output hello_world.wav \
   "{url}/v1/synthesize?voice=en-US_AllisonV3Voice"
   ```

Step 3: Synthesize text in Spanish

The following command uses the `GET /v1/synthesize` method to synthesize Spanish input to an audio file. The `GET` method includes three query parameters: `accept` to specify the audio format, `text` to specify the input text for the audio, and `voice` to specify a Spanish voice. Because `accept` and `text` are passed as query parameters, the request is URL-encoded.

1. Issue the following command to synthesize the string "hola mundo" and produce a WAV file that is named `hola_mundo.wav`.

IBM Cloud

- Replace `{apikey}` and `{url}` with your API key and URL.

```
$ curl -X GET -u "apikey:{apikey}" \
--output hola_mundo.wav \
"{url}/v1/synthesize?accept=audio%2Fwav&text=hola%20mundo&voice=es-ES_EnriqueV3Voice"
```

IBM Cloud Pak for Data IBM Software Hub

- Replace `{token}` and `{url}` with the access token and URL for your service instance.

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--output hola_mundo.wav \
"{url}/v1/synthesize?accept=audio%2Fwav&text=hola%20mundo&voice=es-ES_EnriqueV3Voice"
```

Next steps

- To try an example application that accepts text and generates speech with different voices, see the Text to Speech demo.
- For more information about the service's interfaces and features, see Service features.
- For more information about all methods of the service's interfaces, see the API & SDK reference.

About Text to Speech

The IBM Watson® Text to Speech service provides APIs that use IBM's speech-synthesis capabilities to convert written text to natural-sounding speech. The service streams the synthesized audio back to the client with minimal delay. The audio uses appropriate cadence and intonation for its language and dialect to provide voices that are smooth and natural.

The service can be used in applications such as voice-automated chatbots, as well as a variety of voice-driven and screenless applications, such as tools for the disabled or visually impaired, video narration and voice over, and educational and home-automation solutions. It is appropriate for any application where audio is the preferred method of output.

Product versions

Text to Speech can be deployed as a managed cloud service or can be installed on premises. This documentation describes how to use both versions of the product. Information such as topics, paragraphs, and examples that applies exclusively to one version is clearly denoted:

- IBM Cloud for managed instances of Text to Speech that are hosted on IBM Cloud or for instances that are hosted on IBM Cloud Pak for Data as a Service. For information about all service updates, see the Release notes for Text to Speech for IBM Cloud.
- IBM Cloud Pak for Data IBM Software Hub for installed or on-premises instances of Text to Speech for IBM Cloud Pak for Data. For more information about installing and managing Watson Speech services, see Installation overview. For information about all service updates, see the Release notes for Text to Speech for IBM Cloud Pak for Data and Release notes for Text to Speech for IBM Software Hub.

Speech synthesis

The Text to Speech service supports both HTTP and WebSocket interfaces for speech synthesis. Both interfaces accept plain text and text that is marked up with the XML-based Speech Synthesis Markup Language (SSML). The WebSocket interface can also produce timing information about the words of the audio. For more information, see the following service features:

- Synthesizing speech with the service
- Using speech synthesis features

Customization

The service provides a customization interface that you can use to specify how the service pronounces unusual words that occur in your input text. You can define custom models to include dictionaries of words for your application's lexicon. For more information, see Customizing the service in the service features.

Language support

The service offers neural voices to synthesize text to speech in many languages and dialects:

- Dutch (Netherlands)
- English (Australian, United Kingdom, and United States dialects)
- French (Canadian and France dialects)
- German
- Italian
- Japanese
- Korean
- Portuguese (Brazilian)
- Spanish (Castilian, Latin American, and North American dialects)

For different languages, the service offers female voices, male voices, or both. For more information about the supported languages and voices, the types of voices that the service provides for each language, and their status for both versions of the service, see Languages and voices.

Audio support

The service produces audio in many popular formats:

- A-law
- Basic audio
- Free Lossless Audio Codec (FLAC)

- Linear 16-bit Pulse-Code Modulation (PCM)
- MP3 (or MPEG)
- Mu-law (or u-law)
- Ogg or Web Media (WebM) audio with the Opus or Vorbis codec
- Waveform Audio File Format (WAV)

Different formats support different sampling rates and other characteristics. For more information, see Using audio formats.

Beta features

IBM occasionally releases features and language support that are classified as beta. Such features are provided so that you can evaluate their functionality. They might be unstable and are subject to change or removal with short notice. They are not intended for use in a production environment.

Beta features might not provide the same level of performance or compatibility as generally available features. Generally available features are ready for use in a production environment.

Pricing

IBM Cloud

The service offers multiple pricing plans to suit your usage and application needs. For more information about the pricing plans or to purchase a plan, see the Text to Speech service in the IBM Cloud® Catalog.

Service features

You can access the speech synthesis capabilities of the IBM Watson® Text to Speech service via an HTTP or WebSocket interface. Both interfaces provide features that let you submit and receive different information from the service. And as with all Watson services, SDKs are available to simplify application development in many programming languages.

Using languages and voices

The service supports speech synthesis with voices for the languages listed in Language support. For different languages, the service offers female voices, male voices, or both. Some languages and voices might be supported for IBM Cloud® only.

All of the service's voices use neural voice technology, which produces more natural-sounding speech. The service offers three types of voices, *natural*, *expressive neural*, and *enhanced neural*, which have different qualities and features. For information about the types of voices and about the supported languages and voices for each type, see Languages and voices.

Using audio formats

The service can return synthesized audio in the many formats listed in Audio support. For information about the supported audio formats, see Using audio formats.

Synthesizing speech with the service

The Text to Speech service offers an HTTP Representational State Transfer (REST) interface and a WebSocket interface:

- The HTTP interface provides both `GET` and `POST` versions of the service's `/v1/synthesize` method. The two versions of the method offer generally equivalent functionality. You pass the text that is to be synthesized as a query parameter with the `GET` method and as the body of the request with the `POST` method.
- The WebSocket interface provides a `/v1/synthesize` method. You pass the text that is to be synthesized over an established WebSocket connection.

With both the HTTP and WebSocket interfaces, you specify the language and voice that are to be used, and the format for the audio that is to be returned.

- For an overview of the features that are available for speech synthesis, see Using speech synthesis features.
- For detailed descriptions and examples of the speech synthesis methods, see the API & SDK reference.

Data limits

The interfaces accept the following maximum amounts of text with a single request:

- The HTTP `GET /v1/synthesize` method accepts a maximum of 8 KB of input, which includes the input text and the URL and headers.
- The HTTP `POST /v1/synthesize` method accepts a maximum of 8 KB for the URL and headers, and a maximum of 5 KB for the input text that is sent in the body of the request.
- The WebSocket `/v1/synthesize` method accepts a maximum of 5 KB of input text.

These limits include all characters of the input, including whitespace.

IBM Cloud For billing purposes, whitespace characters are not counted. However, all other characters are counted, including those that are part of SSML elements.

Using speech synthesis features

The service supports additional features that you can use to tailor the text that you send and the audio that you receive.

SSML

You can pass the service plain text or text that is annotated with the Speech Synthesis Markup Language (SSML). SSML is an XML-based markup language that provides annotations of text for speech synthesis applications such as the Text to Speech service.

- For more information about specifying input text, see Specifying input text.
- For more information about using SSML, see Understanding SSML.

Modifying the speaking rate

To modify the global rate of speech synthesis for a request, you can use the `rate_percentage` query parameter. The speaking rate is the speed at which the

service speaks the text that it synthesizes into speech. A higher rate causes the text to be spoken more quickly; a lower rate causes the text to be spoken more slowly. The parameter changes the per-voice default rate for an entire request. For more information, see Modifying the speaking rate.

The `rate_percentage` parameter is beta functionality.

Modifying the speaking pitch

To modify the global pitch of speech synthesis for a request, you can use the `pitch_percentage` query parameter. The speaking pitch represents the tone of the speech that the service synthesizes. It represents how high or low the tone of the voice is perceived by the listener. A higher pitch results in speech that is spoken at a higher tone and is perceived as a higher voice; a lower pitch results in speech that is spoken in a lower tone and is perceived as a lower voice. The parameter changes the per-voice default pitch for an entire request. For more information, see Modifying the speaking pitch.

The `pitch_percentage` parameter is beta functionality.

Spelling out strings

To indicate how individual characters of a string (alphabetic, numeric, or alphanumeric) are to be spelled out, you can include the `spell_out_mode` query parameter with a request. By default, the service spells out individual characters at the same rate at which it synthesizes text for a language. You can use the parameter to direct the service to spell out individual characters more slowly, in groups of one, two, or three characters. Use the parameter with the SSML `<say-as>` element to control how the characters of a string are synthesized. For more information, see Specifying how strings are spelled out.

The `spell_out_mode` parameter is beta functionality that is supported only for German voices.

Word timings

With the WebSocket interface, you can obtain timing information about the location of words in the audio that the service returns. Timing information is useful for synchronizing the input text and the audio.

You can use the SSML `<mark>` element to identify specific locations, such as word boundaries, in the audio. For languages other than Japanese, you can also request word timing information for all words of the input text. For more information, see Generating word timings.

> **Note:** Word timings are not supported for Natural voices.

Using speech synthesis features with expressive neural voice

With expressive neural voices, the service supports additional features that modify how the text that you pass is synthesized into audio.

Using speaking styles

The expressive neural voices determine the sentiment of the text from the context of its words and phrases. The speech that they produce, in addition to having a very conversational style, reflects the mood of the text. You can embellish the voices' natural tendencies by indicating that all or some of the text is to emphasize a specific style: cheerful, empathetic, neutral, or uncertain. You use SSML to indicate the style and the text to which it is to be applied. For more information, see Using speaking styles.

Emphasizing interjections

When you use expressive neural voices, the service automatically detects a collection of common interjections based on context. When it synthesizes these interjections, it gives them the natural emphasis that a human would use in normal conversation. For some of the interjections, you can use SSML to enable or disable their emphasis. For more information, see Emphasizing interjections.

Emphasizing words

The expressive voices use a conversational style that naturally applies the correct intonation from context. But you can indicate that one or more words are to be given more or less emphasis. The change in stress can be indicated by an increase or decrease in pitch, timing, volume, or other acoustic attributes. For more information, see Emphasizing words.

Customizing the service

The service includes a customization interface that you can use to create custom models for use during speech synthesis. A custom model is a dictionary of words and their translations for a specific language. Each word/translation pair in a model tells the service how to pronounce a word when it occurs in input text.

You can use custom models to create application-specific translations for unusual words for which the service's regular pronunciation rules might yield imperfect pronunciations. For example, your application might routinely encounter domain-specific terms, special terms with foreign origins, personal or

geographic names, or abbreviations and acronyms. By using customization, you can define translations that tell the service how you want such terms to be pronounced.

You can define the custom entry for a word/translation pair based on other words, or you can create pronunciations based on phoneme symbols in the standard International Phonetic Alphabet (IPA) or the proprietary IBM Symbolic Phonetic Representation (SPR). Customization is available for all languages.

- For more information about customization, see Understanding customization.
- For more information about using phonetic IPA and SPR symbols, see Understanding phonetic symbols.

Note: IBM Cloud You must have the Standard or Premium pricing plan to use customization. Users of the Lite plan cannot use the customization interface. For more information about pricing plans, see the Text to Speech service in the IBM Cloud® Catalog.

Creating a custom voice

IBM Cloud

Premium customers can work with IBM to train a new custom voice for their specific application needs. A custom voice is a unique voice that is based on audio training data that the customer provides. IBM can train a custom voice with as little as one hour of training data.

To request a custom voice or for more information, complete and submit this IBM Request Form.

Using software development kits

SDKs are available for the Text to Speech service to simplify the development of speech applications. The SDKs support many popular programming languages and platforms.

- For a complete list of SDKs and links to the SDKs on GitHub, see Watson SDKs.
- For more information about all methods of the SDKs for the Text to Speech service, see the API & SDK reference.

Learning more about application development

For more information about working with Watson services and IBM Cloud:

- For an introduction, see Getting started with Watson and IBM Cloud.
- For information about using IBM Cloud Identity and Access Management, see Authenticating to Watson services.

Next steps

Explore the features introduced in this topic to gain a more in-depth understanding of the service's capabilities. Each feature includes links to topics that describe it in much greater detail.

- Using languages and voices and Using audio formats describe the basic underpinnings of the service's capabilities. You must choose a language and voice that are suitable for your text and application, and you must understand the characteristics of the audio the service returns.
- Synthesizing speech with the service provides links to detailed presentations of each of the service's interfaces. Experiment with the interfaces to determine which is best suited to your application needs.
- Using speech synthesis features briefly describes the features that are available for speech synthesis and provides links for more information. Use the features to tailor the text that you send and the audio that you receive.
- Using speech synthesis features with expressive neural voice introduces three additional features that are available for speech synthesis with expressive neural voices.
- Customizing the service describes the more advanced topic of customization, which you can use to create custom models that contain dictionaries of words and their translations for specific languages.
- Using software development kits provide links to the SDKs that are available to simplify application development in many programming languages.
- Learning more about application development provides links to help you get started with Watson services and understand authentication.

Data security

The IBM Watson® Text to Speech service provides the following security features to help you protect your user data.

Authenticating to IBM Cloud

IBM Cloud

You authenticate to the service by using IBM Cloud Identity and Access Management (IAM). You can pass either an API key or a bearer token in an `Authorization` header. For more information, see Authenticating to IBM Cloud.

Authenticating to IBM Cloud Pak for Data

IBM Cloud Pak for Data

You authenticate to the service by passing an access token with each request. You pass a bearer token in an `Authorization` header to authenticate. Several methods exist to generate the token, including by using an API key or by username. For more information, see Authenticating to IBM Cloud Pak for Data.

Text to Speech for IBM Cloud Pak for Data is a multi-tenant cloud solution. Your credentials provide access to your data only, and your data is isolated from other users.

Basic data security

The service provides security for all user data both in motion and at rest:

- Transport Layer Security (TLS) 1.2 is used to secure data in transit.
- Advanced Encryption Standard (AES)-256 with Secure Hash Algorithm (SHA)-256 is used to secure data at rest.

For more information about data security for cloud applications, see Security architecture for cloud applications.

Information security

The service supports the European Union General Data Protection Regulation (GDPR) to manage user data. You can pass the `X-Watson-Metadata` header with a request to associate a customer ID with data that the request passes to the service. If necessary, you can then delete the data by using the `DELETE /v1/user_data` method. For more information about these features and their use, see Information security.

IBM Cloud For Premium plans, the service also offers US Health Insurance Portability and Accountability Act (HIPAA) readiness in the Washington, DC, (`us-east`) and Dallas (`us-south`) regions.

Request logging

IBM Cloud

The service lets you control the default request logging that is performed for all Watson services. The service logs request and response data only to improve the service for future users. The logged data is never shared or made public. No data can be exported from the service. For example, users must retain the data that they use for training custom prompts because it cannot be retrieved from the service.

If you are concerned about the privacy of users' personal information or otherwise do not want your requests to be logged by IBM, you can opt out of the default logging to prevent the service from logging your request and response data. If you opt out, the service logs *no* user data from your HTTP or WebSocket requests. No user data is written to disk.

You can choose to opt out of logging at either the account level or the API request level. For more information, see Controlling request logging for Watson services.

Data separation and encryption

IBM Cloud

The service's Standard and Premium pricing plans offer different levels of data separation and encryption for users:

- Standard plans are multi-tenant solutions that provide logical separation of data by using common encryption keys.
- Premium plans are single-tenant solutions that provide physical separation of data. Premium plans provide dedicated data storage accounts that use unique encryption keys.

Network endpoints

IBM Cloud

IBM Cloud supports both public and private network endpoints with certain plans. Connections to private network endpoints do not require public internet access. Private network endpoints support routing services over the IBM Cloud private network instead of the public network. A private network endpoint provides a unique IP address that is accessible to you without a VPN connection.

Private network endpoints are supported only for paid plans. Check the plan information for your service to learn about the plans that support private network endpoints. For more information, see Public and private network endpoints.

Virtual private endpoints

IBM Cloud

IBM Cloud® Virtual Private Endpoints for Virtual Private Cloud (VPC) are available with certain plans. Virtual private endpoints enable you to connect to supported IBM Cloud services from your VPC network by using the IP addresses of your choosing, allocated from a subnet within your VPC. Virtual private endpoints are an evolution of the private connectivity to IBM Cloud services. They are virtual IP interfaces that are bound to an endpoint gateway created on a per service or service instance basis.

Virtual private endpoints are supported only for paid plans. Check the plan information for your service to learn about the plans that support virtual private endpoints. For more information, see Virtual Private Endpoints.

CORS support

The service supports Cross-Origin Resource Sharing (CORS). By using CORS, web pages can request resources directly from a foreign domain. CORS circumvents the same-origin security policy, which otherwise prevents such requests. Because the service supports CORS, a web page can communicate directly with the service without passing the request through the web server that hosts the page.

For instance, a web page that is loaded from a server in IBM Cloud can call the customization API directly, bypassing the IBM Cloud server. For more information, see enable-cors.org.

Signed URLs

Signed URLs provide authentication information as a query string. Although signed URLs provide access for a limited time and with limited permissions, they allow any user with such a URL to access the service, regardless of whether that user has an account. The Text to Speech service does not support signed URLs.

FIPS support

IBM Cloud Pak for Data IBM Software Hub

Text to Speech for IBM Cloud Pak for Data supports running on Federal Information Processing Standard (FIPS)-enabled clusters as of version 4.5.1.

Installing

Installation overview

IBM Cloud Pak for Data IBM Software Hub

Find information about how to install Watson Speech services.

Full installation instructions

- 5.1.0
- 5.0.0
- 4.8.x
- 4.7.x
- 4.6.x
- 4.5.x
- 4.0.x

Known issues

IBM Cloud Pak for Data IBM Software Hub

Known issues are listed by the release in which they were identified.

5.1.x releases

Limitations and known issues in Watson Speech services .

5.0.x releases

Limitations and known issues in Watson Speech services .

4.8.x releases

Limitations and known issues in Watson Speech services .

4.7.x releases

Limitations and known issues in Watson Speech services .

4.6.x releases

Limitations and known issues in Watson Speech services .

4.5.x releases

Limitations and known issues in Watson Speech services .

The following known issues apply to service functionality that spans releases for all platforms.

Issue: The Ogg audio format is not supported with the Safari browser

9 September 2022: By default, the service returns audio in the Ogg audio format with the Opus codec (`audio/ogg;codecs=opus`). However, the Ogg audio format is not supported with the Safari browser. If you are using the Text to Speech service with the Safari browser, you must specify a different format in which you want the service to return the audio.

- For more information about the available formats, see Supported audio formats.
- For more information about specifying a format, see Specifying an audio format.

Issue: Sampling rate for ogg format with opus codec is always 48 kHz

22 August 2019: When you specify the `audio/ogg;codecs=opus` audio format, you can optionally specify a sampling rate other than the default 48,000 Hz.

However, although the service accepts `48000`, `24000`, `16000`, `12000`, or `8000` as a valid sampling rate, it currently disregards a specified value and always returns the audio with a sampling rate of 48 kHz.

Release notes

Release notes for Text to Speech for IBM Cloud

IBM Cloud

The following features and changes were included for each release and update of managed instances of IBM Watson® Text to Speech that are hosted on IBM Cloud or for instances that are hosted on IBM Cloud Pak for Data as a Service. Unless otherwise noted, all changes are compatible with earlier releases and are automatically and transparently available to all new and existing applications.

For information about known limitations of the service, see Known limitations.

Note: For information about releases and updates of the service for IBM Cloud Pak for Data, see Release notes for Text to Speech for IBM Cloud Pak for Data.

26 March 2026

Deprecated v1 to v3 voice transition

Previously deprecated standard concatenating (v1) voices are now replaced by their neural (v3) equivalents during synthesis. When a request specifies one of these v1 voices, the corresponding v3 voice is used instead.

- `de-DE_BirgitVoice` -> `de-DE_BirgitV3Voice`
- `de-DE_DieterVoice` -> `de-DE_DieterV3Voice`
- `en-GB_KateVoice` -> `en-GB_KateV3Voice`
- `en-US_AllisonVoice` -> `en-US_AllisonV3Voice`
- `en-US_LisaVoice` -> `en-US_LisaV3Voice`
- `en-US_MichaelVoice` -> `en-US_MichaelV3Voice`
- `es-ES_EnriqueVoice` -> `es-ES_EnriqueV3Voice`
- `es-ES_LauraVoice` -> `es-ES_LauraV3Voice`
- `es-LA_SofiaVoice` -> `es-LA_SofiaV3Voice`
- `es-US_SofiaVoice` -> `es-US_SofiaV3Voice`
- `fr-FR_ReneeVoice` -> `fr-FR_ReneeV3Voice`
- `it-IT_FrancescaVoice` -> `it-IT_FrancescaV3Voice`
- `ja-JP_EmiVoice` -> `ja-JP_EmiV3Voice`
- `pt-BR_IsabelaVoice` -> `pt-BR_IsabelaV3Voice`

If you omit the optional `voice` parameter from a speech synthesis request, the service now uses `en-US_MichaelV3Voice` by default. This neural voice (v3) replaces the previous default standard concatenating (v1) voice, `en-US_MichaelVoice`.

11 March 2026

Important: Deprecation of the Beta feature Tune by Example

The Tune by Example beta feature is deprecated. The APIs for creating new prompts or speaker models are no longer supported.

- *Existing users:* Existing customizations that already contain prompts or speaker models will continue to function for speech synthesis for a limited period of time. However, new prompts or speaker models cannot be created.
- *New users:* The APIs for creating prompts or speaker models are disabled and cannot be used.

The Tune by Example feature will be fully removed from the service in a future release. Once removed, synthesis using customizations that rely on prompts or speaker models will no longer be supported.

13 January 2026

New English Natural Voices

The service now supports two new Natural voices - one male and one female, for AU English:

- `en-AU_JackNatural`
- `en-AU_HeidiNatural`

Natural Voices provide advanced performance in terms of naturalness and expressiveness. These voices use various techniques to provide an edge over Expressive voices. For more information, see

- Natural voices
- English (Australian) symbols

05 December 2025

New Latin American Spanish Natural Voices

The service now supports two new Natural voices - one male and one female, for Latin American Spanish:

- `es-LA_AlejandroNatural`
- `es-LA_DanielaNatural`

Natural Voices provide advanced performance in terms of naturalness and expressiveness. These voices use various techniques to provide an edge over Expressive voices. For more information, see

- Natural voices
- Spanish symbols

31 October 2025

New Brazilian Portuguese Natural Voices

The service now supports two new Natural voices - one male and one female, for Brazilian Portuguese:

- `pt-BR_LucasNatural`
- `pt-BR_CamilaNatural`

Natural Voices provide advanced performance in terms of naturalness and expressiveness. These voices use various techniques to provide an edge over Expressive voices. For more information, see

- Natural voices
- Portuguese (Brazilian) symbols

24 July 2025

New English Natural Voices

The service now supports four new Natural voices - two male and two female, for US English:

- `en-US_EmmaNatural`
- `en-US_EthanNatural`
- `en-US_JacksonNatural`
- `en-US_VictoriaNatural`

The service now supports two new Natural voices - male and female, for UK English:

- `en-GB_ChloeNatural`
- `en-GB_GeorgeNatural`

The service now supports a new female Natural voice for CA English:

- `en-CA_HannahNatural`

Natural Voices provide advanced performance in terms of naturalness and expressiveness. These voices use various techniques to provide an edge over Expressive voices. For more information, see

- Natural voices
- English (United States & Canada) symbols
- English (United Kingdom) symbols

09 July 2025

Deprecation of V1 voices

Effective **07 September 2025**, all V1 voices will be deprecated from the service. All deprecated v1 voices will automatically change to their corresponding v3 voices within the next 12 months.

19 May 2025

New generation of voices - Natural voices

The service now supports a new generation of voices called the Natural Voices, with a new voice for US English:

- `en-US_EllieNatural`

Natural Voices provide advanced performance in terms of naturalness and expressiveness. These voices use various techniques to provide an edge over Expressive voices. For more information, see Natural voices.

The service now supports additional optional `format` attribute for `<say-as interpret-as="letters" format="xx">` SSML tag

You can now specify the value `group` or `single` with the optional `format` attribute. These attributes help improve the legibility of alphanumeric strings like confirmation of numbers and ID by adding strategic silences.

17 February 2025

New Brazilian Portuguese male expressive neural voice

The service now supports a new male expressive neural voice for Brazilian Portuguese:

- `pt-BR_LucasExpressive`

Expressive neural voices offer natural-sounding speech that is clear, crisp, and fluid. The new voice is generally available (GA) for production use. It supports the use of both the standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols. For more information, see

- Expressive neural voices
- Portuguese (Brazilian) symbols
- Portuguese expressive voices

09 December 2024

New UK English male expressive neural voice

The service now supports a new male expressive neural voice for UK English:

- `en-GB_GeorgeExpressive`

Expressive neural voices offer natural-sounding speech that is clear, crisp, and fluid. The new voice is generally available (GA) for production use. It supports the use of both the standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols. For more information, see

- Expressive neural voices
- English (United Kingdom) symbols

15 May 2024

New Latin American Spanish female expressive neural voice

The service now supports a new female expressive neural voice for Latin American Spanish

- `es-LA_DanielaExpressive`

Expressive neural voices offer natural-sounding speech that is clear, crisp, and fluid. The new voice is generally available (GA) for production use. It supports the use of both the standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols. For more information, see

- Expressive neural voices.
- Spanish expressive voices.
- Spanish symbols.

16 Jan 2024

Text to Speech improvements for US English Expressive voices

When you use US English Expressive voices, the synthesis is faster with lower latency.

Defect fix: Inflection issue when using the question mark (?) in a short utterance

Defect fix: When some short utterances were ended with a question mark (?), the inflection was ignored. This issue is now fixed.

30 Nov 2023

Improved Text to Speech synthesis engine for version 3 voices

The Text to Speech engine is improved for delivering faster synthesis with lower latency for version 3 voices.

Improved Text to Speech voice en-AU_HeidiExpressive

The Text to Speech voice en-AU_HeidiExpressive is improved that can adjust the pitch and synthesis.

9 June 2023

Defect fix: TTS no longer fails due to the error message “[Errno 2] No such file or directory“

Defect fix: When using TTS with websockets, it no longer fails due to the error message “[Errno 2] No such file or directory“

18 May 2023

Defect fix: Added support for missing SSML features on the Dutch voice

> **Defect fix:** When using the Dutch voice, all supported SSML features now work as designed. Previously, when using the Dutch voice, some SSML features were not working.

Defect fix: Commas are now respected when using phoneme tags

> **Defect fix:** When using phoneme tags in SSML, commas (pauses) are now working as expected. Previously, when commas were preceding or following text with phoneme tags, the pause was ignored.

6 April 2023

Updates to beta Netherlands Dutch enhanced neural voice

> **Defect fix:** The beta Netherlands Dutch `nl-NL_MerelV3Voice` was updated for internal fixes and improvements. The limitations that are described for the initial release of the voice in the 31 March 2023 service update still apply.

31 March 2023

Important: End of service for all neural voices

> **Important:** All neural voices have reached their end of service date and have been removed from the service and the documentation. *No enhanced neural or expressive neural voices are affected.* For a complete list of all obsolete neural voices, see the 1 March 2023 service updates. An attempt to use an obsolete voice returns the HTTP response code 404 with the message `Model '{voice}' not found`.
>
> New enhanced neural and expressive neural voices are available for the Australian English, Korean, and Netherlands Dutch languages. *You must migrate to one of the new voices for Australian English, Korean, or Netherlands Dutch to continue by using the obsolete languages.*
>
> For more information, see Languages and voices.

22 March 2023

New beta Netherlands Dutch enhanced neural voice

> The service now supports a new enhanced neural female voice for Netherlands Dutch: `nl-NL_MerelV3Voice`. It supports the use of both the standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols.
>
> The new voice is beta functions pending completion of support for SSML. At its initial release, the voice does not support use of the following SSML-related functions:
>
> - The `<prosody>` element with any speech synthesis request
> - The `rate_percentage` and `pitch_percentage` parameters with any speech synthesis request
> - The `<mark>` element with a WebSocket speech synthesis request
> - The `timings` parameter of the JSON text message with a WebSocket speech synthesis request
>
> For more information about the new voice, its support for IPA and SPR symbols, and migrating to the new voice from the deprecated Netherlands Dutch neural voices, see
>
> - Enhanced neural voices
> - Dutch (Netherlands) symbols

Defect fix: Update Korean phonetic symbols in documentation

> **Defect fix:** In the documentation for Korean SPR symbols, two-character symbols for consonants are now enclosed in single quotation marks, making them a single symbol. Previously, they were shown as two separate symbols, without enclosing quotation marks. For more information, see Consonants (Korean).

Documentation updates for IBM SPR symbols

The overview documentation for IBM SPR symbols has been updated to clarify the use of multi-character symbols. For more information, see Speech sound symbols).

1 March 2023

Important: Pending end of service for all neural voices

Important: Effective **31 March 2023**, all neural voices reach their end of service date and will be removed from the service and the documentation. The neural voices have been deprecated since 31 March 2022. *No enhanced neural or expressive neural voices are effected.*

The following neural voices are removed:

- Arabic: `ar-MS_OmarVoice`
- Chinese (Mandarin): `zh-CN_LiNaVoice`, `zh-CN_WangWeiVoice`, and `zh-CN_ZhangJingVoice`
- Czech: `cs-CZ_AlenaVoice`
- Dutch (Belgian): `nl-BE_AdeleVoice` and `nl-BE_BramVoice`
- Dutch (Netherlands): `nl-NL_EmmaVoice` and `nl-NL_LiamVoice`
- English (Australian): `en-AU_CraigVoice`, `en-AU_MadisonVoice`, and `en-AU_SteveVoice`
- Korean: `ko-KR_HyunjunVoice`, `ko-KR_SiWooVoice`, `ko-KR_YoungmiVoice`, and `ko-KR_YunaVoice`
- Swedish: `sv-SE_IngridVoice`

New enhanced neural and expressive neural voices are already available for the Australian English and Korean languages. In the coming weeks, a new enhanced neural voice is available for the Netherlands Dutch language. *You must migrate to one of the new voices for Australian English, Korean, or Netherlands Dutch before 31 March 2023.*

For more information, see Languages and voices.

27 February 2023

New Australian English expressive neural voices

The service now supports two new expressive neural voices, male and female, for Australian English:

- `en-AU_HeidiExpressive`
- `en-AU_JackExpressive`

Expressive neural voices offer natural-sounding speech that is exceptionally clear, crisp, and fluid. The new voices are generally available (GA) for production use. They support the use of both the standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols. For more information, see

- Expressive neural voices
- English (Australian) symbols

You can migrate from Australian English neural voices that are deprecated to the new expressive neural voices.

New Korean enhanced neural voice

The service now supports a new enhanced neural female voice for Korean: `ko-KR_JinV3Voice`. The new voice is generally available (GA) for production use. It supports the use of both the standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols. For more information, see

- Enhanced neural voices
- Korean symbols

You can migrate from Korean neural voices that are deprecated to the new enhanced neural voice.

Defect fix: French Canadian voice now handles numeric times properly

Defect fix: The French and Canadian voices now pronounce times like `19:41` correctly. Previously, the voices were omitting elements of the time in the synthesized audio.

Defect fix: Japanese voice no longer inserts unexpected audio

Defect fix: The Japanese voice no longer inserts unexpected audio in speech synthesis results. Previously, other audio was inserted in certain cases.

20 January 2023

Cloud Foundry deprecation and migration to resource groups

IBM announced the deprecation of IBM Cloud Foundry on 31 May 2022. As of 30 November 2022, new IBM Cloud Foundry applications cannot be created and only existing users are able to deploy applications. IBM Cloud Foundry reaches end of support on 1 June 2023. Then, any IBM Cloud Foundry application runtime instances running IBM Cloud Foundry applications will be permanently disabled, deprovisioned, and deleted.

To continue to use your IBM Cloud applications beyond 1 June 2023, you must migrate to resource groups before that date. Resource groups are conceptually similar to Cloud Foundry spaces. They include several extra benefits, such as finer-grained access control by using IBM Cloud Identity and Access Management (IAM), the ability to connect service instances to apps and services across different regions, and an easy way to view usage per group.

Defect fix: Specifying large cardinal numbers with the `<say-as>` element no longer causes errors for English voices

Defect fix: You can now use the `<say-as>` element to pronounce large numbers as cardinal numbers. Previously, enclosing a large number in the `<say-as>` element with the attribute `interpret-as="cardinal"` might cause speech synthesis to fail for English voices. For example, `<say-as interpret-as="cardinal">3,200</say-as>` might cause the service to generate an error. For more information, see cardinal in the topic *SSML elements*.

Defect fix: Homonyms and other words are now pronounced correctly by English voices

Defect fix: The service now pronounces homonyms and other words correctly based on their context in English text that is to be synthesized. Previously, words such as `advocate` and `wifi` might be pronounced incorrectly by English voices.

30 November 2022

Defect fix: Add rules for custom model naming documentation

Defect fix: The documentation now provides detailed rules for naming custom models. For more information, see

- Creating a custom model
- API and SDK reference

7 October 2022

The service now enforces stricter SSML validation

The service now enforces stricter validation of input text that includes Speech Synthesis Markup Language (SSML) elements. Required elements of attributes must be specified with valid values. Otherwise, the request fails with a 400 error code. For more information about SSML validation and the requirements that marked-up text must meet, see SSML validation.

Defect fix: The gender that is listed for the `en-US_MichaelExpressive` voice is now correct

Defect fix: When you list information about the available voices, the `gender` of the `en-US_MichaelExpressive` voice is now `male`. Previously, the voice's gender was mistakenly described as `female`. For more information, see Listing information about voices.

23 September 2022

New US English expressive neural voices

The service offers four new expressive neural voices for US English:

- `en-US_AllisonExpressive`
- `en-US_EmmaExpressive`
- `en-US_LisaExpressive`
- `en-US_MichaelExpressive`

Expressive neural voices offer natural-sounding speech that is exceptionally clear, crisp, and fluid. The new voices are generally available (GA) for production use. They support the use of both the standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols. For more information, see

- Expressive neural voices
- English (United States) symbols

New speaking styles for expressive neural voices

The expressive neural voices determine the sentiment of the text from the context of its words and phrases. The speech that they produce, in addition to having a conversational style, reflects the mood of the text. But you can embellish the voices' natural tendencies by indicating that all or some of the text is to emphasize one of the following speaking styles:

- **Cheerful** - Expresses happiness and good news.
- **Empathetic** - Expresses empathy or sympathy.
- **Neutral** - Expresses objectivity and evenness.
- **Uncertain** - Expresses confusion or uncertainty.

For more information, see Using speaking styles.

New interjection emphasis with expressive neural voices

With expressive neural voices, the service automatically detects a set of common interjections based on context. When it synthesizes these interjections, it gives them the natural emphasis that a human would use in normal conversation. For some of the interjections, you can use SSML to enable or disable their emphasis. For more information, see Emphasizing interjections.

New word emphasis with expressive neural voices

The expressive voices use a conversational style that naturally applies the correct intonation from context. But you can indicate that one or more words are to be given more or less emphasis. The change in stress can be indicated by an increase or decrease in pitch, timing, volume, or other acoustic attributes. For more information, see Emphasizing words.

21 September 2022

New Activity Tracker event for GDPR deletion of user information

The service now returns an Activity Tracker event when you use the `DELETE /v1/user_data` method to delete all information about a user. The event is named `text-to-speech.gdpr-user-data.delete`. For more information, see Activity Tracker events.

Defect fix: Custom word translations now accept commas in all cases

Defect fix: Word translations added to custom models now accept commas in all cases. Previously, a comma in a translation might occasionally cause the translation to fail to generate valid audio when used for speech syntheses. This problem was identified in US English custom models.

Defect fix: French synthesis of dates is now consistent

Defect fix: French synthesis no longer includes the article "le" before dates of the form "the *ordinal* of *month*." Previously, the article was included only for the first day of the month for French (for example, "the first of September," "le premier septembre").

Known limitations with using the Ogg audio format with the Safari browser

By default, the service returns audio in the Ogg audio format with the Opus codec (`audio/ogg;codecs=opus`). However, the Ogg audio format is not supported by the Safari browser. If you are using the Text to Speech service with the Safari browser, you must specify a different format in which you want the service to return the audio.

- For more information about the available formats, see Supported audio formats.
- For more information about specifying a format, see Specifying an audio format.

31 August 2022

New beta `rate_percentage` query parameter for controlling the global speaking rate

The service offers a new `rate_percentage` query parameter to modify the speaking rate for a speech synthesis request. The speaking rate is the speed at which the service speaks the text that it synthesizes into speech. A higher rate causes the text to be spoken more quickly; a lower rate causes the text to be spoken more slowly. The parameter changes the per-voice default rate for an entire request. For more information, see Modifying the speaking rate.

New beta `pitch_percentage` query parameter for controlling the global speaking pitch

The service offers a new `pitch_percentage` query parameter to modify the speaking pitch for a synthesis request. The speaking pitch represents the tone of the speech that the service synthesizes. It represents how high or low the tone of the voice is perceived by the listener. A higher pitch results in speech that is spoken at a higher tone and is perceived as a higher voice; a lower pitch results in speech that is spoken in a lower tone and is perceived as a lower voice. The parameter changes the per-voice default pitch for an entire request. For more information, see Modifying the speaking pitch.

Defect fix: Japanese synthesis is improved to handle long strings of input text

Defect fix: The service now correctly synthesizes Japanese requests that include long strings of characters. Previously, the service failed to properly synthesize long strings of Japanese text.

Documentation updates for the SSML `<prosody>` element

The documentation for the SSML `<prosody>` element and its `pitch` and `rate` parameters has been improved and clarified. It also now includes a description of the differences between the service and the latest version of the SSML specification. For more information, see The `<prosody>` element.

3 August 2022

The service does not support multilingual speech synthesis

The service does not support multilingual speech synthesis currently. However, you can use customization to approximate the pronunciation of words from other languages. For more information, see Multilingual speech synthesis.

27 July 2022

New beta `spell_out_mode` parameter for German voices

To indicate how individual characters of a string are to be spelled out, you can now include the beta `spell_out_mode` query parameter with a synthesis request for a German voice. By default, the service spells out individual characters at the same rate at which it synthesizes text for a language. You can use the parameter to direct the service to spell out individual characters more slowly, in groups of one, two, or three characters. Use the parameter with the SSML `<say-as>` element to control how the characters of a string are synthesized. For more information, see Specifying how strings are spelled out.

25 May 2022

New support for `audio/alaw` audio format

The list of supported audio formats now includes `audio/alaw;rate={rate}`. Like `audio/basic` and `audio/mulaw`, this format provides single-channel audio that is encoded by using 8-bit u-law (or mu-law) data that is sampled at 8 kHz. For more information, see Using audio formats.

19 May 2022

Defect fix: Multiple consecutive SSML `<phoneme>` tags are now parsed correctly

Defect fix: The service now correctly synthesizes text that contains consecutive `<phoneme>` tags. Previously, if the text contained two or more

consecutive `<phoneme>` tags, the service synthesized only the first tag, ignoring the others.

31 March 2022

Important: Deprecation of all neural voices

Important: Effective **31 March 2022**, all neural voices are deprecated. The deprecated voices remain available to existing users until **31 March 2023**, when they will be removed from the service and the documentation. *No enhanced neural voices or expressive voices are deprecated.*

The following neural voices are now deprecated:

- Arabic: `ar-MS_OmarVoice`
- Chinese (Mandarin): `zh-CN_LiNaVoice`, `zh-CN_WangWeiVoice`, and `zh-CN_ZhangJingVoice`
- Czech: `cs-CZ_AlenaVoice`
- Dutch (Belgian): `nl-BE_AdeleVoice` and `nl-BE_BramVoice`
- Dutch (Netherlands): `nl-NL_EmmaVoice` and `nl-NL_LiamVoice`
- English (Australian): `en-AU_CraigVoice`, `en-AU_MadisonVoice`, and `en-AU_SteveVoice`
- Korean: `ko-KR_HyunjunVoice`, `ko-KR_SiWooVoice`, `ko-KR_YoungmiVoice`, and `ko-KR_YunaVoice`
- Swedish: `sv-SE_IngridVoice`

The deprecated neural voices continue to be available to existing users but are no longer available to new users:

- *Existing users:* Instances of the service created before *31 March 2022* can continue to use the deprecated voices for speech synthesis. The instances can also be used to create and work with custom models that are based on the deprecated voices. And they can continue to list and query the voices with the `GET /v1/voices` and `GET /v1/voices/{voice}` methods.
- *New users:* Instances of the service that is created on or after *31 March 2022* cannot use the deprecated voices for speech synthesis. The instances also cannot be used to create custom models that are based on the deprecated voices. And they cannot list or query the voices with the `GET /v1/voices` and `GET /v1/voices/{voice}` methods. Any API call that includes one of the deprecated voices returns an HTTP error code of 400 or 404, depending on the call.

New voices for Australian English, Dutch, and Korean will be released by 15 February 2023. If you are using Australian English, Dutch, or Korean voices, your API calls will automatically redirect to the new voices in that language. Voices in Arabic, Chinese, Czech, Swedish, and Flemish will be removed from service.

For more information about all available languages and voices, see Languages and voices.

Deprecated standard voices are removed from the documentation

The standard concatenating voices were deprecated on 2 December 2020. These standard voices have now been removed from the API reference. The topic *Migrating from standard to neural voices* has also been removed from the page Languages and voices.

Similarly, the former name of the Arabic voice, `ar-AR_OmarVoice`, was deprecated at the same time. It too has been removed from the documentation. Use the voice `ar-MS_OmarVoice` instead.

28 February 2022

Change to word timing response for WebSocket interface

The response object that the service sends when you request word timings with the WebSocket interface have changed. The service now sends word timing results in a single array that includes a string followed by two floats:

```
{
  "words": [
    ["Hello", 0.0, 0.259],
    ["world", 0.259, 0.532]
  ]
}
```

The service previously sent timing results as an array that included a string following by an array of two floats:

```
{
  "words": [
    ["Hello", [0.0629826778195474, 0.2590192737303819]],
    ["world", [0.2598829173456253, 0.5322130804452672]]
  ]
}
```

Also, the level of precision for word timings and marks is now reduced to three decimal places. For more information about the new responses, see Generating word timings.

Note: Results for enhanced neural and neural voices were different previously. These inconsistencies might cause errors for the Watson SDKs. The results for all voices are now consistent.

26 January 2022

Defect fix: Improve SSML documentation

Defect fix: The SSML documentation was updated to correct the following errors:

- The examples of the `<break>` element are now correct. The element is unary, as now shown in the examples. The previous examples included open and close tags with embedded text. The embedded text was not spoken by the service. For more information, see The `<break>` element.
- The service supports SSML version 1.1. All references and examples now use the correct version. The documentation previously referred to version 1.0.

3 December 2021

New Czech neural voice: `cs-CZ_AlenaVoice`

A new language, Czech, with a new female voice, `cs-CZ_AlenaVoice`, is now available. The voice is a neural voice.

New Belgian Dutch neural voice: `nl-BE_BramVoice`

A new male Belgian Dutch (Flemish) voice, `nl-BE_BramVoice`, is now available. The voice is a neural voice.

Defect fix: Improve SSML and speech synthesis

Defect fix: The following defects for the Speech Synthesis Markup Language (SSML) and speech synthesis were fixed with this release:

- The `pitch` attribute of the `<prosody>` element is now applied to all specified text. Previously, the pitch change was not always applied to the first word of the affected text. Also, the documentation now includes more guidance about specifying a `pitch` value. For more information, see The `pitch` attribute.
- Speech synthesis of Japanese text now speaks the audio more slowly. Previously, the synthesized speech was being spoken too quickly. If you find that synthesis of Japanese text is still spoken too quickly for your application, use the `rate` attribute of the SSML `<prosody>` element to control the rate of speech. For more information, see The `rate` attribute.
- Neural voices now parse the escaped apostrophe character (`&apos;`) properly. Previously, some neural voices were not interpreting the character properly.

22 October 2021

Multiple neural voice improvements

The existing neural voices for Chinese, Dutch (Belgian and Netherlands), Australian English, and Korean have been updated for improved speech synthesis and enhanced audio results. For more information about all available voices, see Languages and voices.

New Australian English neural voice: `en-AU_SteveVoice`

A new male Australian English voice, `en-AU_SteveVoice`, is now available. The voice is a neural voice.

New Swedish neural voice: `sv-SE_IngridVoice`

A new language, Swedish, with a new female voice, `sv-SE_IngridVoice`, is now available. The voice is a neural voice.

6 October 2021

New US HIPAA support for Premium plans in Dallas location

US Health Insurance Portability and Accountability Act (HIPAA) support is now available for Premium plans that are hosted in the Dallas (`us-south`) location. For more information, see Health Insurance Portability and Accountability Act (HIPAA) .

Defect fix: Improve Latin American Spanish enhanced neural voice

Defect fix: For the Latin American Spanish voice (`es-LA_SofiaV3Voice`), questions of all types now use the correct intonation.

16 September 2021

Defect fix: Improve Castilian Spanish and North American Spanish enhanced neural voices

Defect fix: For the Castilian Spanish (`es-ES_EnriqueV3Voice` and `es-ES_LauraV3Voice`) and North American Spanish (`es-US_SofiaV3Voice`) voices, questions of all types now use the correct intonation. For the Latin American Spanish voice (`es-LA_SofiaV3Voice`), some questions do not use the correct intonation and sound instead like statements. The Latin American Spanish voice is fixed soon.

16 July 2021

New Belgian Dutch neural voice: `nl-BE_AdeleVoice`

The service now supports Belgian Dutch (Flemish) with the neural voice `nl-BE_AdeleVoice`. The Belgian Dutch voice supports customization and is generally available (GA) for production use.

12 April 2021

New Canadian French enhanced neural voice: `fr-CA_LouiseV3Voice`

The service now supports Canadian French with the enhanced neural voice `fr-CA_LouiseV3Voice`. The Canadian French voice supports customization and is generally available (GA) for production use.

- To hear a sample of the new voice, see Supported languages and voices .
- For more information about the phonetic symbols and Unicode values that are available for the Canadian French language, see French (Canadian) symbols.

New Tune by Example feature

The new Tune by Example feature lets you control how specified text is spoken by the service. The feature is beta functions that is supported only for US English custom models and voices. It has two components:

- *Custom prompts* include the written text that is to be spoken and recorded audio that speaks the text as you want to hear it. The audio specifies the intonation, cadence, and stress of the synthesized text. The prompt can emphasize different syllables or words, introduce pauses, and generally make the synthesized audio sound more natural and appropriate for its context.
- *Speaker models* provide enrollment audio for a user who speaks one or more prompts. A speaker model provides an audio sample of a user's voice. The service trains itself on the voice, which can help it to produce higher-quality prompts for that speaker.

You specify a custom prompt with a speech synthesis request to indicate how the service's voice is to pronounce the text. To specify a prompt, you use the SSML extension `<ibm:prompt id="{prompt_id}"/>`. The synthesized audio duplicates the prosody of the prompt.

New Tune be Example methods

The service includes eight new methods for working with the Tune by Example feature. The descriptions of the new methods that follow provide links

to their entries in the API and SDK reference. You might need to select the `Curl` tab of the reference to see the new methods.

- The service includes four methods for working with custom prompts:
  - List custom prompts: `GET /v1/customizations/{customization_id}/prompts`
  - Get a custom prompt: `GET /v1/customizations/{customization_id}/prompts/{prompt_id}`
  - Delete a custom prompt: `DELETE /v1/customizations/{customization_id}/prompts/{prompt_id}`
- The service includes four methods for working with speaker models:
  - List speaker models: `GET /v1/speakers`
  - Get a speaker model: `GET /v1/speakers/{speaker_id}`
  - Delete a speaker model: `DELETE /v1/speakers/{speaker_id}`

Updates to Activity Tracker actions for customization

The names of the actions for the Activity Tracker events for the customization methods have changed. The actions now include the string `custom-model` instead of `custom-voice`. The old names of the actions are deprecated. The old names are still available for use but will be removed at a future date. Migrate to the new names that are listed in Customization events at your earliest convenience.

Create events *Deprecated action name -> New action name*

- `text-to-speech.custom-voice.create` -> `text-to-speech.custom-model.create`
- `text-to-speech.custom-voice-word-list.create` -> `text-to-speech.custom-model-word-list.create`
- `text-to-speech.custom-voice-word.create` -> `text-to-speech.custom-model-word.create`

Read events *Deprecated action name -> New action name*

- `text-to-speech.custom-voice-list.read` -> `text-to-speech.custom-model-list.read`
- `text-to-speech.custom-voice.read` -> `text-to-speech.custom-model.read`
- `text-to-speech.custom-voice-word-list.read` -> `text-to-speech.custom-model-word-list.read`
- `text-to-speech.custom-voice-word.read` -> `text-to-speech.custom-model-word.read`

Update event *Deprecated action name -> New action name*

- `text-to-speech.custom-voice.update` -> `text-to-speech.custom-model.update`

Delete events *Deprecated action name -> New action name*

- `text-to-speech.custom-voice.delete` -> `text-to-speech.custom-model.delete`
- `text-to-speech.custom-voice-word.delete` -> `text-to-speech.custom-model-word.delete`

2 December 2020

Multiple voice improvements

The voices that are offered by the service have undergone significant change. The service supports new languages and voices, has improved the quality of many voices, and has deprecated many older voices. In addition, all of the services voices are now customizable and generally available (GA) for production use.

New neural and enhanced neural voices

To optimize the overall quality of voice synthesis, all available voices are now based on neural technology. The service offers two types of voices that are based on neural technology:

- *Neural voices*, which do *not* include the string `V3` in their names, are now available for Arabic, Australian English, Chinese, Netherlands Dutch, and Korean. Neural voices support the use of the International Phonetic Alphabet (IPA) with the Speech Synthesis Markup Language (SSML) `<phoneme>` element.
- *Enhanced neural voices*, which include the string `V3` in their names, are now available for Brazilian Portuguese, United Kingdom and United States English, French, German, Italian, Japanese, and Spanish (all dialects). Enhanced neural voices support the use of both IPA and IBM Symbolic Phonetic Representation (SPR) with the SSML `<phoneme>` element. Enhanced neural voices also achieve a slightly higher degree of natural-sounding speech. And further customization capabilities are exposed for enhanced neural voices in the coming months.

The service no longer offers standard voices for any language. Standard voices used concatenating synthesis to assemble segments of recorded

speech to generate the requested audio.

For more information, see Languages and voices.

New Australian English and Korean neural voices

The following Australian English and Korean voices are new:

- The service now supports Australian English with the following two neural voices: `en-AU_CraigVoice` and `en-AU_MadisonVoice`.
- The service now supports two new Korean neural voices: `ko-KR_HyunjunVoice` and `ko-KR_SiWooVoice`.

Improved neural voices

The voices for the existing Arabic, Chinese, Netherlands Dutch, and Korean languages, all of which were concatenating, are now neural:

- `ar-MS_OmarVoice`
- `ko-KR_YoungmiVoice`
- `ko-KR_YunaVoice`
- `nl-NL_EmmaVoice`
- `nl-NL_LiamVoice`
- `zh-CN_LiNaVoice`
- `zh-CN_WangWeiVoice`
- `zh-CN_ZhangJingVoice`

The following other changes have also been made:

- The Arabic voice is now named `ar-MS_OmarVoice`. The former name, `ar-AR_OmarVoice`, is deprecated. It will continue to function for at least one year but might be removed at a future date. You are encouraged to migrate to the new name at your earliest convenience.
- The Arabic language now supports the use of IPA symbols and Unicode values with the SSML `<phoneme>` element. To create a custom model for Arabic, you must use the language identifier `ar-MS`. The identifier `ar-AR` is not supported for customization.
- The IPA symbols for the Arabic language are new. The previously documented symbols have been replaced.
- The IPA symbols for the Netherlands Dutch language have been changed as follows:
  - Netherlands Dutch no longer supports the following IPA symbols: `tʲ` (`0074+02B2`), `ɲ` (`0272`), `ʦ` (`02A6`), and `ʔ` (`0294`).
  - Netherlands Dutch now supports the following IPA symbol: `ɣ` (`0263`).

Deprecated standard voices

The following standard concatenating voices are now deprecated:

- `de-DE_BirgitVoice`
- `de-DE_DieterVoice`
- `en-GB_KateVoice`
- `en-US_AllisonVoice`
- `en-US_LisaVoice`
- `en-US_MichaelVoice`
- `es-ES_EnriqueVoice`
- `es-ES_LauraVoice`
- `es-LA_SofiaVoice`
- `es-US_SofiaVoice`
- `fr-FR_ReneeVoice`
- `it-IT_FrancescaVoice`
- `ja-JP_EmiVoice`
- `pt-BR_IsabelaVoice`

All of the standard voices that have been deprecated have equivalent neural counterparts, so no voice is being taken away. Rather, you can change to the equivalent neural version of the voice (for example, from `de-DE_BirgitVoice` to `de-DE_BirgitV3Voice`) for better speech-synthesis results.

These deprecated voices are removed from the published documentation. They will continue to function for at least one year but might be removed at a future date. You are encouraged to migrate to the equivalent neural voices at your earliest convenience.

If you omit the optional `voice` parameter from a speech synthesis request, the service uses `en-US_MichaelV3Voice` by default. This neural voice replaces the now-deprecated `en-US_MichaelVoice` standard voice that was the previous default.

Deprecated features

The following features were available only for standard concatenating voices. They are deprecated and have been removed from the published documentation. They will continue to function for at least one year but might be removed at a future date. You are encouraged to remove these features from your applications and to migrate to neural voices at your earliest convenience.

- *Expressive SSML.* This was an IBM extension to SSML that was supported only for the `en-US_AllisonVoice` voice.
- *Voice transformation SSML.* This was an IBM extension to SSML that was supported only for the `en-US_AllisonVoice`, `en-US_LisaVoice`, and `en-US_MichaelVoice` voices.
- *The `volume` attribute of the `<prosody>` element.* This attribute was available for all standard voices.

Including these SSML elements with a synthesis request for a neural voice generates an HTTP 400 response code because the request fails SSML validation. For more information about SSML validation, see SSML validation.

New support for Cross-Origin Resource Sharing

Cross-Origin Resource Sharing (CORS) support is now available for all voices from the Google Chrome™ and Apple® Safari browsers. It is *not* available from the Mozilla Firefox™ browser for voices in the following languages: Arabic, Australian English, Chinese, Netherlands Dutch, and Korean. For more information, see using CORS support.

10 September 2020

Defect fix: Improve Japanese voice

Defect fix: For the `ja-JP_EmiV3Voice` voice, the service now correctly parses SSML input text that includes a prosody rate specification. Previously, the following use of the `<prosody>` element worked properly:

```
<speak>成功する/繁栄する</speak>
```

But the following use of the `rate` attribute with the `<prosody>` element caused the service to read and speak the embedded SSML notation:

```
<speak>
 <prosody rate="fast">成功する/繁栄する</prosody>
</speak>
```

The service now correctly parses and applies the `rate` attribute of the `<prosody>` element for Japanese input.

4 September 2020

A customization interface is now generally available

The customization interface is now generally available (GA). Customization is no longer beta functions. You can use the customization interface to specify how the service pronounces unusual words that occur in your input text by creating language-specific custom dictionaries. It can be used with all generally available and beta voices. For more information, see Understanding customization.

24 June 2020

New UK English and French neural voices

The service now offers two new neural voices:

- UK English: `en-GB_CharlotteV3Voice`
- French: `fr-FR_NicolasV3Voice`

It also offers an improved version of the existing UK neural voice, `en-GB_KateV3Voice`. For more information about all available voices, see Languages and voices.

Support for SSML `digits` attribute of `<say-as>` element for Japanese

The service now supports the `digits` attribute of the SSML `<say-as>` element with its Japanese voice. For more information, see The say-as element.

1 April 2020

New Korean standard voices: `ko-KR_YoungmiVoice` and `ko-KR_YunaVoice`

The service now supports two standard female Korean voices: `ko-KR_YoungmiVoice` and `ko-KR_YunaVoice`. The following information applies to both Korean voices:

- The voices are beta functions. They might not be ready for production use and are subject to change. They are initial offerings that are expected to improve in quality with time and usage.
- The voices support customization and the `/v1/pronunciation` method.
- The voices support the `<mark>` element and the `timings` parameter that are available with the WebSocket interface.
- The voices support all Speech Synthesis Markup Language (SSML) elements except for expressive SSML and voice transformation SSML.
- The voices support only the International Phonetic Alphabet (IPA), not IBM Symbolic Phonetic Representation (SPR), with the `<phoneme>` element.

New feature support for Arabic, Chinese, and Netherlands Dutch voices

The beta Arabic, Chinese, and Netherlands Dutch voices now support the following features:

- Customization and the `/v1/pronunciation` method.
- The `<mark>` element and the `timings` parameter that are available with the WebSocket interface.

See the following sections for additional information:

- For more information about all available voices, see Languages and voices.
- For more information about using the WebSocket interface to obtain word timings, see Generating word timings.
- For more information about customization, see Understanding customization.
- For more information about the use of IPA and SPR with customization, see Understanding phonetic symbols. (The IPA symbols for the Arabic, Chinese, Netherlands Dutch, and Korean languages are not yet documented. This documentation will be made available soon.)

24 February 2020

New US English and German neural voices

The service now supports five new neural voices:

- US English: `en-US_EmilyV3Voice`, `en-US_HenryV3Voice`, `en-US_KevinV3Voice`, and `en-US_OliviaV3Voice`
- German: `de-DE_ErikaV3Voice`

The new voices do not support the following SSML elements:

- Expressive SSML with the `<express-as>` element
- Voice Transformation with the `<voice-transformation>` element
- The `volume` attribute of the `<prosody>` element

For more information about these and all available voices, see Languages and voices.

New support for Activity Tracker

The service now supports the use of Activity Tracker events for all customization operations. IBM Cloud Activity Tracker records user-initiated activities that change the state of a service in IBM Cloud. You can use this service to investigate abnormal activity and critical actions and to comply with regulatory audit requirements. In addition, you can be alerted about actions as they happen. For more information, see Activity Tracker events.

18 December 2019

New standard Arabic, Chinese, and Netherlands Dutch voices

The service now supports six new standard voices in three new languages:

- *Arabic:* `ar-AR_OmarVoice`
- *Chinese (Mandarin):* `zh-CN_LiNaVoice`, `zh-CN_WangWeiVoice`, and `zh-CN_ZhangJingVoice`
- *Netherlands Dutch:* `nl-NL_EmmaVoice` and `nl-NL_LiamVoice`

The following information applies to these new standard voices:

- The new voices are beta functions. The voices might not be ready for production use and are subject to change. They are initial offerings that are expected to improve in quality with time and usage.
- The voices do not support the `<mark>` element and `timings` parameter that are available with the WebSocket interface.
- The voices do not support customization or the `/v1/pronunciation` method.
- The voices do not support expressive SSML or voice transformation SSML.
- The voices do support all other SSML elements. However, they support only the International Phonetic Alphabet (IPA), not IBM Symbolic Phonetic Representation (SPR), with the `<phoneme>` element.

For more information about these and all available voices, see Languages and voices.

12 December 2019

Full support for IBM Cloud IAM

The Text to Speech service now supports the full implementation of IBM Cloud Identity and Access Management (IAM). API keys for Watson services are no longer limited to a single service instance. You can create access policies and API keys that apply to more than one service, and you can grant access between services. For more information about IAM, see Authenticating to Watson services.

To support this change, the API service endpoints use a different domain and include the service instance ID. The pattern is `api.{location}.text-to-speech.watson.cloud.ibm.com/instances/{instance_id}`.

- Example HTTP URL for an instance hosted in the Dallas location:

  `https://api.us-south.text-to-speech.watson.cloud.ibm.com/instances/6bbda3b3-d572-45e1-8c54-22d6ed9e52c2`

- Example WebSocket URL for an instance hosted in the Dallas location:

  `wss://api.us-south.text-to-speech.watson.cloud.ibm.com/instances/6bbda3b3-d572-45e1-8c54-22d6ed9e52c2`

For more information about the URLs, see the API and SDK reference.

These URLs do not constitute a breaking change. The new URLs work for both your existing service instances and for new instances. The original URLs continue to work on your existing service instances for at least one year, until December 2020.

New network and data security features

Support for the following new network and data security features is now available:

- *Support for private network endpoints*

  Users of Premium plans can create private network endpoints to connect to the Text to Speech service over a private network. Connections to private network endpoints do not require public internet access. For more information, see Public and private network endpoints.

12 November 2019

New Seoul location now available

The Text to Speech service is now available in the IBM Cloud Seoul location (**kr-seo**). As with other locations, the IBM Cloud location uses token-based IAM authentication. All new service instances that you create in this location use IAM authentication.

1 October 2019

New US HIPAA support for Premium plans in Washington, DC, location

US HIPAA support is available for Premium plans that are hosted in the Washington, DC, location, and are created on or after 1 April 2019. For more information, see US Health Insurance Portability and Accountability Act (HIPAA) .

22 August 2019

Defect fix: Multiple small improvements

Defect fix: The service was updated for small defect fixes and improvements.

30 July 2019

New Japanese neural voice: `ja-JP_EmiV3Voice`

The service now offers a neural voice in Japanese: `ja-JP_EmiV3Voice`. Both standard and neural versions of all available voices in all supported languages are now available. For more information, see Languages and voices.

24 June 2019

New support for standard and neural voices

The service now offers two versions of most of its available voices:

- *Standard voices* that use concatenating synthesis to assemble segments of recorded speech to generate audio. Standard voices do not include a version string in their name (for example, `en-US_AllisonVoice`).
- *Neural voices* that use Deep Neural Networks (DNNs) to predict the acoustic (spectral) features of the speech. Neural voices include a version string (`V3`) in their name (for example, `en-US_AllisonV3Voice`).

Neural versions are available for all standard voices except for the `ja-JP_EmiVoice` voice, which is pending and will be available soon. You cannot use the SSML `<express-as>` and `<voice-transformation>` elements with the neural voices, and you cannot use the `volume` attribute of the `<prosody>` element with the neural voices. For more information about all available voices, see Languages and voices.

V2 neural voices withdrawn from service

The service no longer includes the `V2` DNN voices that were previously available. If you use a `V2` voice in your application, the service automatically uses the equivalent `V3` voice instead.

24 March 2019

New V2 neural German voices

The service now offers `V2` Deep Neural Network (DNN) versions of its German voices:

- `de-DE_BirgitV2Voice`
- `de-DE_DieterV2Voice`

For more information about DNN-based voices, see Languages and voices.

New support for `pitch` and `rate` attributes of the SSML `<prosody>` element

All of the service's DNN-based voices now support the `pitch` and `rate` attributes of the SSML `<prosody>` element. The DNN-based voices do not support the `volume` attribute of the `<prosody>` element. For more information, see The prosody element.

21 March 2019

Visibility of service credentials now restricted by role

Users can now see only service credential information that is associated with the role that has been assigned to their IBM Cloud account. For example, if you are assigned a `reader` role, any `writer` or higher levels of service credentials are no longer visible.

This change does not affect API access for users or applications with existing service credentials. The change affects only the viewing of credentials within IBM Cloud.

4 March 2019

New V2 neural English and Italian voices

The service now offers four new `V2` voices that use deep-learning synthesis to generate audio:

- `en-US_AllisonV2Voice`
- `en-US_LisaV2Voice`
- `en-US_MichaelV2Voice`
- `it-IT_FrancescaV2Voice`

These new voices use machine learning and a DNN to synthesize text to speech. Deep-learning, or Deep Neural Network (DNN)-based, synthesis produces audio with a more natural prosody and a more consistent overall quality.

But the new voices also produce audio with different signal qualities from the existing voices, so they might not be appropriate for all applications. Also, the new voices do not support the SSML elements `<prosody>`, `<express-as>`, and `<voice-transformation>`.

For more information about these DNN-based voices and how they differ from the existing voices, see Languages and voices.

28 January 2019

New support for IBM Cloud IAM by WebSocket interface

The WebSocket interface now supports token-based Identity and Access Management (IAM) authentication from browser-based JavaScript code. The limitation to the contrary has been removed. To establish an authenticated connection with the WebSocket `/v1/synthesize` method:

- If you use IAM authentication, include the `access_token` query parameter.
- If you use Cloud Foundry service credentials, include the `watson-token` query parameter.

For more information, see Open a connection.

13 December 2018

New London location now available

The Text to Speech service is now available in the IBM Cloud® London location (**eu-gb**). Like all locations, London uses token-based Identity and Access Management (IAM) authentication. All new service instances that you create in this location use IAM authentication.

7 November 2018

New Tokyo location now available

The Text to Speech service is now available in the IBM Cloud® Tokyo location (**jp-tok**). Like all locations, Tokyo uses token-based Identity and Access Management (IAM) authentication. All new service instances that you create in this location use IAM authentication.

30 October 2018

New support for token-based IBM Cloud IAM

The Text to Speech service has migrated to token-based Identity and Access Management (IAM) authentication for all locations. All IBM Cloud services now use IAM authentication. The Text to Speech service migrated in each location on the following dates:

- Dallas (**us-south**): October 30, 2018
- Frankfurt (**eu-de**): October 30, 2018
- Washington, DC (**us-east**): June 12, 2018
- Sydney (**au-syd**): May 15, 2018

The migration to IAM authentication affects new and existing service instances differently:

- *All new service instances that you create in any location* now use IAM authentication to access the service. You can pass either a bearer token or an API key: Tokens support authenticated requests without embedding service credentials in every call; API keys use HTTP basic authentication. When you use any of the Watson SDKs, you can pass the API key and let the SDK manage the lifecycle of the tokens.
- *Existing service instances that you created in a location before the indicated migration date* continue to use the `{username}` and `{password}` from their previous Cloud Foundry service credentials for authentication until you migrate them to use IAM authentication.

For more information, see the following documentation:

- To learn which authentication mechanism your service instance uses, view your service credentials by clicking the instance on the IBM Cloud dashboard.
- For more information about using IAM tokens with Watson services, see Authenticating to Watson services.
- For examples that use IAM authentication, see the API & SDK reference.

12 June 2018

New features for applications hosted in Washington, DC, location

The following features are enabled for applications that are hosted in Washington, DC (**us-east**):

- The service now supports a new API authentication process. For more information, see the 30 October 2018 service update.
- The service now supports the `X-Watson-Metadata` header and the `DELETE /v1/user_data` method. For more information, see Information security.

15 May 2018

New features for applications hosted in Sydney location

The following features are enabled for applications that are hosted in Sydney (**au-syd**):

- The service now supports a new API authentication process. For more information, see the 30 October 2018 service update.
- The service now supports the `X-Watson-Metadata` header and the `DELETE /v1/user_data` method. For more information, see Information security.

2 October 2017

Changes to `audio/l16` audio format

For the `audio/l16` format, you can now optionally specify the endianness of the audio that is returned. (You must already specify the sampling rate.) Examples are `audio/l16;rate=22050;endianness=big-endian` and `audio/l16;rate=22050;endianness=little-endian`; the default is significant endian. For more information, see Using audio formats.

14 July 2017

New support for MP3 (MPEG) audio format

The service now supports the MP3 or Motion Picture Experts Group (MPEG) audio format. For more information about supported audio formats, see Using audio formats.

10 April 2017

New support for Web Media (WebM) audio format

The service now supports the Web Media (WebM) audio format with the Opus or Vorbis codec. The service now also supports the Ogg audio format with the Vorbis codec in addition to the Opus codec. For more information about supported audio formats, see Using audio formats.

New support for Cross-Origin Resource Sharing

The service now supports Cross-Origin Resource Sharing (CORS) to allow browser-based clients to call the service directly. For more information, see using CORS support.

Changes to successful HTTP response codes

The HTTP response codes for successful completion of some methods of the customization interface changed:

- The `POST /v1/customizations` method now returns 201 (instead of 200).
- The `POST /v1/customizations/{customization_id}` method now returns 200 (instead of 201).
- The `POST /v1/customizations/{customization_id}/words` method now returns 200 (instead of 201).
- The `PUT /v1/customizations/{customization_id}/words/{word}` method now returns 200 (instead of 201).

New errors for misuse of Japanese `part_of_speech` parameter

The `POST /v1/customizations/{custom_id}/words` and `PUT /v1/customizations/{customization_id}/words/{word}` methods now return HTTP response code 400 with the error message `Part of speech is supported for ja-JP language only` if you attempt to specify a `part_of_speech` for a language other than Japanese.

Changes to response body for adding words method

The `POST /v1/customizations/{custom_id}/words` method now returns an empty response body (`{}`).

1 December 2016

New Latin American Spanish voice: `es-LA_SofiaVoice`

The service includes a new voice, `es-LA_SofiaVoice`, which is the Latin American equivalent of the `es-US_SofiaVoice` voice. The most significant difference between the two voices concerns how they interpret a `$` (dollar sign): The Latin American version uses the term *pesos*; the North American version uses the term *dolares*. Other minor differences might also exist between the two voices.

New US English voices: `en-US_LisaVoice` and `en-US_MichaelVoice`

In addition to the `en-US_AllisonVoice`, two more voices are now transformable with SSML voice transformation: `en-US_LisaVoice` and `en-US_MichaelVoice`.

Changes to customization for Japanese

When you use the customization interface with Japanese, the service now matches the longest word from the word/translation pairs that are defined

for a custom model. For example, consider the following two entries for a custom model:

```
{
  "words": [
    {"word":"ＮＹ", "translation":"ニューヨーク", "part_of_speech":"Mesi"},
    {"word":"ＮＹＣ", "translation":"ニューヨークシティ", "part_of_speech":"Mesi"}
  ]
}
```

If the service finds the string `ＮＹＣ` in the input text, it matches that word because it is a longer match than `ＮＹ`. Previously, the service would have matched the string `ＮＹ`. For more information about working with Japanese entries for a custom model, see Working with Japanese entries.

22 September 2016

Customization is now available for all languages

The customization interface, which includes the customization and `GET /v1/pronunciation` methods, is now available for all languages that are supported by the service. The interface remains a beta release. For more information, see Understanding customization.

New Japanese support for SSML

The service now supports SSML for Japanese. For general information about SSML support, see Understanding SSML. For information about Japanese SPR and IPA symbols, see Japanese symbols. Extra considerations and a `part_of_speech` field apply when creating entries for words in a Japanese custom model. For more information, see Working with Japanese entries.

New voice transformation SSML feature

The service now offers SSML voice transformation via the new `<voice-transformation>` element. You can expand the range of possible voices by creating custom transformations that modify the pitch, pitch range, glottal tension, breathiness, rate, and timbre of a voice. The service also offers two built-in virtual voices, *Young* and *Soft*. The service currently supports voice transformation only for the US English Allison voice.

Word timings now available with WebSocket interface

The service can now return word timing information for all strings of the input text that you pass to the WebSocket interface. To receive the start and end time of every string in the input, specify an array that includes the string `words` for the optional `timings` parameter of the JSON object that you pass to the service. The feature is not currently available for Japanese input text. For more information, see Generating word timings.

New support for SSML validation

The service now validates all SSML elements that you submit in any context. If it finds an invalid tag, the service reports an HTTP 400 response code with a descriptive message, and the method fails. In previous releases, the service handled errors inconsistently; specifying an invalid word pronunciation, for example, might lead to unpredictable or inconsistent behavior. For more information, see SSML validation.

IBM SPR format now specified with `ibm` instead of `spr`

The use of `spr` is deprecated as an argument to the `format` option of the `GET /v1/pronunciation` method and for use with the `alphabet` attribute of an SSML `<phoneme>` element. To use IBM SPR notation, use the `ibm` argument instead of `spr` in all cases.

New support for `audio/mulaw` audio format

The list of supported audio formats now includes `audio/mulaw;rate={rate}`. Like `audio/basic`, this format provides single-channel audio that is encoded by using 8-bit u-law (or mu-law) data that is sampled at 8 kHz. For more information, see Using audio formats.

New supported features identified when listing voices

The `GET /v1/voices` and `GET /v1/voices/{voice}` methods now return a `supported_features` object as part of their output for each voice. The object describes whether the voice supports customization and the SSML `<voice_transformation>` element. For more information, see Languages and voices.

23 June 2016

New WebSocket interface for speech synthesis

The service now offers a WebSocket interface for synthesizing text to speech. The interface offers the same features as the `/v1/synthesize` method of the HTTP interface. It accepts plain text or text that is marked up with SSML. In addition, it also supports use of the SSML `<mark>` element to identify the time in the audio at which it finishes synthesizing all text that precedes the mark. For more information, see The WebSocket interface.

Expanded language support for SSML

The service now offers support for text that is annotated with SSML for the languages Castilian and North American Spanish, Italian, and Brazilian Portuguese. The service already supported the use of SSML for US and UK English, French, and German. As of this update, the service supports SSML for all languages but Japanese. Moreover, you can use both IBM SPR and IPA notations to define word pronunciations with the SSML `<phoneme>` element. For more information, see Understanding SSML and Understanding phonetic symbols.

For US English, you can also use the SSML `<phoneme>` element to create word entries in a custom model; customization is supported only for US English. For more information, see Understanding customization.

Voices updated for improved speech synthesis

The service features improved expressiveness and naturalness for the most frequently used voices. The improvements are based on Recursive Neural Network (RNN)-based prosody prediction from input text. They are made available as a new service engine and voice-model updates for the following languages:

- `en-US_AllisonVoice`
- `en-US_LisaVoice`
- `en-US_MichaelVoice`
- `es-ES_EnriqueVoice`
- `fr-FR_ReneeVoice`

New customization ID parameter for word pronunciation

The `GET /v1/pronunciation` method now accepts an optional `customization_id` query parameter. The parameter obtains a word translation from a specified custom model. If the custom model does not contain the word, the method returns the word's default pronunciation. For more information, see the API and SDK reference.

When using the `GET /v1/pronunciation` method without a customization ID and for a language other than US English, you can request a word's pronunciation only in IBM SPR notation. For a language other than US English, you must specify `spr` with the method's `format` option.

New support for `audio/basic` audio format

The list of supported audio formats now includes `audio/basic`, which provides single-channel audio that is encoded using 8-bit u-law (or mu-law) data that is sampled at 8 kHz. For more information, see Using audio formats.

HTTP and WebSocket interfaces can now return warnings

The HTTP and WebSocket `/v1/synthesize` methods can return a `warnings` response that includes messages about invalid query parameters or JSON fields that are included with a request. The format of the warnings changed. The following example shows the previous format:

```
"warnings": "Unknown arguments: [u'{invalid_arg_1}', u'{invalid_arg_2}']."
```

The same warning now has the following format:

```
"warnings": "Unknown arguments: {invalid_arg_1}, {invalid_arg_2}."
```

10 March 2016

Synthesis methods can now return warnings

The `GET` and `POST /v1/synthesize` methods can now return a `Warnings` response header that includes a list of warning messages about invalid query parameters or JSON fields that are included with the request. Each element of the list includes a string that describes the nature of the warning followed by an array of invalid argument strings; for example, `Unknown arguments: [u'{invalid_arg_1}', u'{invalid_arg_2}'].` For more information, see the API & SDK reference.

Beta Apple iOS SDK is deprecated

The beta *Watson Speech Software Development Kit (SDK) for the Apple® iOS operating system* is deprecated and replaced by the *Watson Swift SDK*. The new SDK is available from the swift-sdk repository in the `watson-developer-cloud` namespace on GitHub.

22 February 2016

New expressive SSML feature

The service was updated with a new expressive SSML feature. The service extends SSML with an `<express-as>` element that you can use to indicate expressiveness in one of three speaking styles: `GoodNews`, `Apology`, or `Uncertainty`. You can apply the element to the entire body of the text, a sentence, a phrase, or a word. The service currently supports expressiveness only for the US English Allison voice (`en-US_AllisonVoice`).

17 December 2015

New beta customization interface

The service offers a new customization interface that you can use to specify how it pronounces unusual words that occur in your input. The interface includes some new methods that you can use to create and manage custom models and the word/translation pairs that they contain. You can then use your custom models when synthesizing text to audio.

The service supports sounds-like translations and phonetic translations. Phonetic translations can use either the standard International Phonetic Alphabet (IPA) representation or the proprietary IBM Symbolic Phonetic Representation (SPR). You use SSML to specify phonetic translations.

The customization interface includes a collection of new HTTP methods that have the names `POST /v1/customizations`, `POST /v1/customizations/{customization_id}`, `POST /v1/customizations/{customization_id}/words`, and `PUT /v1/customizations/{customization_id}/words/{word}`. The service also provides a new `GET /v1/pronunciation` method that returns the pronunciation for any word and a new `GET /v1/voices/{voice}` method that returns detailed information about a specific voice. In addition, existing methods of the service's interface now accept custom model parameters as needed.

For more information about customization and its interface, see Understanding customization and the API and SDK reference.

The customization interface is a beta release that currently supports US English only. All customization methods and the `GET /v1/pronunciation` method can currently be used to create and manipulate custom models and word translations only in US English.

New Brazilian Portuguese voice: `pt-BR_IsabelaVoice`

The service supports a new voice, `pt-BR_IsabelaVoice`, to synthesize audio in Brazilian Portuguese with a female voice. For more information, see Languages and voices.

21 September 2015

New mobile SDKs available

Two new beta mobile Software Development Kits (SDKs) are available for the speech services. The SDKs enable mobile applications to interact with both the Text to Speech and Speech to Text services. You can use the SDKs to send text to the Text to Speech service and receive an audio response.

- The *Watson Swift SDK* is available from the swift-sdk repository in the `watson-developer-cloud` namespace on GitHub.
- The *Watson Speech Android™ SDK* is available from the speech-android-sdk repository in the `watson-developer-cloud` namespace on GitHub.

Both SDKs support authenticating with the speech services by using either your IBM Cloud service credentials or an authentication token. Because the SDKs are beta functions, they are subject to change in the future.

New Japanese voice: `ja-JP_EmiVoice`

The service supports a new language, Japanese. The voice `ja-JP_EmiVoice` is a Japanese female voice.

1 July 2015

The Text to Speech service is now generally available

The service moved from beta to general availability (GA) on 1 July 2015. The following differences that are existed between the beta and GA versions of the Text to Speech API. The GA release requires that users upgrade to the new version of the service.

New token-based programming model

A new programming model supports direct interaction between a client and the service. By using this model, a client can obtain an authentication token for communicating directly with the service. By using the token, the client can bypass the need for a server-side proxy application in IBM Cloud to call the service on its behalf. Tokens are the preferred means for clients to interact with the service.

The service continues to support the old programming model that relied on a server-side proxy to relay communications and data between the client and the service. But the new model is more efficient and provides higher throughput.

New support for the Speech Synthesis Markup Language

You can now pass Speech Synthesis Markup Language (SSML) to the HTTP `GET` and `POST` versions of the `/v1/synthesize` method. SSML is an XML-based markup language that is designed to provide annotations of text for speech synthesis applications such as the Text to Speech service. For more information about passing SSML input to the service, see Specifying input text.

The service initially supports the use of SSML only for the UK and US English, French, and German languages. The service does not support SSML for use with Italian and Spanish. When you use SSML, make sure that you do not select a voice for the audio in one of the unsupported languages. Results in this case are not meaningful.

Changes to available voices

The voices that are supported for synthesized speech that is changed and expanded. The service now supports several other voices, languages, and dialects with the `/v1/synthesize` methods. For more information about supported voices, see Languages and voices.

The three voices that were available at beta are renamed for GA:

- `VoiceEnUsMichael` is now `en-US_MichaelVoice`
- `VoiceEnUsLisa` is now `en-US_LisaVoice`
- `VoiceEsEsEnrique` is now `es-ES_EnriqueVoice`

The previous names of the voices continue to work with the beta version of the service (via `-beta` API endpoints) while that version remains available. However, you must use the new names with the GA version of the service.

New support for Free Lossless Audio Codec (FLAC) audio format

You can now request the service to return audio in the Free Lossless Audio Codec (FLAC) format. The service can still return audio in the Ogg format with the Opus codec (the default) and in the Waveform Audio File Format (WAV). For more information about using audio formats with the `/v1/synthesize` methods, see Using audio formats.

New limits on maximum amount of synthesized text

The text that you send to the `/v1/synthesize` method in the URL of an HTTP `GET` request or in the body of an HTTP `POST` request is now limited to a maximum of 5 KB. The text had a maximum size of 4 MB for the beta version.

New header to opt out of contributing to service improvements

The `/v1/synthesize` methods now include the header `X-WDC-PL-OPT-OUT` to control whether the service uses the text and audio results from an operation to improve future results. Specify a value of `1` for the header to prevent the service from using text and audio results. The parameter applies only to the current request. The new header replaces the `X-logging` header from the beta methods. For more information, see Controlling request logging for Watson services.

Changes to HTTP error codes for speech synthesis

For the `/v1/synthesize` methods, the following error codes changed:

- Error code 406 ("Not acceptable. Unsupported MIME type.") is removed.
- Error code 415 ("Unsupported Media Type") is added.
- Error code 503 ("Service Unavailable") is added.

Changes to HTTP error codes for listing voices

For the `GET /v1/voices` method, the following error codes changed:

- Error code 406 ("Not acceptable. Unsupported MIME type.") is removed.
- Error code 415 ("Unsupported Media Type") is added.

Release notes for Text to Speech for IBM Cloud Pak for Data

IBM Cloud Pak for Data

The following features and changes were included for each release and update of installed or on-premises instances of IBM Watson® Text to Speech for IBM Cloud Pak for Data. Unless otherwise noted, all changes are compatible with earlier releases and are automatically and transparently available to all new and existing applications.

For information about known limitations of the service, see Known limitations.

Note: For information about releases and updates of the service for IBM Cloud, see Release notes for Text to Speech for IBM Cloud.

30 October 2024 (Version 4.8.7)

Version 4.8.7 is now available

Speech to Text for IBM Cloud Pak for Data version 4.8.7 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

25 September 2024 (Version 5.0.3)

Version 5.0.3 is now available

Speech to Text for IBM Cloud Pak for Data version 5.0.3 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

28 August 2024 (Version 4.8.6)

Version 4.8.6 is now available

Speech to Text for IBM Cloud Pak for Data version 4.8.6 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

28 August 2024 (Version 5.0.2)

Version 5.0.2 is now available

Speech to Text for IBM Cloud Pak for Data version 5.0.2 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

31 July 2024 (Version 5.0.1)

Version 5.0.1 is now available

Speech to Text for IBM Cloud Pak for Data version 5.0.1 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

19 June 2024 (Version 5.0.0)

Version 5.0.0 is now available

Speech to Text for IBM Cloud Pak for Data version 5.0.0 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

24 April 2024 (Version 4.8.5)

Version 4.8.5 is now available

Speech to Text for IBM Cloud Pak for Data version 4.8.5 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

27 March 2024 (Version 4.8.4)

Version 4.8.4 is now available

Speech to Text for IBM Cloud Pak for Data version 4.8.4 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

28 February 2024 (Version 4.8.3)

Version 4.8.3 is now available

Speech to Text for IBM Cloud Pak for Data version 4.8.3 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

31 January 2024 (Version 4.8.2)

Version 4.8.2 is now available

Speech to Text for IBM Cloud Pak for Data version 4.8.2 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

30 November 2023 (Version 4.8.0)

Version 4.8.0 is now available

Speech to Text for IBM Cloud Pak for Data version 4.8.0 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

27 September 2023 (Version 4.7.3)

Version 4.7.3 is now available

Speech to Text for IBM Cloud Pak for Data version 4.7.3 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

28 July 2023 (Version 4.7.1)

Version 4.7.1 is now available

Speech to Text for IBM Cloud Pak for Data version 4.7.1 is now available. For more information, see Watson Speech services on IBM Cloud Pak for

Data.

9 June 2023 (Version 4.7.0)

Version 4.7.0 is now available

Speech to Text for IBM Cloud Pak for Data version 4.7.0 is now available. For more information, see Watson Speech services on IBM Cloud Pak for Data.

2 May 2023 (Version 4.6.5)

Version 4.6.5 is now available

Text to Speech for IBM Cloud Pak for Data version 4.6.5 is now available. This version supports IBM Cloud Pak for Data version 4.6.x and Red Hat OpenShift versions 4.10 and 4.12. For more information, see Watson Speech services on IBM Cloud Pak for Data .

New Australian English expressive neural voices

The service now supports two new expressive neural voices for Australian English:

- `en-AU_HeidiExpressive`
- `en-AU_JackExpressive`

Expressive neural voices offer natural-sounding speech that is exceptionally clear, crisp, and fluid. The new voices are generally available (GA) for production use. They support the use of both standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols. For more information, see

- Expressive neural voices
- English (Australian) symbols

New Korean enhanced neural voice

The service now supports a new enhanced neural voice for Korean: `ko-KR_JinV3Voice`. The new voice is generally available (GA) for production use. It supports the use of both standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols. For more information, see

- Enhanced neural voices
- Korean symbols

New beta Netherlands Dutch enhanced neural voice

The service now supports a new enhanced neural female voice for Netherlands Dutch: `nl-NL_MerelV3Voice`. It supports the use of both standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols.

The new voice is beta functionality pending completion of support for SSML. At its initial release, the voice does not support use of the following SSML-related functionality:

- The `<prosody>` element with any speech synthesis request
- The `rate_percentage` and `pitch_percentage` parameters with any speech synthesis request
- The `<mark>` element with a WebSocket speech synthesis request
- The `timings` parameter of the JSON text message with a WebSocket speech synthesis request

For more information about the new voice, its support for IPA and SPR symbols, and migrating to the new voice from the deprecated Netherlands Dutch neural voices, see

- Enhanced neural voices
- Dutch (Netherlands) symbols

New environment variable for Speech services custom resource

The documentation now includes instructions to create an environment variable named `${CUSTOM_RESOURCE_SPEECH}`. You append the new variable to the `cpd_vars.sh` script, and source the script to use the variable in your environment. For more information, see *Information you need to*

complete this task in Installing Watson Speech services, or refer to any of the upgrade topics for the Speech services.

Defect fix: French Canadian voice now handles numeric times properly

Defect fix: The French Canadian voices now pronounce times like `19:41` correctly. Previously, the voices were omitting elements of the time in the synthesized audio.

Defect fix: Japanese voice no longer inserts unexpected audio

Defect fix: The Japanese voice no longer inserts unexpected audio in speech synthesis results. Previously, additional audio was inserted in certain cases.

Defect fix: Update Korean phonetic symbols in documentation

Defect fix: In the documentation for Korean SPR symbols, two-character symbols for consonants are now enclosed in single quotes, making them a single symbol. Previously, they were shown as two separate symbols, without enclosing quotes. For more information, see Consonants (Korean).

Documentation updates for IBM SPR symbols

The overview documentation for IBM SPR symbols has been updated to clarify the use of multi-character symbols. For more information, see Speech sound symbols).

Security vulnerabilities addressed

The following security vulnerabilities have been fixed:

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Python (CVE-2020-10735)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to phishing attacks in Python (CVE-2021-28861)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Pypa Setuptools (CVE-2022-40897)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a sensitive information exposure in systemd (CVE-2022-4415)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Python (CVE-2022-45061)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution in Libksba (CVE-2022-47629)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a heap-based buffer overflow in GNU Tar (CVE-2022-48303)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in FasterXML jackson-databind (CVE-2022-42003)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution in Perl (CVE-2020-10878)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a security restrictions bypass in Apache Tomcat (CVE-2022-45143)
- CVE-2020-10543: Publication of the security bulletin is pending.

29 March 2023 (Version 4.6.4)

Version 4.6.4 is now available

Text to Speech for IBM Cloud Pak for Data version 4.6.4 is now available. This version supports IBM Cloud Pak for Data version 4.6.x and Red Hat OpenShift versions 4.10 and 4.12. For more information, see Watson Speech services on IBM Cloud Pak for Data .

Important: Back up your data before upgrading to version 4.6.3 or 4.6.4

Important: Before upgrading to Watson Speech services version 4.6.3 or 4.6.4, you must make a backup of your data. Preserve the backup in a safe location. For more information about backing up your Watson Speech services data, see *Backing up and restoring Watson Speech services data* in Administering Watson Speech services. That topic also includes information about restoring your data if that becomes necessary.

Defect fix: You can now change the installed models and voices with the advanced installation options

Defect fix: During installation, you can now specify different models or voices with the advanced installation options of the command-line interface.

Previously, the service always installed the default models and voices. The limitation continues to apply for Watson Speech services versions 4.6.0, 4.6.2, and 4.6.3. For information about installing models and voices, see *Specifying additional installation options* in Installing Watson Speech services.

Setting load balancer timeouts

Watson Speech services require that you change the load balancer timeout settings for both the server and client to 300 seconds. These settings ensure that long-running speech recognition requests, those with long or difficult audio, have sufficient time to complete. For more information, see *Information you need to complete this task* in Installing Watson Speech services.

Documentation updates for IBM SPR symbols

The overview documentation for IBM SPR symbols has been updated to clarify the use of multi-character symbols. For more information, see Speech sound symbols.

Security vulnerabilities addressed

The following security vulnerabilities have been fixed:

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to cross-site scripting in GNOME libxml2 (CVE-2016-3709
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in SQlite (CVE-2020-35525)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a security restrictions bypass in Amazon AWS S3 Crypto SDK for GoLang (CVE-2020-8912)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to elevated system privileges in the Red Hat Build of OpenJDK (CVE-2021-20264)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to an arbitrary code execution in e2fsprogs (CVE-2022-1304)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to errors in TrustCor (CVE-2022-23491)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in GnuTLS (CVE-2022-2509)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to an arbitrary code execution in systemd (CVE-2022-2526)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to sensitive information exposure in AWS SDK for Go (CVE-2022-2582)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to denial of service in cURL libcurl (CVE-2022-32206)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a man-in-the-middle attack in cURL libcurl (CVE-2022-32208)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to spoofing attacks in GnuPG (CVE-2022-34903)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in SQLite (CVE-2022-35737)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a heap-based buffer overflow in zlib (CVE-2022-37434)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in systemd (CVE-2022-3821)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to an arbitrary code execution in Gnome libxml2 (CVE-2022-40303)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to an arbitrary code execution in Gnome libxml2 (CVE-2022-40304)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Python Charmers Future (CVE-2022-40899)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a security restrictions bypass in Golang Go (CVE-2022-41716)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Golang Go (CVE-2022-41717)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Freedesktop D-Bus (CVE-2022-42010)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Freedesktop D-Bus

(CVE-2022-42011)

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Freedesktop D-Bus (CVE-2022-42012)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in MIT krb5 (CVE-2022-42898)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in libexpat (CVE-2022-43680)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to an arbitrary commands execution in Python (CVE-2015-20107)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution in SQlite (CVE-2020-35527)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a security restrictions bypass in GNU Libtasn1 (CVE-2021-46848)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution in Git (CVE-2022-23521)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution in GnuPG Libksba (CVE-2022-3515)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to an arbitrary code execution in libexpat (CVE-2022-40674)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution in Git (CVE-2022-41903)

23 February 2023 (Version 4.6.3)

Version 4.6.3 is now available

Text to Speech for IBM Cloud Pak for Data version 4.6.3 is now available. This version supports IBM Cloud Pak for Data version 4.6.x and Red Hat OpenShift version 4.10. Red Hat OpenShift version 4.8 is no longer supported. For more information, see Watson Speech services on IBM Cloud Pak for Data.

Known issue: You cannot change the installed models and voices with the advanced installation options

Known issue: You currently cannot specify different models or voices with the advanced installation options. The service always installs the default models and voices. For information about changing the models after installation, see *Updating models and voices for your Watson Speech services* in the *Administration* topic of Watson Speech services on IBM Cloud Pak for Data .

Known issue: Upgrade to version 4.6.3 can fail to complete

Known issue: When upgrading to version 4.6.3, the MinIO backup job can fail to be deleted upon completion. If this happens, the solution is to delete the job, after which the upgrade proceeds normally. Perform the following steps to resolve the problem.

1. To determine whether the MinIO backup job remains undeleted, issue the following command:

   ```
   $ oc get job --namespace {${PROJECT_CPD_INSTANCE} | grep speech-cr-ibm-minio-backup
   ```

   The MinIO job that is not deleted is identified by an entry of the following form:

   ```
   speech-cr-ibm-minio-backup   1/1   3m25s   1d
   ```

2. To delete the MinIO backup job, issue the following command:

   ```
   $ oc delete job speech-cr-ibm-minio-backup --namespace ${PROJECT_CPD_INSTANCE}
   ```

Once the backup job is deleted, upgrade continues and completes.

Additional information about working with service instances

The documentation now includes information about creating a service instance with the command-line interface (`cpl-cli`) and about managing service instances. For more information, see the following topics of Watson Speech services on IBM Cloud Pak for Data :

- *Creating a Watson Speech services instance* under *Post-installation setup*
- *Managing your Watson Speech services instances* under *Administering*

Defect fix: The beta Tune by Example is now available

Defect fix: The beta Tune by example feature is now available for Text to Speech for IBM Cloud Pak for Data. Previously, it was not possible to create speaker models.

Defect fix: Specifying large cardinal numbers with the `<say-as>` element no longer causes errors for English voices

Defect fix: You can now use the `<say-as>` element to pronounce large numbers as cardinal numbers. Previously, enclosing a large number in the `<say-as>` element with the attribute `interpret-as="cardinal"` could cause speech synthesis to fail for English voices. For example, `<say-as interpret-as="cardinal">3,200</say-as>` could cause the service to generate an error. For more information, see cardinal in the topic *SSML elements*.

Defect fix: Homonyms and other words are now pronounced correctly by English voices

Defect fix: The service now pronounces homonyms and other words correctly based on their context in English text that is to be synthesized. Previously, words such as `advocate` and `wifi` could be pronounced incorrectly by English voices.

Security vulnerability addressed

The following security vulnerability has been fixed:

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to denial of service in Pypa Setuptools (CVE-2022-40897)

30 January 2023 (Version 4.6.2)

Version 4.6.2 is now available

Text to Speech for IBM Cloud Pak for Data version 4.6.2 is now available. This version supports IBM Cloud Pak for Data version 4.6.x and Red Hat OpenShift versions 4.8 and 4.10. For more information, see Watson Speech services on IBM Cloud Pak for Data .

The custom resource now includes a new `fileStorageClass` property

The custom resource for the Watson Speech services now includes a `fileStorageClass` property in addition to the existing `blockStorageClass` property. You specify both block and file storage classes when you install or upgrade a service. During upgrade from a previous version, the new property is added automatically to the custom resource by the `--file_storage_class` option on `cli manage apply-cr` command.

For more information about the available block and file storage classes you use with each of the supported storage solutions, see the table of *Storage requirements* under *Information you need to complete this task* on the page "Installing Watson Speech services" in Watson Speech services on IBM Cloud Pak for Data.

Additional information about provisioning a service instance

The documentation now includes information about creating a service instance programmatically. It also includes examples of listing service instances and deleting a service instance. For more information, see *Creating a Watson Speech services instance* in the *Post-installation setup* documentation in Watson Speech services on IBM Cloud Pak for Data .

Server-side encryption is enabled for the MinIO datastore

The Speech services have now enabled server-side encryption for object storage in the MinIO datastore. No action is required on your part.

Change to audit webhooks

The Speech services have now removed the audit webhook dependency. The services now write audit events directly to the server. After upgrading to version 4.6.2, some webhook resources might remain until all services can remove the dependency. The remaining resources will be removed in a future release. No action is required on your part.

New US English expressive neural voices

The service offers four new expressive neural voices for US English:

- `en-US_AllisonExpressive`
- `en-US_EmmaExpressive`
- `en-US_LisaExpressive`

- `en-US_MichaelExpressive`

  Expressive neural voices offer natural-sounding speech that is exceptionally clear, crisp, and fluid. The new voices are generally available (GA) for production use. They support the use of both standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols. For more information, see

  - Supported languages and voices
  - Expressive neural voices

New speaking styles with expressive neural voices

The expressive neural voices determine the sentiment of the text from the context of its words and phrases. The speech that they produce, in addition to having a very conversational style, reflects the mood of the text. But you can embellish the voices' natural tendencies by indicating that all or some of the text is to emphasize one of the following speaking styles:

- **Cheerful** - Expresses happiness and good news.
- **Empathetic** - Expresses empathy or sympathy.
- **Neutral** - Expresses objectivity and evenness.
- **Uncertain** - Expresses confusion or uncertainty.

For more information, see Using speaking styles.

New interjection emphasis with expressive neural voices

With expressive neural voices, the service automatically detects a set of common interjections based on context. When it synthesizes these interjections, it gives them the natural emphasis that a human would use in normal conversation. For some of the interjections, you can use SSML to enable or disable their emphasis. For more information, see Emphasizing interjections.

New word emphais with expressive neural voices

The expressive voices use a conversational style that naturally applies the correct intonation from context. But you can indicate that one or more words are to be given more or less emphasis. The change in stress can be indicated by an increase or decrease in pitch, timing, volume, or other acoustic attributes. For more information, see Emphasizing words.

The service now enforces stricter SSML validation

The service now enforces stricter validation of input text that includes Speech Synthesis Markup Language (SSML) elements. Required elements of attributes must be specified with valid values. Otherwise, the request fails with a 400 error code. For more information about SSML validation and the requirements that marked-up text must meet, see SSML validation.

Defect fix: The gender listed for the `en-US_MichaelExpressive` voice is now correct

Defect fix: When you list information about the available voices, the `gender` of the `en-US_MichaelExpressive` voice is now `male`. Previously, the voice's gender was mistakenly described as `female`. For more information, see Listing information about voices.

Security vulnerabilities addressed

The following security vulnerabilities have been fixed:

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to issues in OpenSSL (CVE-2022-1434, CVE-2022-1343, CVE-2022-1292, CVE-2022-1473)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary command execution in OpenSSL (CVE-2022-2068)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in protobuf (CVE-2022-1941)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a buffer overflow in GNU glibc (CVE-2021-3999)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a security bypass in GNU gzip (CVE-2022-1271)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Golang Go (CVE-2022-27664)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Golang Go (CVE-2022-2879)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to query parameter smuggling in Golang Go (CVE-2022-2880)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Golang Go (CVE-

2022-32189)

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Golang Go (CVE-2022-41715)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to information exposure in OpenSSL (CVE-2022-2097)

30 November 2022 (Version 4.6.0)

Version 4.6.0 is now available

Text to Speech for IBM Cloud Pak for Data version 4.6.0 is now available. This version supports IBM Cloud Pak for Data version 4.6.x and Red Hat OpenShift versions 4.8 and 4.10. For more information, see Watson Speech services on IBM Cloud Pak for Data .

Amazon Web Services (AWS) is now supported

Watson Speech services for IBM Cloud Pak for Data are now supported on Amazon Web Services™ (AWS™). The services support Amazon Elastic Block Store, which you specify by setting the `blockStorageClass` property of the Speech services custom resource to `gp2-csi` or `gp3-csi` .

New storage classes are now supported

Watson Speech services for IBM Cloud Pak for Data now support two additional storage classes:

- IBM Cloud Block Storage (`ibmc-block-gold`)
- NetApp Trident (`ontap-nas`)

You specify the storage class with the `blockStorageClass` property of the Speech services custom resource. For more information about all supported storage classes, see the following topics in Watson Speech services on IBM Cloud Pak for Data :

- *Before you begin* in *Installing Watson Speech services*
- *Specifying a storage class* in *Using the Watson Speech services custom resource*

Known issue: Some Watson Speech services pods do not have annotations that are used for scheduling

Known issue: Some Watson Speech services pods are missing the `cloudpakInstanceId` annotation. If you use the IBM Cloud Pak for Data scheduling service, any Watson Speech services pods without the `cloudpakInstanceId` annotation are

- Scheduled by the default Kubernetes scheduler rather than the scheduling service
- Not included in the quota enforcement

Monitoring of the PostgreSQL datastore is now available

You can now enable monitoring of the PostgreSQL datastore to receive updates on its usage and status by the Watson Speech services. The events can be consumed by Prometheus monitoring software or whatever application you use for monitoring. By enabling monitoring for user-defined projects in addition to the default platform monitoring, you can monitor your own projects with the Red Hat® OpenShift® Container Platform monitoring stack. This capability includes an additional property, `spec.global.datastores.postgressql.enablePodMonitor` , in the Speech services custom resource.

For more information, see the topic *Monitoring the PostgreSQL datastore for Watson Speech services* in the *Administering* section of Watson Speech services on IBM Cloud Pak for Data.

Defect fix: PostgreSQL datastore is no longer installed if only runtime microservices are enabled

Defect fix: The PostgreSQL datastore is no longer installed if only the runtime microservices are enabled. The datastore is now installed only if at least one of the `sttAsync` , `sttCustomization` , or `ttsCustomization` microservices is installed. PostgreSQL is not uninstalled if at a later date these microservices are disabled.

Prior to version 4.6.0, PostgreSQL was always installed with the Speech services. If you are an existing customer who used only the runtime microservices of the Speech services prior to version 4.6.0, PostgreSQL remains installed but is not used. In this case, installation of PostgreSQL persists across upgrades.

The MinIO datastore is always installed because the runtime microservices depend on it. The RabbitMQ datastore is installed only if the `sttAsync` microservice is installed.

For more information, see *Datastore properties* in *Using the Watson Speech services custom resource* in Watson Speech services on IBM Cloud Pak for Data.

Defect fix: Creation of a Network Policy is no longer necessary for the PostgreSQL operator to monitor its operands

Defect fix: For version 4.6.0, it is not necessary to create a Network Policy to allow the PostgreSQL operator to monitor its operands, as described in the 10 November 2022 (Versions 4.0.x and 4.5.x) service update. As of version 4.6.0, the service handles this situation automatically.

New beta `rate_percentage` query parameter for controlling the global speaking rate

The service offers a new `rate_percentage` query parameter to modify the speaking rate for a speech synthesis request. The speaking rate is the speed at which the service speaks the text that it synthesizes into speech. A higher rate causes the text to be spoken more quickly; a lower rate causes the text to be spoken more slowly. The parameter changes the per-voice default rate for an entire request. For more information, see Modifying the speaking rate.

New beta `pitch_percentage` query parameter for controlling the global speaking pitch

The service offers a new `pitch_percentage` query parameter to modify the speaking pitch for a synthesis request. The speaking pitch represents the tone of the speech that the service synthesizes. It represents how high or low the tone of the voice is perceived by the listener. A higher pitch results in speech that is spoken at a higher tone and is perceived as a higher voice; a lower pitch results in speech that is spoken in a lower tone and is perceived as a lower voice. The parameter changes the per-voice default pitch for an entire request. For more information, see Modifying the speaking pitch.

Defect fix: Custom word translations now accept commas in all cases

Defect fix: Word translations added to custom models now accept commas in all cases. Previously, a comma in a translation could occasionally cause the translation to fail to generate valid audio when used for speech syntheses. This problem was identified in US English custom models.

Defect fix: French synthesis of dates is now consistent

Defect fix: French synthesis no longer includes the article "le" before dates of the form "the *ordinal* of *month*." Previously, the article was included only for the first day of the month for French (for example, "the first of September," "le premier septembre").

Defect fix: Japanese synthesis is improved to handle long strings of input text

Defect fix: The service now correctly synthesizes Japanese requests that include long strings of characters. Previously, the service failed to properly synthesize very long strings of Japanese text.

Defect fix: Add rules for custom model naming documentation

Defect fix: The documentation now provides detailed rules for naming custom models. For more information, see

- Creating a custom model
- API & SDK reference

Security vulnerabilities addressed

The following security vulnerabilities have been fixed:

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a cross-configuration attack against OpenPGP (CVE-2021-40528)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution in PCRE2 (CVE-2022-1586)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a heap-based buffer overflow in Vim (CVE-2022-1621)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a buffer overflow in Vim (CVE-2022-1629)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution in Vim (CVE-2022-1785, CVE-2022-1897, CVE-2022-1927)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a security restrictions bypass in cURL libcurl (CVE-2022-22576)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to credential exposure in cURL libcurl (CVE-2022-27774)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to data information exposure in cURL libcurl (CVE-2022-27776)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a security restrictions bypass in cURL libcurl (CVE-2022-27782)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in GNOME libxml2 (CVE-2022-29824)

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a SQL injection in PostgreSQL (CVE-2022-31197)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in libexpat (CVE-2022-25313)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution in libexpat (CVE-2022-25314)

10 November 2022 (Versions 4.0.x and 4.5.x)

Known issue: Updated Network Policy needed for PostgreSQL operator

Known issue: For Speech services version 4.0.x (not including version 4.0.0) and 4.5.x, if the PostgreSQL operator and the Speech services are installed in different namespaces, the PostgreSQL operator is not able to monitor the PostgreSQL operands for the Speech services. The operator is prevented from monitoring the operands by the Network Policy that is in place for the Speech services.

This problem does not prevent the PostgreSQL cluster from functioning properly. The cluster remains active and fully functional. However, the operator is not able to update the operands when you upgrade to new versions of the Speech services.

The solution for the problem is to create an additional Network Policy for the PostgreSQL operator, as shown in the following steps. You can perform the steps regardless of whether the PostgreSQL operator is installed in the same namespace as the Speech services or in a different namespace.

1. Log in as an administrator of the Red Hat® OpenShift® project where the Speech services are installed.
2. Enter the following command to update the Network Policy for the Speech services:

```
$ cat << EOF | oc apply -f -
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  labels:
    app.kubernetes.io/component: stt
    app.kubernetes.io/instance: {{ <custom-resource-name> }}
    app.kubernetes.io/name: speech-to-text
    release: {{ <custom-resource-name> }}
  name: <custom-resource-name>-postgres-network-policy
  namespace: {{ <cpd-instance-namespace> }}
spec:
  ingress:
  - from:
    - namespaceSelector: {}
      podSelector:
        matchLabels:
          app.kubernetes.io/name: cloud-native-postgresql
EOF
```

   where

   - `<custom-resource-name>` is the name of the Speech services custom resource. The recommended name for version 4.0.x is `speech-prod-cr`; the recommended name for version 4.5.x is `speech-cr`.
   - `<cpd-instance-name>` is the name of the project (namespace) in which the Speech services are installed. The documentation uses the environment variable `${PROJECT_CPD_INSTANCE}` to identity the namespace.

3. To verify that the updated Network Policy allows the operator to monitor the operands and that the PostgreSQL cluster is in a healthy state, enter the following command, where `<custom-resource-name>` and `<cpd-instance-name>` are the values you used in the previous step:

```
oc -get cluster {{ <custom-resource-name> }}-postgres -n {{ <cpd-instance-namespace> }}
```

   If the PostgreSQL cluster is functioning properly, the command produces output similar to the following:

```
NAME                AGE   INSTANCES   READY   STATUS                  PRIMARY
speech-cr-postgres  14d   3           3       Cluster in healthy state   speech-cr-postgres-1
```

These steps do not cause operator to update the operands to the latest versions. However, the operands are upgraded as expected when you next

upgrade the Speech services software.

13 October 2022 (Version 4.5.3)

Version 4.5.3 is now available

Text to Speech for IBM Cloud Pak for Data version 4.5.3 is now available. This version supports IBM Cloud Pak for Data version 4.5.x and Red Hat OpenShift versions 4.6, 4.8, and 4.10. For more information, see Watson Speech services on IBM Cloud Pak for Data .

Audit events are available for the Speech services

The IBM Cloud Pak for Data Audit Logging Service generates and forwards audit events for both the Speech to Text and Text to Speech services. The audit events match those that are available for Activity Tracker with the public service. For more information, see Audit events.

You cannot uninstall individual Speech service components

The documentation now notes that you cannot uninstall individual service components (microservices) once they are installed. To remove any of the following components, you must uninstall the Watson Speech services in their entirety and reinstall only the components that you need: Speech to Text runtime, Speech to Text asynchronous HTTP, Speech to Text customization, Text to Speech runtime, and Text to Speech customization. For more information about installing the Speech services, see Watson Speech services on IBM Cloud Pak for Data .

New beta `spell_out_mode` parameter for German voices

To indicate how individual characters of a string are to be spelled out, you can now include the beta `spell_out_mode` query parameter with a synthesis request for a German voice. By default, the service spells out individual characters at the same rate at which it synthesizes text for a language. You can use the parameter to direct the service to spell out individual characters more slowly, in groups of one, two, or three characters. Use the parameter with the SSML `<say-as>` element to control how the characters of a string are synthesized. For more information, see Specifying how strings are spelled out.

Known limitation with using the Ogg audio format with the Safari browser

By default, the service returns audio in the Ogg audio format with the Opus codec (`audio/ogg;codecs=opus`). However, the Ogg audio format is not supported with the Safari browser. If you are using the the Text to Speech service with the Safari browser, you must specify a different format in which you want the service to return the audio.

- For more information about the available formats, see Supported audio formats.
- For more information about specifying a format, see Specifying an audio format.

Troubleshooting upgrade from version 4.0.x to version 4.5.x

When you upgrade the Speech services from version 4.0.x to version 4.5.x, you might encounter an issue where the PostgreSQL pods become stuck in the `Terminating` state. If this problem occurs during your upgrade, perform the following steps to resolve the problem. The information and steps are also documented in *Upgrading Watson Speech services from Version 4.0 to Version 4.5* in the *Upgrading* topic of Watson Speech services on IBM Cloud Pak for Data.

1. Use the following command to identify pods that remain in the `Terminating` state:

```
oc get pods -n ${PROJECT_CPD_INSTANCE} -o wide | awk {'print $1'}
```

1. Use the following command to set the environment variable `pods` to include the list of pods that remain in the `Terminating` state:

```
pods=$(oc get pods -n ${PROJECT_CPD_INSTANCE} -o wide | grep Terminating | awk {'print $1'})
```

1. Use the following command to delete the stuck pods so that the upgrade process can continue:

```
oc delete pod $pods -n ${PROJECT_CPD_INSTANCE} --force=true --grace-period=0
```

Documentation updates for the SSML `<prosody>` element

The documentation for the SSML `<prosody>` element and its `pitch` and `rate` parameters has been improved and clarified. It also now includes a description of the differences between the service and the latest version of the SSML specification. For more information, see The <prosody> element.

Security vulnerabilities addressed

The following security vulnerabilities have been fixed:

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a buffer over-read flaw in Linux Kernel (CVE-2020-28915)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a security bypass in GNU Gzip (CVE-2022-1271)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to elevated privileges in Apple macOS Monterey and macOS Big Sur (CVE-2022-26691)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to elevated privileges in Linux Kernel (CVE-2022-27666)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to cross-site scripting in Apache Tomcat (CVE-2022-34305)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a security restrictions bypass in GNU C Library (CVE-2019-19126)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in GNU C Library (CVE-2020-10029)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in GNU glibc (CVE-2020-1751)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in GNU glibc (CVE-2020-1752)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to information disclosure or denial of service in GNU glibc (CVE-2021-35942)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to buffer overflow in OpenSSL (CVE-2021-3711)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to information disclosure or denial of service in OpenSSL (CVE-2021-3712)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to weakened security in OpenSSL (CVE-2021-4160)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in OpenSSL (CVE-2022-0778)

3 August 2022 (Version 4.5.1)

Version 4.5.1 is now available

Text to Speech for IBM Cloud Pak for Data version 4.5.1 is now available. This version supports IBM Cloud Pak for Data version 4.5.x and Red Hat OpenShift versions 4.6, 4.8, and 4.10. For more information, see Watson Speech services on IBM Cloud Pak for Data .

Support for FIPS-enabled clusters

Both Text to Speech for IBM Cloud Pak for Data and Speech to Text for IBM Cloud Pak for Data now support running on Federal Information Processing Standard (FIPS)-enabled clusters. For more information, see Services that support FIPS.

Defect fix: Fixed ephemeral storage calculations to prevent occasional pod evictions

Defect fix: A defect was fixed and calculation of ephemeral storage limits is now more precise for the Text to Speech for IBM Cloud Pak for Data and Speech to Text for IBM Cloud Pak for Data runtimes. These changes prevent occasional pod evictions when the services' runtimes are under heavy load.

The service does not support multilingual speech synthesis

The service does not support multilingual speech synthesis at this time. However, you can use customization to approximate the pronunciation of words from other languages. For more information, see Multilingual speech synthesis.

Security vulnerabilities addressed

The following security vulnerabilities have been fixed:

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a heap-based buffer overflow in rsyslog (CVE-2022-24903)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to an HTTP request smuggling issue in Twisted (CVE-2022-24801)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service, caused by a buffer

overflow in Twisted (CVE-2022-21716)

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service, caused by incomplete string comparison in NumPy (CVE-2021-34141)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service, caused by a buffer overflow in NumPy (CVE-2021-41496)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to cookie and authorization header exposure in Twisted (CVE-2022-21712)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a heap-based buffer overflow in Perl (CVE-2018-18311)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a heap-based buffer overflow in Perl (CVE-2018-18312)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a heap-based buffer overflow in Perl (CVE-2018-18313)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a heap-based buffer overflow in Perl (CVE-2018-18314)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a heap-based buffer overflow in Perl (CVE-2018-6913)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to CRLF injection in Python (CVE-2019-11236)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in GNU Tar (CVE-2019-9923)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a heap-based buffer overflow in Perl (CVE-2020-10543)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to an integer overflow in Perl (CVE-2020-10878)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a buffer overflow in Perl (CVE-2020-12723)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in urllib3 (CVE-2021-33503)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to injection attacks in Ansible (CVE-2021-3583)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Golang Go (CVE-2022-23772)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to incorrect access control in Golang Go (CVE-2022-23773)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Golang Go (CVE-2022-23806)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Golang Go (CVE-2022-24675)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Golang Go (CVE-2022-24921)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Golang Go (CVE-2022-28327)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a heap-based buffer overflow in libssh, caused by improper bounds checking (CVE-2021-3634)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in Python (CVE-2021-3737)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a possible sensitive information exposure in Python (CVE-2021-4189)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a security restrictions bypass in lxml (CVE-2021-43818)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution in MS Visual Studio (CVE-2021-21300)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a security restrictions bypass in Git (CVE-2021-40330)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution in MS Visual

Studio (CVE-2022-24765)

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary command execution in Git (CVE-2018-1000021)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to cross-site scripting in jQuery (CVE-2015-9251)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to cross-site scripting in jQuery (CVE-2019-11358)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to cross-site scripting in jQuery (CVE-2020-11022)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to cross-site scripting in jQuery (CVE-2020-11023)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a data binding rules security weakness in Spring Framework (CVE-2022-22968)

29 June 2022 (Version 4.5.0)

Version 4.5.0 is now available

Text to Speech for IBM Cloud Pak for Data version 4.5.0 is now available. This version supports IBM Cloud Pak for Data version 4.5.x and Red Hat OpenShift versions 4.6, 4.8, and 4.10. For more information, see Watson Speech services on IBM Cloud Pak for Data .

Unified Speech services for IBM Cloud Pak for Data documentation

The installation and administration documentation for both Speech to Text and Text to Speech is now combined in the IBM Cloud Pak for Data documentation. For more information about installing and managing the Speech services, see Watson Speech services on IBM Cloud Pak for Data .

Changes to Speech services custom resource

The custom resource is now created when you initially install the Speech services. The process is described in the IBM Cloud Pak for Data installation documentation. The content of the custom resource has changed:

- The recommended name of the custom resource has changed from `speech-prod-cr` to `speech-cr`.
- All references to storage class have changed from variants of `storageClass` to `blockStorageClass`.
- The name of the Portworx block storage class has changed from `portworx-shared-gp3` to `portworx-db-gp3-sc`.
- The `createSecret` property has been removed for the MinIO and PostgreSQl datastores. The property is only used internally. The Speech services always use a secrets object if you create one, and they always automatically create the object if none is provided.

User-provided secrets object now supported for RabbitMQ datastore

You can now provide security credentials for the RabbitMQ datastore, just as you can for the MinIO and PostgreSQL datastores. The documented process is similar for all three datastores.

Defect fix: Multiple consecutive SSML `<phoneme>` tags are now parsed correctly

Defect fix: The service now correctly synthesizes text that contains consecutive `<phoneme>` tags. Previously, if the text contained two or more consecutive `<phoneme>` tags, the service synthesized only the first tag, ignoring the others.

Security vulnerabilities addressed

No security vulnerabilities were fixed for version 4.5.0.

25 May 2022 (Version 4.0.9)

Version 4.0.9 is now available

Text to Speech for IBM Cloud Pak for Data version 4.0.9 is now available. This version supports IBM Cloud Pak for Data version 4.x and Red Hat OpenShift versions 4.6 and 4.8. For more information about installing and managing the service, see Installing Watson Text to Speech.

New support for `audio/alaw` audio format

The list of supported audio formats now includes `audio/alaw;rate={rate}`. Like `audio/basic` and `audio/mulaw`, this format provides single-channel audio that is encoded by using 8-bit u-law (or mu-law) data that is sampled at 8 kHz. For more information, see Using audio formats.

The Speech services do not support the OADP backup and restore utility

Watson Speech services do not support the IBM Cloud Pak for Data OpenShift APIs for Data Protection (OADP) backup and restore utility. If the Speech services are installed on a cluster, you might not be able to use the IBM Cloud Pak for Data OADP backup and restore utility to back up other services that are installed on that cluster. This limitation applies to version 4.0.0 and later versions of the Speech services.

Security vulnerabilities addressed

The following security vulnerabilities have been fixed:

- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable a denial of service, caused by a buffer overflow with Twisted (CVE-2022-21716)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service in NumPy. (CVE-2021-33430)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a denial of service, caused by improper input validation with Spring Framework (CVE-2022-22950)

1 May 2022 (Version 1.2.x)

Important: End of service for Text to Speech version 1.2.x on IBM Cloud Pak for Data version 3.5

Important: Text to Speech version 1.2.x on IBM Cloud Pak for Data version 3.5 is out of service as of 1 May 2022. Text to Speech version 1.2.x is no longer supported, available, or documented. For more information about End of Service for Text to Speech, which is part of the Watson API Kit, see Software support discontinuance: IBM Watson API Kit for IBM Cloud Pak for Data 1.2.x .

27 April 2022 (Version 4.0.8)

Version 4.0.8 is now available

Text to Speech for IBM Cloud Pak for Data version 4.0.8 is now available. This version supports IBM Cloud Pak for Data version 4.x and Red Hat OpenShift versions 4.6 and 4.8. For more information about installing and managing the service, see Installing Watson Text to Speech.

New environment variables used in IBM Cloud Pak for Data documentation

Most commands in the Text to Speech for IBM Cloud Pak for Data documentation have been updated to use a common set of environment variables. The documentation provides a script to automatically export the environment variables before you run installation, upgrade, and administration commands. After you source the script, you can copy most commands from the documentation and run them without making any changes.

The environment variables that the script defines include the following:

- `${PROJECT_CPD_INSTANCE}` identifies the project where you plan to install IBM Cloud Pak for Data and the Speech services.
- `${PROJECT_CPD_OPS}` identifies the project for the IBM Cloud Pak for Data platform operator.
- `${PROJECT_CPFS_OPS}` identifies the project for the IBM Cloud Pak for Data foundational services.

For more information about using the environment variables, see Best practice: Setting up install variables .

The `ttsVoiceMarginalCPU` property is no longer documented

The `ttsVoiceMarginalCPU` property has been removed from the documentation for the Speech services custom resource. The property manages the tradeoff between concurrency and speech synthesis speed. The default value of `400` ensures a reasonable balance for most customers and maintains real-time synthesis.

Security vulnerabilities addressed

The following security vulnerabilities have been fixed:

- Security Bulletin: A vulnerability with Guava affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2020-8908)
- Security Bulletin: A Google Guava vulnerability affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2018-10237)
- Security Bulletin: Vulnerabilities in Apache Tomcat affect IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2022-23181)
- Security Bulletin: A Cyrus SASL vulnerability affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2022-24407)

- Security Bulletin: A vulnerability with GNU wget affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2016-4971)
- Security Bulletin: A vulnerability with GNU Wget affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2018-0494)
- Security Bulletin: A vulnerability in 'GNU Wget' affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2018-20483)
- Security Bulletin: A vulnerability in ISC BIND affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2018-5741)
- Security Bulletin: A vulnerability in Python affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2019-20916)
- Security Bulletin: A vulnerability with ISC BIND affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2021-25214)
- Security Bulletin: A vulnerability in ISC BIND affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2021-25215)
- Security Bulletin: A vulnerability in ISC BIND affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2021-25216)
- Security Bulletin: A vulnerability in ISC BIND affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2021-25219)
- Security Bulletin: A vulnerability in PostgreSQL JDBC Driver (PgJDBC) affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2022-21724)
- Security Bulletin: A vulnerability in GNU Tar affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2019-9923)
- Security Bulletin: A vulnerability in logback-classic affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2021-42550)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a stack-based buffer overflow in GNU C Library (CVE-2022-23218)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to stack-based buffer overflow in GNU C Library (CVE-2022-23219)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to a buffer overflow and underflow in GNU C Library (CVE-2021-3999)

30 March 2022 (Version 4.0.7)

Version 4.0.7 is now available

Text to Speech for IBM Cloud Pak for Data version 4.0.7 is now available. This version supports IBM Cloud Pak for Data version 4.x and Red Hat OpenShift versions 4.6 and 4.8. For more information about installing and managing the service, see Installing Watson Text to Speech.

Custom resource property for specifying a default voice

The default voice for speech synthesis and pronunciation requests is `en-US_MichaelV3Voice`. If you do not install the `en-US_MichaelV3Voice`, you must either

- Use the `voice` parameter to pass the voice that is to be used with each request.
- Specify a new default voice for your installation of Text to Speech for IBM Cloud Pak for Data by using the `defaultTTSVoice` property in the Speech services custom resource. For more information, see Installing Watson Text to Speech and Using the default voice.

Change to word timing response for WebSocket interface

The response object that the service sends when you request word timings with the WebSocket interface has changed. The service now sends word timing results in a single array that includes a string followed by two floats:

```
{
  "words": [
    ["Hello", 0.0, 0.259],
    ["world", 0.259, 0.532]
  ]
}
```

The service previously sent timing results as an array that included a string following by an array of two floats:

```
{
  "words": [
    ["Hello", [0.0629826778195474, 0.2590192737303819]],
    ["world", [0.2598829173456253, 0.5322130804452672]]
  ]
}
```

Also, the level of precision for word timings and marks is now reduced to three decimal places. For more information about the new responses, see

Generating word timings.

Security vulnerabilities addressed

The following security vulnerabilities have been fixed:

- Red Hat CVE-2022-24407: A flaw was found in the SQL plugin shipped with Cyrus SASL. The vulnerability occurs due to failure to properly escape SQL input and leads to an improper input validation vulnerability. This flaw allows an attacker to execute arbitrary SQL commands and the ability to change the passwords for other accounts allowing escalation of privileges.
- Security Bulletin: A jwt-go vulnerability affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2020-26160)
- Security Bulletin: A vulnerability in Golang Go affects IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2021-29923)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is affected but not classified as vulnerable by a remote code execution in Spring Framework (CVE-2022-22965)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to arbitrary code execution with IBM WebSphere Application Server (CVE-2021-23450)

23 February 2022 (Version 4.0.6)

Version 4.0.6 is now available

Text to Speech for IBM Cloud Pak for Data version 4.0.6 is now available. This version supports IBM Cloud Pak for Data version 4.x and Red Hat OpenShift versions 4.6 and 4.8. For more information about installing and managing the service, see Installing Watson Text to Speech.

All neural voices are now deprecated for IBM Cloud Pak for Data

The neural voices that were available with Text to Speech for IBM Cloud Pak for Data are now deprecated. The neural voices continue to be available to users of Text to Speech for IBM Cloud. Only the enhanced neural voices continue to be available to users of Text to Speech for IBM Cloud Pak for Data.

All voices for the following languages are now deprecated for IBM Cloud Pak for Data:

- Arabic
- Chinese (Mandarin)
- Czech
- Dutch (Belgian)
- Dutch (Netherlands)
- English (Australian)
- Korean
- Swedish

Existing users of these voices can continue to use them for now, but the voices will be removed entirely in a future release. These voices can no longer be installed by new users and have been removed from the installation documentation for IBM Cloud Pak for Data. The `voiceType` property has been removed from the Speech services custom resource.

For more information, see

- Languages and voices
- Installing Watson Text to Speech

Updates to import/export scripts

The `import_export.sh` and `transfer_ownership.sh` scripts have been updated. These scripts are used to import and export data between clusters, back up and restore data, and migrate data from version 3.5 to version 4.0.x. The scripts have been modified and improved as follows:

- The `transfer_ownership.sh` script now requires a `-c` option to be included on the command line before the `<custom_resource_name>` argument.
- The `transfer_ownership.sh` script now requires a `-v <version>` option and argument to indicate the version to which ownership of resources is being transferred. Specify `35` for version 3.5 or `40` for version 4.0.x.
- The `transfer_ownership.sh` script now requires a `-p` option to be included on the command line before the `<postgres_auth_secret_name>` argument.
- The `<postgres_auth_secret_name>` argument provides the Kubernetes secret that is used to authenticate to the PostgreSQL datastore to which you are transferring ownership. You can omit the authentication secret if is the same as the default value (`<custom-resource-name>-postgres-auth-`

`secret` for version 4.0.x, `user-provided-postgressql` for version 3.5). You must provide the secret if it is different from the default value.

- Both scripts now include a `-h` (`--help`) option to display information about the script and its usage.

For more information, see

- Administering Watson Text to Speech, specifically *Importing and exporting data* and *Backing up and restoring data* .
- Upgrading Watson Text to Speech, specifically *Migrating data from IBM Cloud Pak for Data Version 3.5* .

Updated recommendation for OpenShift Container Storage

Starting with Speech services version 4.0.6, the recommended storage class for OpenShift Container Storage is `ocs-storagecluster-ceph-rbd`.

- If you are installing Speech services 4.0.6 or upgrading to Speech services 4.0.6 from IBM Cloud Pak for Data version 3.5, specify the `ocs-storagecluster-ceph-rbd` storage class during installation or upgrade.
- If you are upgrading to Speech services 4.0.6 from a previous refresh of Cloud Pak for Data version 4.0, continue to use `ocs-storagecluster-cephfs`. You cannot change the storage that is used in an existing deployment.

The value is specified with the `storageClass` property in the Speech services custom resource:

```
################
# Storage class
################
storageClass: "ocs-storagecluster-ceph-rbd"
```

The Speech services work with either version of OpenShift Container Storage. The newly recommended version has more restrictive access permissions. For more information, see

- Installing Watson Text to Speech
- Upgrading Watson Text to Speech

31 January 2022 (Version 4.0.5)

Version 4.0.5 has been updated

Text to Speech for IBM Cloud Pak for Data version 4.0.5 has been updated to address installation issues. The case package version is now 4.0.6. Use this package instead of the version 4.0.5 package. For more information about installing and managing the service, see Installing Watson Text to Speech.

Important: Extra steps for mirrored installation are no longer necessary

Important: The 26 January 2022 release notes included important notes for the following steps:

- Additional step for performing a mirrored installation of Minio datastore
- Additional steps for performing a mirrored installation of new next-generation models

These additional steps are no longer needed. The case package has been updated to correct the installation issues.

26 January 2022 (Version 4.0.5)

Version 4.0.5 is now available

Text to Speech for IBM Cloud Pak for Data version 4.0.5 is now available. This version supports IBM Cloud Pak for Data version 4.x and Red Hat OpenShift versions 4.6 and 4.8. For more information about installing and managing the service, see Installing Watson Text to Speech.

Important: Additional step for performing a mirrored installation of Minio datastore

Important: These steps are no longer needed if you install case package 4.0.6. For more information, see 31 January 2022 (Version 4.0.5).

If you are performing a mirrored installation (for example, in an air-gapped environment), you need to perform an additional step *before* completing either of the following steps:

- Step 7 Mirroring the images to the private container registry of *Mirroring images with a bastion model*

- Step 8 Mirroring the images to the intermediary container registry of *Mirroring images with an intermediary container registry*

This step is mandatory to copy the necessary images for the Minio datastore:

```
echo 'cp.icr.io,cp/opencontent-minio-
client,1.1.4,sha256:7b4cf5e47a0455cfa7ca9ab246b80916e4dccbc1483b3e0f276fb7b0ab3e5c60,IMAGE,linux,x86_64,"",0,CASE,"","""" \
>> $CASE_PATH/ibm-watson-speech-4.0.5-images.csv
```

Failure to perform this step will cause installation errors for both Text to Speech and Speech to Text.

License Server is now automatically installed

The Speech services operator now automatically installs the required License Server when it installs the Speech services. You no longer need to install the License Server from the IBM Cloud Pak for Data foundational services, and you no longer need to use additional YAML content to create an OperandRequest with the necessary bindings.

Removal of steps specific to PostgreSQL EnterpriseDB server

The previous version of the documentation included steps for the PostgreSQL EnterpriseDB server that were specific to the Speech services. These steps were documented in the topics *Upgrading Watson Text to Speech (Version 4.0)* and *Uninstalling Watson Text to Speech* . These additional steps are no longer necessary and have been removed from the documentation.

RabbitMQ datastore is now used only by the `sttAysnc` component

The RabbitMQ datastore was previously used by components of both Speech services, Speech to Text and Text to Speech. It now handles non-persistent message queuing for the Speech to Text asynchronous HTTP component (`sttAsync`) only. It is used only if the `sttAsync` component is installed and enabled.

New Belgian Dutch and Czech neural voices

Two new neural voices are now available:

- *Belgian Dutch:* A new male Belgian Dutch (Flemish) voice, `nl-BE_BramVoice` .
- *Czech:* A new language, Czech, with a new female voice, `cs-CZ_AlenaVoice` .

You can install the new voices along with all neural voices by setting the `voiceType` property of the custom resource to `neuralVoices` .

- For more information about using the custom resource to install voices, see Installing Watson Text to Speech.
- For more information about all available languages and voices, see Languages and voices.

Defect fix: Update SSML documentation

Defect fix: The SSML documentation was updated to correct the following errors:

- The examples of the `<break>` element are now correct. The element is unary, as now shown in the examples. The previous examples included open and close tags with embedded text. The embedded text was not spoken by the service. For more information, see The `<break>` element.
- The service supports Speech Synthesis Markup Language (SSML) version 1.1. All references and examples now use the correct version. The documentation previously referred to version 1.0.

Security vulnerabilities addressed

The following security vulnerabilities associated with Apache Log4j have been fixed:

- Security Bulletin: Vulnerability in Apache Log4j may affect IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2021-4104)
- Security Bulletin: IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data is vulnerable to denial of service and arbitrary code execution due to Apache Log4j (CVE-2021-45105 and CVE-2021-45046)

20 December 2021 (Version 4.0.4)

Version 4.0.4 is now available

Text to Speech for IBM Cloud Pak for Data version 4.0.4 is now available. This version supports IBM Cloud Pak for Data version 4.x and Red Hat OpenShift versions 4.6 and 4.8. For more information about installing and managing the service, see Installing Watson Text to Speech.

Important: Changes to properties for disabling the storage and logging of user data

Important: The names of the properties of the Speech services custom resource that specify whether user data is stored and logged have changed. The custom resource formerly contained the following properties:

```
#################
# Anonymize logs
#################
  sttRuntime:
    anonymizeLogs: "false"  # If true, disables storage and logging of user data
  sttAMPatcher:
    anonymizeLogs: "false"  # If true, disables storage and logging of user data
  ttsRuntime:
    anonymizeLogs: "false"  # If true, disables storage and logging of user data
```

These properties are now named as follows:

```
###################################
# Storage and logging of user data
###################################
  sttRuntime:
    skipAudioAndResultLogging: "false"  # If true, disables storage and logging of user data
  sttAMPatcher:
    skipAudioAndResultLogging: "false"  # If true, disables storage and logging of user data
  ttsRuntime:
    skipAudioAndResultLogging: "false"  # If true, disables storage and logging of user data
```

If you already set these properties in your custom resource to change the default value of `false` to `true`, you need to edit your custom resource. You must manually change the names of the properties to the new values and save the updated custom resource. For more information, see Installing Watson Text to Speech.

Important: Changes to properties of PostgreSQL secrets object

Important: When you install the Speech services, an object that contains a randomly generated password for the PostgreSQL datastore is created by default. You can choose instead to specify the password manually. If you do, the properties of the YAML file for the secrets object have changed. For more information, see the topic about managing your datastores in Administering Watson Text to Speech.

Important: PostgreSQL pods do not start with EnterpriseDB version 1.10 operator

Important: With Text to Speech for IBM Cloud Pak for Data version 4.0.3, PostgreSQL pods based on the EnterpriseDB version 1.10 operator can fail to start. This prevents the Speech services from starting. A workaround exists for this problem. If your Speech services fail to start, see PostgreSQL pods do not start with EnterpriseDB version 1.10 operator for information about diagnosing and resolving the problem.

This problem is fixed in Text to Speech for IBM Cloud Pak for Data version 4.0.4.

New support for IBM Spectrum Scale Container Native storage class

Since version 4.0.3, the Speech services support the IBM Spectrum® Scale Container Native storage class. To use IBM Spectrum Scale, specify `"ibm-spectrum-scale-sc"` for the `storageClass` property of the Speech services custom resource. For more information, see Installing Watson Text to Speech.

Interaction of Speech services with MinIO datastore during installation

The Speech services runtime components, `sttRuntime` and `ttsRuntime`, cannot start until the models and voices for the services are fully uploaded into the MinIO datastore. During installation, the services might fail and automatically restart themselves one or more times until upload of the models and voices is complete. They then start properly. No user action is required.

Defect fix: Improve upgrade documentation

Defect fix: Documentation for upgrading the Speech services to new versions of IBM Cloud Pak for Data version 4.0.x included incorrect references in some commands. These references are now correct:

- The strings `watsonSpeechToTextStatus` and `watsonTextToSpeechStatus` have been changed to `speechStatus` in both cases.
- The strings `status.watsonSpeechToTextVersion` and `status.watsonTextToSpeechVersion` have been changed to `.spec.version` in both cases.

For more information, see Upgrading Watson Text to Speech.

Defect fix: Improve SSML and speech synthesis

Defect fix: The following defects for the Speech Synthesis Markup Language (SSML) and speech synthesis were fixed with this release:

- The `pitch` attribute of the `<prosody>` element is now applied to all specified text. Previously, the pitch change was not always applied to the

first word of the affected text. Also, the documentation now includes additional guidance about specifying a `pitch` value. For more information, see The `pitch` attribute.

- Speech synthesis of Japanese text now speaks the audio more slowly. Previously, the synthesized speech was being spoken too quickly. If you find that synthesis of Japanese text is still spoken too quickly for your application, use the `rate` attribute of the SSML `<prosody>` element to control the rate of speech. For more information, see The `rate` attribute.
- Neural voices now parse the escaped apostrophe character (`&apos;`) properly. Previously, some neural voices were not interpreting the character properly.

Security vulnerability addressed

The following security vulnerability associated with Apache Log4j has been fixed:

- Security Bulletin: Vulnerability in Apache Log4j may affect IBM Watson Speech Services Cartridge for IBM Cloud Pak for Data (CVE-2021-4428)

20 December 2021 (Version 1.2.x)

Important: You can no longer install Text to Speech version 1.2.x on IBM Cloud Pak for Data version 3.5

Important: You can no longer perform new installations of Text to Speech version 1.2.x on IBM Cloud Pak for Data version 3.5. You can install only Text to Speech version 4.0.x on IBM Cloud Pak for Data version 4.x. For more information, see Installing Watson Text to Speech.

The Speech services for IBM Cloud Pak for Data version 3.5 reach their End of Support date on 30 April 2022. You are encouraged to upgrade to the latest version 4.0.x release of the services at your earliest convenience. For more information, see Upgrading Watson Text to Speech.

30 November 2021 (Version 4.0.3)

Version 4.0.3 is now available

Text to Speech for IBM Cloud Pak for Data version 4.0.3 is now available. This version supports IBM Cloud Pak for Data version 4.x and Red Hat OpenShift versions 4.6 and 4.8. For more information about installing and managing the service, see Installing Watson Text to Speech.

License Server now a mandatory prerequisite

You must now install the License Server from the IBM Cloud Pak for Data foundational services. You must install the License Server by using the YAML content that is provided to create an OperandRequest with the necessary bindings. You must also install the License Service in the same namespace as the service (operand), which is also where IBM Cloud Pak for Data is installed. For more information, see Installing Watson Text to Speech.

New support for in-place upgrade

The service now supports in-place, operator-based upgrade from version 4.0.0 to version 4.0.3. Moving from IBM Cloud Pak for Data version 3.5 to version 4.0.3 continues to require use of migration utilities. For more information, see Upgrading Watson Text to Speech.

EDB PostgreSQL operator and license installation changes

Installation, upgrade, and uninstallation for the Enterprise DB PostgreSQL operator and license have changed:

- Instructions for installing the EDB PostgreSQL operator and license are now included with the IBM Cloud Pak for Data foundational services. The instructions for installing the Speech services have been updated accordingly. For more information, see Installing Watson Text to Speech.
- Instructions for upgrading from Text to Speech version 4.0.0 to 4.0.3 include instructions for uninstalling the previous EDB PostgreSQL operator and license and reinstalling them with the IBM Cloud Pak for Data foundational services. For more information, see Upgrading Watson Text to Speech.
- Instructions for uninstalling the Speech services now include steps for removing the EDB PostgreSQL operator and license that were previously installed with Text to Speech. For more information, see Uninstalling Watson Text to Speech.

New guidance for scaling up your installation

The service now provides updated guidance about scaling up your installation. The information includes specifying the number of pods and the maximum number of concurrent sessions for enhanced neural or neural voices. For more information, see Administering Watson Text to Speech.

Command-line updates to import and export utilities

The commands that are used with the import and export utilities for the Speech services include new options and arguments. The import and export utilities are also the foundation for backing up and restoring the services and for migrating from IBM Cloud Pak for Data version 3.5 to version 4.0.3. For more information about using the utilities, see

- Administering Watson Text to Speech
- Upgrading Watson Text to Speech

New property for managing concurrency and speech synthesis

The new `global.ttsVoiceMarginalCPU` property manages the tradeoff between concurrency and speech synthesis speed. The default value of 400 offers a reasonable balance for most customers and maintains real-time synthesis. For information about modifying this value to suit your needs, contact IBM Support.

New support for neural voices

All neural voices that are currently available for Text to Speech for IBM Cloud are now also available for installation on Text to Speech for IBM Cloud Pak for Data. The following languages and voices are now available:

- *Arabic:* `ar-MS_OmarVoice`
- *Chinese (Mandarin):* `zh-CN_LiNaVoice`, `zh-CN_WangWeiVoice`, and `zh-CN_ZhangJingVoice`
- *Dutch (Belgian):* `nl-BE_AdeleVoice`
- *Dutch (Netherlands):* `nl-NL_EmmaVoice` and `nl-NL_LiamVoice`
- *English (Australian):* `en-AU_CraigVoice`, `en-AU_MadisonVoice`, and `en-AU_SteveVoice`
- *Korean:* `ko-KR_HyunjunVoice`, `ko-KR_SiWooVoice`, `ko-KR_YoungmiVoice`, and `ko-KR_YunaVoice`
- *Swedish:* `sv-SE_IngridVoice`

For more information about all available languages and voices, see Languages and voices.

Installing voices

You can install either the enhanced neural voices or the neural voices. You can install only one of the two types of voices. When you install the service, you use the `voiceType` property of the custom resource to indicate the voices that are to be installed:

- Specify `enhancedNeuralVoices` to install the enhanced neural voices. You must then specify the individual enhanced neural voices that are to be installed. By default, only `en-US_AllisonV3Voice`, `en-US_LisaV3Voice`, and `en-US_MichaelV3Voice` are installed. You can choose to install these default voices, these and other voices, or just other voices. Only the voices that you install are available.
- Specify `neuralVoices` to install the neural voices. All of the neural voices are installed and available. You cannot refine the list of installed voices.

For more information about using the custom resource to install voices, see Installing Watson Text to Speech.

Specifying a voice for speech synthesis

Both the HTTP `POST` and `GET /v1/synthesize` methods, as well as the WebSocket `/v1/synthesize` method, accept an optional `voice` query parameter that you use to specify the voice that is to be used for speech synthesis. If you omit the `voice` parameter, the service uses a default voice. The default voice depends on the voices that you installed:

- *If you installed the enhanced neural voices,* the service uses the US English `en-US_MichaelV3Voice` by default. If that voice is not installed, you must specify a voice.
- *If you installed the neural voices,* the service always uses the Australian English `en-AU_MadisonVoice` by default.

For more information, see Using a voice for speech synthesis.

Specifying a language for a custom model

You use the `POST /v1/customizations` method to create a custom model. The method includes a `language` parameter that you use to identify the language of the new custom model.

- *If you installed the enhanced neural voices,* the `language` parameter is optional. By default, the service uses the `en-US` identifier for the language.
- *If you installed the neural voices,* the `language` parameter is required. You must specify the language for the custom model in the indicated format (for example, `en-AU` for Australian English).

For more information about specifying a language when you create a custom model, see Creating a custom model.

Defect fix: Correct intonation for Spanish enhanced neural voices

Defect fix: For the Castilian Spanish (`es-ES_EnriqueV3Voice` and `es-ES_LauraV3Voice`), Latin American Spanish (`es-LA_SofiaV3Voice`), and North American Spanish (`es-US_SofiaV3Voice`) voices, questions of all types now use the correct intonation. The voices previously did not use the correct intonation for some questions, instead pronouncing them like statements.

Defect fix: Correct multitenancy documentation

Defect fix: The IBM Cloud Pak for Data topic Multitenancy support incorrectly stated that the Speech services do not support multitenancy. The topic has been updated to state that the Speech services support the following operations:

- Install the service in separate projects
- Install the service multiple times in the same project
- Install the service once and deploy multiple instances in the same project

The documentation that is specific to the Speech services correctly stated the multitenancy support.

1 October 2021 (Version 1.1.x)

Version 1.1.x is out of service

Text to Speech and Speech to Text for IBM Cloud Pak for Data version 1.1.x went out of service on 30 September 2021. As of 1 October 2021, the documentation for version 1.1.x is no longer available. For more information, see Software withdrawal and support discontinuance.

29 July 2021 (Version 4.0.0)

Version 4.0.0 is available

IBM Watson® Text to Speech for IBM Cloud Pak® for Data version 4.0.0 is now available. Installation and administration of the service include many changes. This version supports IBM Cloud Pak for Data version 4.x and Red Hat OpenShift version 4.6. For more information about installing and managing the service, see Installing IBM Watson Text to Speech for IBM Cloud Pak for Data .

Enhanced neural voices

To optimize the overall quality of voice synthesis, all available voices are now *enhanced neural voices*. Enhanced neural voices, which include the string `V3` in their names, are now available for Brazilian Portuguese, United Kingdom and United States English, French, German, Italian, Japanese, and Spanish (all dialects).

Enhanced neural voices support the use of both IPA and IBM Symbolic Phonetic Representation (SPR) with the SSML `<phoneme>` element. Enhanced neural voices also achieve a slightly higher degree of natural-sounding speech. For more information, see Languages and voices.

New Canadian French voice

The service now supports Canadian French with the enhanced neural voice `fr-CA_LouiseV3Voice` . The Canadian French voice supports customization and is generally available (GA) for production use.

- To hear a sample of the new voice, see Supported languages and voices .
- For more information about the phonetic symbols and Unicode values that are available for the Canadian French language, see French (Canadian) symbols.

New Tune by Example feature

The new Tune by Example feature lets you control how specified text is spoken by the service. The feature is beta functionality that is supported only for US English custom models and voices. The feature has two components:

- *Custom prompts* include the written text that is to be spoken and recorded audio that speaks the text as you want to hear it. The audio specifies the intonation, cadence, and stress of the synthesized text. The prompt can emphasize different syllables or words, introduce pauses, and generally make the synthesized audio sound more natural and appropriate for its context.
- *Speaker models* provide enrollment audio for a user who speaks one or more prompts. A speaker model provides an audio sample of a user's voice. The service trains itself on the voice, which can help it to produce higher-quality prompts for that speaker.

You specify a custom prompt with a speech synthesis request to indicate how the service's voice is to pronounce the text. To specify a prompt, you use the SSML extension `<ibm:prompt id="{prompt_id}"/>` . The synthesized audio duplicates the prosody of the prompt.

The service includes eight new methods for working with the Tune by Example feature. The descriptions of the new methods that follow provide links to their entries in the API & SDK reference.

- The service includes four methods for working with custom prompts:
  - List custom prompts: `GET /v1/customizations/{customization_id}/prompts`
  - Get a custom prompt: `GET /v1/customizations/{customization_id}/prompts/{prompt_id}`
  - Delete a custom prompt: `DELETE /v1/customizations/{customization_id}/prompts/{prompt_id}`
- The service includes four methods for working with speaker models:
  - List speaker models: `GET /v1/speakers`
  - Get a speaker model: `GET /v1/speakers/{speaker_id}`
  - Delete a speaker model: `DELETE /v1/speakers/{speaker_id}`

Unified Text to Speech documentation

The documentation for IBM Watson Text to Speech for IBM Cloud Pak for Data is now combined with the documentation for managed instances of the Text to Speech service that are hosted on IBM Cloud. This is true of both the guide and reference documentation for the two forms of the service. Links to the formerly separate version of the IBM Cloud Pak for Data documentation for the service redirect to the unified documentation.

For more information about identifying information that pertains to only one version of the product, see About Text to Speech.

Version 1.1.x is going out of service

Speech to Text and Text to Speech for IBM Cloud Pak for Data version 1.1.x go out of service on **30 September 2021**. You must upgrade to a later version of the services on IBM Cloud Pak for Data before that date. As of 1 October 2021, the documentation for version 1.1.4 will no longer be available.

12 April 2021 (Version 1.2.1)

Addition to `speech-override.yaml` file

The minimal `speech-override.yaml` file includes an extra definition, `dockerRegistryPrefix`:

```
global:
  dockerRegistryPrefix: "{Registry}"
  image:
    pullSecret: "{Registry_pull_secret}"
```

`{Registry}` is the path for the internal Docker registry. It must be `image-registry.openshift-image-registry.svc:5000/{namespace}`, where `{namespace}` is the namespace in which IBM Cloud Pak® for Data is installed, normally `zen`.

9 April 2021 (Version 1.2.1)

Support for modifying installed models and voices

The Speech services let you add or remove installed models and voices for version 1.2 or 1.2.1 of the services.

26 March 2021 (Version 1.2.1)

Version 1.2.1 is available

Text to Speech for IBM Cloud Pak for Data version 1.2.1 is now available. Versions 1.2 and 1.2.1 use the same version 1.2 documentation and installation instructions. Version 1.2.1 supports installation on Red Hat OpenShift version 4.6 in addition to versions 4.5 and 3.11.

New installation instructions

For both clusters connected to the internet and air-gapped clusters, the installation instructions include the following steps:

- Use the `oc label` command to set up required labels for the namespace where IBM Cloud Pak for Data is installed.
- Use the `oc project` command to ensure that you are pointing at the correct OpenShift project.
- Use the `cpd-cli install` command to install an Enterprise DB PostgreSQL server that is used by the Speech services.

You perform these steps before you install the Speech services.

New uninstallation instructions

A step was added to the procedure for uninstalling the Speech services to clean up all of the resources from the installation.

Entitled registry for PostgreSQL datastore

The entitled registry path from which the service pulls images for the PostgreSQL datastore has changed. The registry location changed from `cp.icr.io/cp/watson-speech` to `cp.icr.io/cp/cpd`. This change is transparent to users.

Secrets for Minio and PostgreSQL datastores

The Minio and PostgreSQL datastores require the following hard-coded values for their secrets:

- For *Minio*, use `minio`.
- For *PostgreSQL*, use `user-provided-postgressql`.

You cannot use your own values for these secrets. The secrets must be created before you install the Speech services.

Deletions from `speech-override.yaml` file

The following entries have been removed from the `speech-override.yaml` file. They were added to work around a problem that has now been fixed.

```
sttRuntime:
  images:
    miniomc:
      tag:
        1.0.5
sttAMPatcher:
  images:
    miniomc:
      tag:
        1.0.5
ttsRuntime:
  images:
    miniomc:
      tag:
        1.0.5
```

The abbreviated `speech-override.yaml` file has generally been reduced further by fine-tuning its contents to the essential elements.

9 December 2020 (Version 1.2)

Version 1.2 is available

Text to Speech for IBM Cloud Pak for Data version 1.2 is now available. Installation and administration of the service include many changes. This version supports IBM Cloud Pak for Data versions 3.5 and 3.0.1, and Red Hat OpenShift versions 4.5 and 3.11.

New voices

The service now offers two new voices:

- UK English: `en-GB_CharlotteV3Voice`
- French: `fr-FR_NicolasV3Voice`

The service also offers an improved version of the existing UK voice, `en-KateV3Voice`. For more information about all supported languages and voices, see Languages and voices.

Defect fix: Fix `<prosody>` element for Japanese

Defect fix: For the `ja-JP_EmiV3Voice` voice, the service now correctly parses SSML input text that includes a prosody rate specification. Previously, the following use of the `<prosody>` element worked properly:

```
<speak>成功する/繁栄する</speak>
```

But the following use of the rate attribute with the `<prosody>` element caused the service to read and speak the embedded SSML notation:

```
<speak>
 <prosody rate="fast">成功する/繁栄する</prosody>
</speak>
```

The service now correctly parses and applies the `rate` attribute of the `<prosody>` element for Japanese input.

4 September 2020 (Version 1.1.4)

Customization interface is generally available

The customization interface is now generally available. Customization is no longer beta functionality. You can use the customization interface to specify how the service pronounces unusual words that occur in your input text by creating language-specific custom dictionaries. For more information, see Understanding customization.

15 July 2020 (Version 1.1.4)

Red Hat OpenShift version 4.3 is going out of service

IBM Cloud Pak for Data 3.0.1 is deprecating support for Red Hat OpenShift 4.3 on 1 September 2020. Red Hat OpenShift 4.3 is going out of service on **22 October 2020**. IBM Cloud Pak for Data is introducing support for Red Hat OpenShift 4.5. IBM Cloud Pak for Data is recommending that clients upgrade to Red Hat OpenShift 4.5 before 22 October 2020. IBM Support will work with any customers who already installed IBM Cloud Pak for Data 3.0.1 on Red Hat OpenShift 4.3. New customers who want to install on Red Hat OpenShift 4.x are instructed to install Red Hat OpenShift 4.5.

19 June 2020 (Version 1.1.4)

Version 1.1.4 is available

Text to Speech for IBM Cloud Pak for Data version 1.1.4 is now available. Installation and administration of the service include many changes. This version supports IBM Cloud Pak for Data versions 2.5 and 3.0.1, and Red Hat OpenShift versions 3.11 and 4.3. For more information about installing and managing the service, see Installing and managing Text to Speech for IBM Cloud Pak for Data .

New neural voices

The service now supports five new neural voices:

- US English: `en-US_EmilyV3Voice`, `en-US_HenryV3Voice`, `en-US_KevinV3Voice`, and `en-US_OliviaV3Voice`
- German: `de-DE_ErikaV3Voice`

These new voices have the same capabilities for customization and SSML as all existing voices. For more information, see Supported languages and voices.

Support for SSML `digits` attribute of `<say-as>` element for Japanese

The service now supports the `digits` attribute of the SSML `<say-as>` element with its Japanese voice. For more information, see The `<say-as>` element.

Simplified backup and restore procedures

The backup and restore procedures are greatly simplified. They now back up data from the datastores, so you no longer need to re-create the operations you have run. For more information, see Backing up and restoring Watson Speech services data .

28 February 2020 (Version 1.1.3)

Version 1.1.3 is available

Text to Speech for IBM Cloud Pak for Data version 1.1.3 is now available.

27 November 2019 (Version 1.1.2)

Version 1.1.2 is available

Text to Speech for IBM Cloud Pak for Data version 1.1.2 is now available.

30 August 2019 (Version 1.0.1)

Version 1.0.1 is available

Text to Speech for IBM Cloud Pak for Data version 1.0.1 is now available. The service now works with IBM Cloud Pak for Data 2.1.0.1. The service now supports installing IBM Cloud Pak for Data with Red Hat OpenShift.

New Japanese neural voice

The service now offers the neural Japanese voice `ja-JP_EmiV3Voice`. For more information, see Supported languages and voices.

FISMA support

Federal Information Security Management Act (FISMA) support is now available for Text to Speech for IBM Cloud Pak for Data. The service is FISMA High Ready.

28 June 2019 (Version 1.0.0)

Version 1.0.0 is available

Version 1.0.0, the initial release of the service, is now available. Text to Speech for IBM Cloud Pak for Data is based on the IBM Watson® Text to Speech service on the public IBM Cloud. Text to Speech for IBM Cloud Pak for Data differs from the public Text to Speech service in the following ways. You might find this information helpful if you are already familiar with the Text to Speech service on the public IBM Cloud.

- Text to Speech for IBM Cloud Pak for Data uses access tokens for authentication. For more information, see the API & SDK reference.
- The endpoints for Text to Speech for IBM Cloud Pak for Data are specific to your IBM Cloud Pak for Data cluster. For more information, see the API & SDK reference.
- Text to Speech for IBM Cloud Pak for Data supports only neural voices. It does not support standard (concatenative) voices. The neural voices do not support the SSML `<express-as>` and `<voice-transformation>` elements.
- Text to Speech for IBM Cloud Pak for Data does not perform any request logging. You do not need to use the `X-Watson-Learning-Opt-Out` request header.
- Text to Speech for IBM Cloud Pak for Data does not support Watson tokens. You cannot use the `X-Watson-Authorization-Token` request header to authenticate with the service.

Release notes for IBM® Software Hub

Learn about features and changes that were included for each release and update of the product software.

IBM Software Hub

Note: This information applies only to instances of Watson Speech services that are installed on IBM® Software Hub. For information about

releases and updates for managed deployments, see Release notes for Text to Speech for IBM Cloud.

For the list of known issues, see Known issues and limitations for Watson Speech services.

27 February 2026 (Version 5.3.1)

Watson Speech services for IBM Software Hub is now available.

For a list of new features and fixes, see What's new and changed in Watson Speech services

16 December 2025 (Version 5.3.0)

Watson Speech services for IBM Software Hub is now available.

For a list of new features and fixes, see What's new and changed in Watson Speech services

29 October 2025 (Version 5.2.2)

Watson Speech services for IBM Software Hub is now available.

For a list of new features and fixes, see What's new and changed in Watson Speech services

27 August 2025 (Version 5.2.1)

Watson Speech services for IBM Software Hub is now available.

For a list of new features and fixes, see What's new and changed in Watson Speech services

11 June 2025 (Version 5.2.0)

Watson Speech services for IBM Software Hub is now available.

For a list of new features and fixes, see What's new and changed in Watson Speech services

30 April 2025 (Version 5.1.3)

Watson Speech services for IBM Software Hub is now available.

For a list of new features and fixes, see What's new and changed in Watson Speech services

26 March 2025 (Version 5.1.2)

Watson Speech services for IBM Software Hub is now available.

For a list of new features and fixes, see What's new and changed in Watson Speech services

26 February 2025 (Version 5.1.1)

Watson Speech services for IBM Software Hub is now available.

For a list of new features and fixes, see What's new and changed in Watson Speech services

11 December 2024 (Version 5.1.0)

Watson Speech services for IBM Software Hub is now available.

For a list of new features and fixes, see What's new and changed in Watson Speech services

Using languages and voices

Languages and voices

The IBM Watson® Text to Speech service supports a variety of languages, voices, and dialects. For different languages, the service offers female voices, male voices, or both. Each voice uses appropriate cadence and intonation for its dialect.

All of the service's voices use neural voice technology. Neural voice technology uses multiple Deep Neural Networks (DNNs) to predict the acoustic (spectral) features of the speech. The DNNs are trained on natural human speech and generate the resulting audio from the predicted acoustic features. During synthesis, the DNNs predict the pitch and phoneme duration (prosody), spectral structure, and waveform of the speech. Neural voices produce speech that is crisp and clear, with a very natural-sounding, smooth, and consistent audio quality.

Supported languages and voices

The service offers three types of voices with different qualities and capabilities:

- *Natural voices* provide advanced performance in terms of naturalness and expressiveness. These voices use various techniques to provide an edge over Expressive voices. For a list of all natural voices, see Natural voices.
- *Expressive neural voices* offer natural-sounding speech that is exceptionally clear and crisp. Their pronunciation and inflections are natural and conversational, and the resulting speech offers extremely smooth transitions between words. They also support the use of additional features that are not available with enhanced neural voices. For a list of all expressive voices, see Expressive neural voices.
- *Enhanced neural voices* achieve a high degree of natural-sounding speech and support most service features. For a list of all enhanced neural voices, see Enhanced neural voices.

The following pages provide more information about the voices and their technology:

- For a blog that introduces the expressive voices, see Is your conversational AI setting the right tone?.
- For more information about the service's neural voice technology, see The science behind the service.

Language support by type of voice

Table 1 shows the service's support for languages by type of voice. The following topics list the available languages and voices for each voice type.

Language	Natural voices	Expressive neural voices	Enhanced neural voices
Dutch (Netherlands)			✔
English (Canadian)	✔		
English (United Kingdom)	✔	✔	✔
English (Australian)	✔	✔	
English (United States)	✔	✔	✔
French (Canadian)			✔
French (France)			✔
German			✔
Italian			✔
Japanese			✔

Korean			✔
Portuguese (Brazilian)	✔	✔	✔
Spanish (Castilian)			✔
Spanish (Latin American)	✔	✔	✔
Spanish (South American)			✔

Language support by type of voice

Natural voices

Table 2 lists and provides audio samples for all available Natural voices. The *Availability* column indicates whether each voice is generally available (GA) for production use or beta. The column also indicates whether each voice is available for IBM Cloud, IBM Cloud Pak for Data, IBM Software Hub, or all 3 of them (no product version is cited).

Language	Availability	Voice / Gender	Audio sample
English (Australian)	GA	en-AU_HeidiNatural Female	Your browser does not support the audio tag.
English (Australian)	GA	en-AU_JackNatural Male	Your browser does not support the audio tag.
English (Canadian)	GA	en-CA_HannahNatural Female	Your browser does not support the audio tag.
English (United Kingdom)	GA	en-GB_ChloeNatural Female	Your browser does not support the audio tag.
English (United Kingdom)	GA	en-GB_GeorgeNatural Male	Your browser does not support the audio tag.
English (United States)	GA	en-US_EllieNatural Female	Your browser does not support the audio tag.
English (United States)	GA	en-US_EmmaNatural Female	Your browser does not support the audio tag.
English (United States)	GA	en-US_EthanNatural Male	Your browser does not support the audio tag.
English (United States)	GA	en-US_JacksonNatural Male	Your browser does not support the audio tag.
English (United States)	GA	en-US_VictoriaNatural Female	Your browser does not support the audio tag.
Portuguese (Brazilian)	GA	pt-BR_LucasNatural Male	Your browser does not support the audio tag.
Portuguese (Brazilian)	GA	pt-BR_CamilaNatural Female	Your browser does not support the audio tag.

Spanish (Latin American)	GA	es-LA_AlejandroNatural Male	Your browser does not support the audio tag.
Spanish (Latin American)	GA	es-LA_DanielaNatural Female	Your browser does not support the audio tag.

Natural languages and voices

Expressive neural voices

Table 3 lists and provides audio samples for all available expressive neural voices. The *Availability* column indicates whether each voice is generally available (GA) for production use or beta. The column also indicates whether each voice is available for IBM Cloud , IBM Cloud Pak for Data , IBM Software Hub , or all 3 of them (no product version is cited).

- Expressive neural voices support additional features that are not available with other types of voices. These features include additional speaking styles, automatic emphasis of interjections, and emphasis of specified words. For more information, see Modifying speech synthesis with expressive neural voices.
- When used with the SSML <prosody> element, expressive voices support only percentage values for the rate and pitch attributes. For more information, see The <prosody> element.

Expressive neural voices determine sentiment from context and automatically use the proper intonation to suit the text. To produce the most natural-sounding prosody, expressive neural voices need to consider the context of all words and phrases of a sentence. Expressive voices are therefore more compute-intensive and have slightly higher latency than other types of voices. The initial response for a synthesis request that uses an expressive voice might take a fraction of a second longer (for example, a few hundred milliseconds) to arrive. The total response time for the request to complete is also longer.

Tip: To minimize the latency and response time for an expressive voice, use shorter sentences wherever possible.

Language	Availability	Voice / Gender	Audio sample
English (Australian)	GA	en-AU_HeidiExpressive Female	Your browser does not support the audio tag.
English (Australian)	GA	en-AU_JackExpressive Male	Your browser does not support the audio tag.
English (United States)	GA	en-US_AllisonExpressive Female	Your browser does not support the audio tag.
English (United States)	GA	en-US_EmmaExpressive Female	Your browser does not support the audio tag.
English (United States)	GA	en-US_LisaExpressive Female	Your browser does not support the audio tag.
English (United States)	GA	en-US_MichaelExpressive Male	Your browser does not support the audio tag.
English (United Kingdom)	GA	en-GB_GeorgeExpressive Male	Your browser does not support the audio tag.
Portuguese (Brazilian)	GA	pt-BR_LucasExpressive Male	Your browser does not support the audio tag.
Spanish (Latin American)	GA	es-LA_DanielaExpressive Female	Your browser does not support the audio tag.

Expressive neural languages and voices

Enhanced neural voices

Table 4 lists and provides audio samples for all available enhanced neural voices. The *Availability* column indicates whether each voice is generally

available (GA) for production use or beta. The column also indicates whether each voice is available for IBM Cloud, IBM Cloud Pak for Data, IBM Software Hub or all 3 of them (no product version is cited).

Language	Availability	Voice / Gender	Audio sample
Dutch (Netherlands)	Beta	nl-NL_MerelV3Voice Female	Your browser does not support the audio tag.
English (United Kingdom)	GA	en-GB_CharlotteV3Voice Female	Your browser does not support the audio tag.
English (United Kingdom)	GA	en-GB_KateV3Voice Female	Your browser does not support the audio tag.
English (United States)	GA	en-US_AllisonV3Voice Female	Your browser does not support the audio tag.
English (United States)	GA	en-US_EmilyV3Voice Female	Your browser does not support the audio tag.
English (United States)	GA	en-US_HenryV3Voice Male	Your browser does not support the audio tag.
English (United States)	GA	en-US_KevinV3Voice Male	Your browser does not support the audio tag.
English (United States)	GA	en-US_LisaV3Voice Female	Your browser does not support the audio tag.
English (United States)	GA	en-US_MichaelV3Voice Male	Your browser does not support the audio tag.
English (United States)	GA	en-US_OliviaV3Voice Female	Your browser does not support the audio tag.
French (Canadian)	GA	fr-CA_LouiseV3Voice Female	Your browser does not support the audio tag.
French (France)	GA	fr-FR_NicolasV3Voice Male	Your browser does not support the audio tag.
French (France)	GA	fr-FR_ReneeV3Voice Female	Your browser does not support the audio tag.
German	GA	de-DE_BirgitV3Voice Female	Your browser does not support the audio tag.
German	GA	de-DE_DieterV3Voice Male	Your browser does not support the audio tag.
German	GA	de-DE_ErikaV3Voice Female	Your browser does not support the audio tag.
Italian	GA	it-IT_FrancescaV3Voice Female	Your browser does not support the audio tag.
Japanese	GA	ja-JP_EmiV3Voice Female	Your browser does not support the audio tag.

Korean	GA	`ko-KR_JinV3Voice` Female	Your browser does not support the audio tag.
Portuguese (Brazilian)	GA	`pt-BR_IsabelaV3Voice` Female	Your browser does not support the audio tag.
Spanish (Castilian)	GA	`es-ES_EnriqueV3Voice` Male	Your browser does not support the audio tag.
Spanish (Castilian)	GA	`es-ES_LauraV3Voice` Female	Your browser does not support the audio tag.
Spanish (Latin American)	GA	`es-LA_SofiaV3Voice` Female	Your browser does not support the audio tag.
Spanish (North American)	GA	`es-US_SofiaV3Voice` Female	Your browser does not support the audio tag.

Enhanced neural languages and voices

Note: The Spanish Latin American and North American `Sofia` voices are essentially the same voice. The most significant difference concerns how the two voices interpret a $ (dollar sign). The Latin American version uses the term *pesos*; the North American version uses the term *dólares*. Other minor differences might also exist between the two voices.

Creating a custom model

When you synthesize text, the service applies language-dependent pronunciation rules to convert the ordinary spelling of each word to a phonetic spelling. The service's pronunciation rules work well for common words, but they can yield imperfect results for unusual words, such as terms with foreign origins, personal names, and abbreviations or acronyms. If your application's lexicon includes such words, you can use the customization interface to specify how the service pronounces them.

A custom model is a dictionary of words and their translations. You create a custom model for a specific language, not for a specific voice. So a custom model can be used with any voice for its specified language. For example, a custom model that you create for the `en-US` language can be used with any US English voice. It cannot, however, be used with an `en-GB` or `en-AU` voice.

Customization is available for all languages. All voices support the use of both standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) phonetic symbols for word customization. For more information, see Understanding customization.

Creating a custom voice

IBM Cloud

Premium customers can work with IBM to train a new custom voice for their specific use case and target market. Creating a custom voice is different from customizing one of the service's existing voices. A custom voice is a unique new voice that is based on audio training data that the customer provides. IBM can train a custom voice with as little as one hour of training data.

To request a custom voice or for more information, complete and submit this IBM Request Form.

Using a voice for speech synthesis

Both the HTTP `POST` and `GET /v1/synthesize` methods, as well as the WebSocket `/v1/synthesize` method, accept an optional `voice` query parameter. You use the `voice` parameter to indicate the voice and language that are to be used for speech synthesis. The service bases its understanding of the language for the input text on the language of the specified voice.

Be sure to specify a voice that matches the language of the input text. For example, if you specify the French voice `fr-FR_ReneeV3Voice`, the service expects to receive input text that is written in French. If you pass text that is not written in the language of the voice (for example, English text for the French voice), the service might not produce meaningful results.

Specify a voice examples

The following example HTTP `POST` request uses the voice `en-US_AllisonV3Voice` for speech synthesis:

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--data "{\"text\":\"hello world\"}" \
--output hello_world.wav \
"{url}/v1/synthesize?voice=en-US_AllisonV3Voice"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--data "{\"text\":\"hello world\"}" \
--output hello_world.wav \
"{url}/v1/synthesize?voice=en-US_AllisonV3Voice"
```

The following example shows an equivalent HTTP `GET` request for speech synthesis:

IBM Cloud

```
$ curl -X GET -u "apikey:{apikey}" \
--output hello_world.wav \
"{url}/v1/synthesize?accept=audio%2Fwav&text=hello%20world&voice=en-US_AllisonV3Voice"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--output hello_world.wav \
"{url}/v1/synthesize?accept=audio%2Fwav&text=hello%20world&voice=en-US_AllisonV3Voice"
```

Using the default voice

If you omit the `voice` parameter from a request, the service uses the US English `en-US_MichaelV3Voice` by default. This default applies to all speech synthesis requests and to the `GET /v1/pronunciation` method.

IBM Cloud Pak for Data IBM Software Hub If you do not install the `en-US_MichaelV3Voice`, it cannot serve as the default voice. In this case, you must either

- Use the `voice` parameter to pass the voice that is to be used with each request.
- Specify a new default voice for your installation of Text to Speech for IBM Cloud Pak for Data by using the `defaultTTSVoice` property in the Speech services custom resource. For more information, see Installing Watson Text to Speech.

Multilingual speech synthesis

The service does not support multilingual speech synthesis at this time. All synthesis is based on the language of the voice that is specified by the `voice` parameter. Depending on the language and the word in question, you might be able to use customization to approximate the pronunciation of a word in a language that is different from the voice of the request. For more information, see Creating a custom model.

If you decide to use customization to emulate pronunciation in a different language, use the HTTP `GET /v1/pronunciation` method to see the pronunciation of the word in the other language. The method returns the phonemes that the service uses to pronounce the word in that language. For more information, see Phonetic translation.

You can adjust the phonemes that the method returns to match as closely as possible the phonemes that are available in your language. You can then create a custom model that includes a custom word with that translation and use that model with your synthesis request. Becuase two different languages might not support the same phonemes, it might not be possible to match exactly the sounds and pronunciation of one language with the phonetic symbols of another language.

Note: The Speech Synthesis Markup Language (SSML) `<speak>` element includes an `xml:lang` element, but that element applies to the entire request, and the service does not support its use as a way of specifying a different language for speech synthesis.

Listing information about voices

The IBM Watson® Text to Speech service provides methods for listing information about all of its available voices or about a specific voice.

Listing all voices

The `GET /v1/voices` method lists information about all available voices. It takes no arguments and returns a JSON array that is named `voices`. The array includes a separate object for each voice. The order in which the service returns voices can change from call to call. Do not rely on an alphabetized or static list of voices.

The following example lists all voices that are supported by the service:

IBM Cloud

```
$ curl -X GET -u "apikey:{apikey}" \
"{url}/v1/voices"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X GET \
--header "Authorization: Bearer {token}" \
"{url}/v1/voices"
```

This abbreviated response shows only the first few voices of the response:


```
{
  "voices": [
    {
      "name": "en-US_LisaV3Voice",
      "language": "en-US",
      "gender": "female",
      "url": "{url}/v1/voices/en-US_LisaV3Voice",
      "customizable": true,
      "supported_features": {
        "voice_transformation": false,
        "custom_pronunciation": true
      },
      "description": "Lisa: American English female voice.",
    },
    {
      "name": "es-LA_SofiaV3Voice",
      "language": "es-LA",
      "customizable": true,
      "gender": "female",
      "url": "{url}/v1/voices/es-LA_SofiaV3Voice",
      "supported_features": {
        "voice_transformation": false,
        "custom_pronunciation": true
      },
      "description": "Sofia: Latin American Spanish (español latinoamericano) female voice."
    },
    {
      "name": "pt-BR_IsabelaV3Voice",
      "language": "pt-BR",
      "customizable": true,
      "gender": "female",
      "url": "{url}/v1/voices/pt-BR_IsabelaV3Voice",
      "supported_features": {
        "voice_transformation": false,
        "custom_pronunciation": true
      },
      "description": "Isabela: Brazilian Portuguese (português brasileiro) female voice."
    },
    . . .
  ]
}
```


The fields of the voice objects provide the following information:

- `name` is an identifier for the voice (for example, `en-US_LisaV3Voice`). Specify this value for the `voice` parameter of the `/v1/synthesize` method.
- `language` specifies the language and region of the voice (for example, `en-US`).
- `gender` identifies the voice as `male` or `female`.

- `url` identifies the URL for the voice.
- `description` provides a brief description of the voice.
- `customizable` is a boolean value that indicates whether the voice can be customized with the service's customization interface. (This field, which provides the same information as the `custom_pronunciation` field, is maintained for backward compatibility.)
- `supported_features` describes the additional service features that are supported by the voice:
  - `voice_transformation` is a boolean value that indicates whether the voice supported the SSML `<voice-transformation>` element. The feature was available only for obsolete standard US English voices. You cannot use the `<voice-transformation>` element with any supported voice.
  - `custom_pronunciation` is a boolean value that indicates whether the voice can be customized with the service's customization interface.

Listing a specific voice

The `GET /v1/voices/{voice}` method lists information about a specific voice. It accepts two parameters.

- `voice` (path parameter, *required* string) - Identifies the voice for which information is to be returned. You specify a voice by its name (for example, `en-US_LisaV3Voice`).
- `customization_id` (query parameter, *optional* string) - Provides the globally unique identifier (GUID) of a custom model that is defined for the language of the specified voice. If you include a customization ID, you must make the request with credentials for the instance of the service that owns the custom model.

If you omit the `customization_id` parameter, the method returns JSON output for the specified voice that is identical to the information returned for a voice by the `GET /v1/voices` method. If you specify a `customization_id`, the output includes a `customization` field that provides information about the specified custom model.

The following example returns information about the `en-US_LisaV3Voice` and the specified custom model:

IBM Cloud

```
$ curl -X GET -u "apikey:{apikey}" \
"{url}/v1/voices/en-US_LisaV3Voice?customization_id=64f4807f-a5f1-5867-924f-7bba1a84fe97"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X GET \
--header "Authorization: Bearer {token}" \
"{url}/v1/voices/en-US_LisaV3Voice?customization_id=64f4807f-a5f1-5867-924f-7bba1a84fe97"
```

The attributes of the additional `customization` field provide metadata information such as the GUID, name, language, and description of the custom model. They also show the credentials of the model's owner, the date and time at which the model was created, and the date and time of its last modification.

```
{
  "name": "en-US_LisaV3Voice",
  "language": "en-US",
  "gender": "female",
  "url": "{url}/v1/voices/en-US_LisaV3Voice",
  "description": "Lisa: American English female voice.",
  "customizable": true,
  "supported_features": {
    "voice_transformation": false,
    "custom_pronunciation": true
  },
  "customization": {
    "customization_id": "64f4807f-a5f1-5867-924f-7bba1a84fe97",
    "owner": "297cfd08-330a-22ba-93ce-1a73f454dd98",
    "created": "2017-09-16T17:12:31.743Z",
    "name": "Customization test",
    "language": "en-US",
    "description": "Customization test",
    "last_modified": "2017-09-16T17:12:31.743Z"
  }
}
```

To see the custom words and prompts that the model includes, use the `GET /v1/customizations/{customization_id}` method. For more information, see Querying a custom model.

Using audio formats

The IBM Watson® Text to Speech service can return synthesized audio in a number of popular audio formats (or MIME types). For information about all supported formats, see Supported audio formats.

To make the best use of the service, you need to understand the sampling rate of the audio that the service returns and how to specify a different rate if you need to. For more information, see Sampling rate. The service always returns single-channel audio for all formats.

Sampling rate

The sampling rate (or sampling frequency) is the number of samples that are generated per second for the audio. Sampling rate is measured in Hertz (Hz) or kilohertz (kHz). For example, a rate of 16,000 samples per second is equal to 16,000 Hz (or 16 kHz).

Internally, the service always synthesizes audio with a sampling rate of 22,050 Hz. For many formats, the service also returns audio with this sampling rate. For other formats, the service returns audio with a different sampling rate.

For most formats, you can specify a different sampling rate for the audio. For the `audio/alaw`, `audio/l16`, and `audio/mulaw` formats, you must specify a sampling rate. You specify a sampling rate by including the `rate={integer}` parameter with the audio format specification. For more information, see Specifying an audio format.

When you specify a sampling rate, the service resamples the audio from 22,050 Hz to the specified rate before it returns the audio. A specified sampling rate must lie in the range of 8 kHz to 192 kHz. Some audio formats restrict the rate to specific values; the descriptions of the formats identify such restrictions.

Determining the sampling rate

The most reliable way to identify the sampling rate for any audio that the service returns is to extract the information from the audio stream itself. To determine the rate, call the `/v1/synthesize` method with some simple text (for example, "hello world") and specify the format and codec that you plan to use. You can then obtain the sampling rate by saving the audio stream to a file and opening it in an audio player such as one of those listed in Playing an audio file.

Supported audio formats

Table 1 lists the audio formats in which you can request synthesized audio. By default, the service returns the audio in the Ogg format with the Opus codec (`audio/ogg;codecs=opus`).

> ⚠ **Important:** The Ogg audio format is not supported with the Safari browser. If you are using the the Text to Speech service with the Safari browser, you must specify a different format in which you want the service to return the audio. For more information, see Specifying an audio format.

The service provides the following information for each format:

- *Default sampling rate* shows the sampling rate for audio in the indicated format if you do not specify an alternative rate.
- *Required parameters* indicates those formats for which you must specify a sampling rate for the returned audio.
- *Optional parameters* identifies formats for which you can optionally specify a sampling rate or other characteristics of the returned audio.

As shown in the *Audio formats* column for those formats that accept a `codecs` parameter, you separate all parameters of the format specification with a `;` (semicolon). For more information about the different formats, see the sections that follow the table.

Audio formats	Default sampling rate	Required parameters	Optional parameters
audio/alaw	None	rate={integer}	None
audio/basic	8000 Hz	None	None
audio/flac	22,050 Hz	None	rate={integer}
audio/l16	None	rate={integer}	endianness=big-endian endianness=little-endian
audio/mp3 audio/mpeg	22,050 Hz	None	rate={integer}

audio/mulaw	None	rate={integer}	None
audio/ogg audio/ogg;codecs=vorbis	22,050 Hz	None	rate={integer}
audio/ogg;codecs=opus	48,000 Hz	None	rate={integer}
audio/wav	22,050 Hz	None	rate={integer}
audio/webm audio/webm;codecs=opus	48,000 Hz	None	None
audio/webm;codecs=vorbis	22,050 Hz	None	rate={integer}

Summary of supported audio formats

audio/alaw format

A-law (audio/alaw) is a single-channel, lossy audio format that is encoded by using u-law (or mu-law) data that is similar to the audio/basic and audio/mulaw formats, though the A-law algorithm produces different signal characteristics. You must specify the sampling rate with this format. For example, specify audio/alaw;rate=8000 for audio that is sampled at 8 kHz.

Note: Due to the streaming nature of the returned audio, the A-law audio that is generated might not work in all audio players. Specifically, the attribute numSamples in the header of the audio stream is set to 0 regardless of the length of the audio.

audio/basic format

Basic audio is a single-channel, lossy audio format that is encoded by using 8-bit u-law (or mu-law) data that is sampled at 8 kHz. This format provides a lowest-common denominator media type. Audio in this format always has a sampling rate of 8 kHz.

For more information, see

- Internet Engineering Task Force (IETF) Request for Comment (RFC) 2046
- iana.org/assignments/media-types/audio/basic

audio/flac format

Free Lossless Audio Codec (FLAC) (.flac) is a lossless compressed audio coding format. You can optionally specify a sampling rate other than the default 22,050 Hz.

audio/l16 format

Linear 16-bit Pulse-Code Modulation (PCM) is an uncompressed audio data format (often .raw or .pcm). You must specify the sampling rate with this audio format. For example, specify audio/l16;rate=16000 for audio that is sampled at 16 kHz.

You can optionally specify the endianness for the audio by using the endianness parameter. Endianness indicates how bytes of data are ordered by the underlying computer architecture:

- Big-endian (endianness=big-endian) orders data by most-significant bit.
- Little-endian (endianness=little-endian) orders data by least-significant bit.

For example, specify audio/l16;rate=16000;endianness=big-endian to obtain audio that is sampled at 16 kHz and returned in big-endian order. If you omit the endianness, the default is little-endian. (Specifying the endianness is an issue only for the audio/l16 format, which does not include a header. Endianness is not a concern for the other formats.)

For more information, see IETF Request for Comment (RFC) 2586 .

audio/mp3 and audio/mpeg formats

MP3 or *Motion Picture Experts Group (MPEG)* is a lossy data compression format (MP3 and MPEG refer to the same format). You can optionally specify a sampling rate other than the default value. The default sampling rate is 22,050 Hz for Enhanced Neural and Expressive voices, and 24,000 Hz for Natural voices.

audio/mulaw format

8-bit mu-law (or u-law) audio is a single-channel, lossy audio format that is encoded by using 8-bit mu-law data. You must specify the sampling rate with this audio format. For example, specify `audio/mulaw;rate=16000` for audio that is sampled at 16 kHz.

audio/ogg format

Ogg format (`.ogg`) is a free, open container format that is maintained by the Xiph.org Foundation. You can specify the `codecs` parameter with the format to request an audio stream that is compressed with one of the following codecs:

- The *Opus* codec by specifying `audio/ogg;codecs=opus`. You can optionally specify a sampling rate other than the default 48,000 Hz. Only the following values are valid sampling rates: `48000`, `24000`, `16000`, `12000`, or `8000`. If you specify a value other than one of these, the service returns an error.

  > ⚠ **Important:** A current limitation causes the service to disregard a valid sampling rate. The service always returns the audio with a sampling rate of 48 kHz.

- The *Vorbis* codec by specifying `audio/ogg;codecs=vorbis` or simply `audio/ogg`. You can optionally specify a sampling rate other than the default 22,050 Hz.

Both codecs are free, open, lossy audio-compression formats. Opus is the preferred codec, but per the Ogg specification, the service returns the audio in Vorbis format if you omit the codec. If you omit an audio format altogether, the service returns the audio in Ogg format with the Opus codec by default.

For more information, see

- xiph.org/ogg
- xiph.org/vorbis
- opus-codec.org
- IETF Request for Comments (RFC) 7845

audio/wav format

Waveform Audio File Format (WAV) (`.wav`) is a standard container format that is often used for uncompressed audio bitstreams but can contain compressed audio, as well. You can optionally specify a sampling rate other than the default 22,050 Hz.

> **Note:** Due to the streaming nature of the returned audio, the WAV audio that is generated might not work in all audio players. Specifically, the attribute `numSamples` in the header of the audio stream is set to `0` regardless of the length of the audio.

audio/webm format

Web Media (WebM) (`.webm`) is an open media-file format. You can specify the `codecs` parameter with the format to request an audio stream that is compressed with one of the following codecs:

- The *Opus* codec by specifying `audio/webm;codecs=opus` or simply `audio/webm`. Audio in this format always has a sampling rate of 48 kHz.
- The *Vorbis* codec by specifying `audio/webm;codecs=vorbis`. You can optionally specify a sampling rate other than the default 22,050 Hz.

Both codecs are free, open, lossy audio-compression formats. Opus is the preferred codec.

For more information, see

- webmproject.org
- opus-codec.org

Specifying an audio format

By default, the service returns audio in the format `audio/ogg;codecs=opus`. You can specify a different audio format with either the HTTP or the WebSocket interface.

- With the HTTP `GET` and `POST /v1/synthesize` methods, you can specify a format by using the `Accept` request header or the `accept` query parameter.
  - If you use the `Accept` request header, you specify the format and any additional parameters as shown in the following example. The example specifies the `audio/l16` format and a sampling rate of 16,000 Hz.

    ```
    audio/l16;rate=16000
    ```

- If you use the `accept` query parameter, you must URL-encode the format and any additional parameters as shown in the following example. The example specifies the same format and sampling rate as the previous example.

  ```
  audio%2Fl16%3Brate%3D16000
  ```

  To receive audio in the default format, omit both the header and the query parameter. For more information, see Synthesizing text to audio.

- With the WebSocket `/v1/synthesize` method, you must specify a format by using the required `accept` parameter of the text message that you pass to initiate synthesis. To receive audio in the default format, specify the value `*/*` for the parameter. For more information, see Send input text.

Playing an audio file

To play an audio file that the service generates, use one of the following tools:

- A web browser such as Google Chrome™, Firefox®, or Microsoft® Internet Explorer®
- An audio player such as Audacity® (audacityteam.org) or FFmpeg (ffmpeg.org)

Synthesizing speech with the service

The HTTP interface

To synthesize text to speech with the HTTP REST interface of the IBM Watson® Text to Speech service, you call the `GET` or `POST /v1/synthesize` method. You specify the text that is to be synthesized and the voice and format for the spoken audio. You can also specify a custom model that is to be used with the request.

For more information about the HTTP interface, see the API & SDK reference.

Synthesizing text to audio

To synthesize text to audio, you call one of the two versions of the service's `/v1/synthesize` method:

- The `GET /v1/synthesize` method accepts the text that is to be synthesized as a required `text` query parameter. The maximum size of the request is 8 KB, which includes the input text, any SSML that you specify, and the URL and headers.
- The `POST /v1/synthesize` method accepts the text that is to be synthesized as a JSON construct in the required body of the request. The maximum size of the request is 8 KB for the URL and headers, and 5 KB for the input text that is sent in the body of the request. The 5 KB limit includes any SSML that you specify.

The two versions of the `/v1/synthesize` method have the following parameters in common:

`accept` (query parameter, *optional* string)

Specifies the requested audio format, or MIME type, in which the service is to return the audio. You can also specify this value with the HTTP `Accept` request header. URL-encode the argument to the `accept` query parameter. By default, the service returns the audio in the format `audio/ogg;codecs=opus`. For more information, see Using audio formats.

> ⚠ **Important:** The Ogg audio format is not supported with the Safari browser. If you are using the the Text to Speech service with the Safari browser, you must specify a different format in which you want the service to return the audio.

`voice` (query parameter, *optional* string)

Specifies the voice in which the text is to be spoken in the audio. Use the `/v1/voices` method to get the current list of supported voices. Omit the parameter to use the default voice. For more information, see Languages and voices and Using the default voice.

`customization_id` (query parameter, *optional* string)

Specifies a globally unique identifier (GUID) for a custom model that is to be used for the synthesis. A specified custom model must match the language of the voice that is used for the synthesis. If you include a customization ID, you must make the request with credentials for the instance of the service that owns the custom model. Omit the parameter to use the specified voice with no customization. For more information, see Understanding customization.

`rate_percentage` (query parameter, *optional* integer)

Specifies the global speaking rate for the entire synthesis request. The speaking rate is the speed at which the service speaks the text that it synthesizes into speech. A higher rate causes the text to be spoken more quickly; a lower rate causes the text to be spoken more slowly. The parameter changes the per-voice default rate for an entire request. For more information, see Modifying the speaking rate.

`pitch_percentage` (query parameter, *optional* integer)

Specifies the global speaking pitch for the entire synthesis request. The speaking pitch represents the tone of the speech that the service synthesizes. It represents how high or low the tone of the voice is perceived by the listener. A higher pitch results in speech that is spoken at a higher tone; a lower pitch results in speech that is spoken in a lower tone. The parameter changes the per-voice default pitch for an entire request. For more information, see Modifying the speaking pitch.

`spell_out_mode` (query parameter, *optional* string)

For German voices, specifies how individual characters of a string are to be spelled out. By default, the service spells out individual characters at the same rate at which it synthesizes text for a language. You can use the parameter to direct the service to spell out individual characters more slowly, in groups of one (`singles`), two (`pairs`), or three (`triples`). For more information, see Specifying how strings are spelled out.

`X-Watson-Metadata` (request header, *optional* string)

Associates a customer ID with data that is passed with a request. For more information, see Information security.

`X-Watson-Learning-Opt-Out` (request header, *optional* boolean)

IBM Cloud Indicates whether the service logs request and response data to improve the service for future users. To prevent IBM from accessing your data for general service improvements, specify `true` for the parameter. Opting out directs IBM to write to disk *no* user data (text or audio) for your request. You can also opt out at the account level. For more information, see Request logging.

If you specify an invalid query parameter or JSON field as part of the input to the `/v1/synthesize` method, the service returns a `Warnings` response header that describes and lists each invalid argument. The request succeeds despite the warnings.

Specifying input text

Both the `POST` and `GET /v1/synthesize` methods accept plain input text or text that is annotated with SSML. The two versions differ primarily in how you specify the text that is to be synthesized. The following examples both pass the plain text `Hello world`.

- The `POST /v1/synthesize` method accepts input text in the body of the request. You specify the input with a simple JSON construct that includes plain text or SSML. You must also specify a value of `application/json` for the `Content-Type` header.

  IBM Cloud

  ```
  $ curl -X POST -u "apikey:{apikey}" \
  --header "Content-Type: application/json" \
  --header "Accept: audio/wav" \
  --output hello_world.wav \
  --data "{\"text\":\"Hello world\"}" \
  "{url}/v1/synthesize?voice=en-US_MichaelV3Voice"
  ```

  IBM Cloud Pak for Data IBM Software Hub

  ```
  $ curl -X POST \
  --header "Authorization: Bearer {token}" \
  --header "Content-Type: application/json" \
  --header "Accept: audio/wav" \
  --output hello_world.wav \
  --data "{\"text\":\"Hello world\"}" \
  "{url}/v1/synthesize?voice=en-US_MichaelV3Voice"
  ```

- The `GET /v1/synthesize` method accepts input text that is specified by the `text` query parameter. You specify the input as plain text or SSML, both of which must be URL-encoded.

  IBM Cloud

  ```
  $ curl -X GET -u "apikey:{apikey}" \
  --header "Accept: audio/wav" \
  --output hello_world.wav \
  "{url}/v1/synthesize?text=Hello%20world&voice=en-US_MichaelV3Voice"
  ```

  IBM Cloud Pak for Data IBM Software Hub

  ```
  $ curl -X GET \
  --header "Authorization: Bearer {token}" \
  --header "Accept: audio/wav" \
  --output hello_world.wav \
  "{url}/v1/synthesize?text=Hello%20world&voice=en-US_MichaelV3Voice""""
  ```

Although the `POST` and `GET` methods offer equivalent functionality, it is always more secure to pass input text to the service with the `POST` method. A `POST` request passes input in the body of the request. A `GET` request exposes the data in the URL.

Punctuating input text

Write text for synthesis with the punctuation you would use normally. For example, include commas, periods, exclamation points, and questions marks as you would in normal writing.

The service considers punctuation when synthesizing text. For example, commas and end-of-sentence punctuation affect the audio by inserting pauses at appropriate places in the resulting synthesized speech. End-of-sentence punctuation such as periods, exclamation points, and question marks also

change the intonation and inflection of the speech. You can also use SSML elements to affect these aspects of the speech.

Specifying SSML input

The Speech Synthesis Markup Language (SSML) is an XML-based markup language that is designed to provide annotations of text for speech synthesis applications such as the Text to Speech service. You can use SSML elements and their attributes to gain greater control over the synthesis and resulting audio output.

For more information about using SSML to annotate input text, see Understanding SSML. For an inventory of all supported elements and attributes, see SSML elements.

Escaping XML control characters

Because you can submit input text that includes XML-based SSML annotations, the service validates all input to ensure that any SSML is correct and well formed. Therefore, you must escape any XML control characters that are present in the input text, regardless of whether the input includes SSML. Use the equivalent escape strings or character encodings from Table 1 instead of the indicated characters.

Character	Escape strings	Character encoding
" (double quotes)	"	"
' (apostrophe or single quote)	'	'
& (ampersand)	&	&
< (left angle bracket)	<	<
> (right angle bracket)	>	>
/ (forward slash)	None	/

Escaping XML control characters

For more information about how the service validates input text, see SSML validation.

Examples of input text

The following examples show how to specify input text with either method of the HTTP interface. They also show how to escape XML control characters. The examples include line breaks for readability. Do *not* include the line breaks in actual input.

Example input with a GET request

The following examples pass URL-encoded input with the `text` query parameter of the `GET /v1/synthesize` method:

- Plain text input:

```
text=This&20is&20the&20first&20sentence&20of&20the&20paragraph.&20Here
&20is&20another&20sentence.&20Finally,&20this&20is&20the&20last&20sentence.
```

- SSML input:

```
text=%22%3Cp%3E%3Cs%3EThis%20is%20the%20first%20sentence%20of%20the%20%3C
break%20time=%225s%22/%3E%20paragraph.%3C/s%3E%3Cs%3EHere%20is%20another
%20sentence.%3C/s%3E%3Cs%3EFinally,%20this%20is%20the%20last%20sentence.
%3C/s%3E%3C/p%3E%22
```

Example input with a POST request

The following examples pass input in the body of the `POST /v1/synthesize` method:

- Plain text input:

```
{
  "text": "This is the first sentence of the paragraph. Here is another
    sentence. Finally, this is the last sentence."
}
```

- SSML input:

```
{
  "text": "<p><s>This is the first sentence of the <break time=\"5s\"/>
    paragraph.</s><s>Here is another sentence.</s><s>Finally, this is
    the last sentence.</s></p>"
}
```

Example input with XML control characters

The following examples send two sentences to the `POST /v1/synthesize` method. The examples properly escape the embedded XML characters.

```
"What have I learned?" he asked. "Everything!"
```

- Plain text input:

```
{
  "text": "&quot;What have I learned?&quot; he asked. &quot;Everything!&quot;"
}
```

- SSML input:

```
{
  "text": "<s>&quot;What have I learned?&quot; he asked.
    &quot;<prodody rate=\"50\">Everything!</prosody>&quot;</s>"
}
```

The WebSocket interface

To synthesize text to speech with the WebSocket interface of the IBM Watson® Text to Speech service, you first establish a connection with the service by calling its `/v1/synthesize` method. You then send the text to be synthesized to the service as a JSON text message over the connection. The service automatically closes the WebSocket connection when it finishes processing the request.

The synthesize request and response cycle includes the following steps:

1. Open a connection.
2. Send input text.
3. Receive a response.

The WebSocket interface accepts identical input and produces identical results as the `GET` and `POST /v1/synthesize` methods of the HTTP interface. In addition, the WebSocket interface also supports use of the SSML `<mark>` element to identify the location of user-specified markers in the audio. It can also return timing information for all strings of the input text. (The `<mark>` element and word timings are available only with the WebSocket interface.)

- For more information about obtaining word timings, see Generating word timings.
- For more information about the WebSocket interface and its parameters, see the API & SDK reference.

Note: The snippets of example code that follow are written in JavaScript and are based on the HTML5 WebSocket API. For more information about the WebSocket protocol, see the Internet Engineering Task Force (IETF) Request for Comment (RFC) 6455.

Open a connection

You call the `/v1/synthesize` method over the WebSocket Secure (WSS) protocol to open a connection to the service. The method is available at the following endpoint:

```
wss://api.{location}.text-to-speech.watson.cloud.ibm.com/instances/{instance_id}/v1/synthesize
```

where `{location}` indicates where your application is hosted:

- `us-south` for Dallas
- `us-east` for Washington, DC
- `eu-de` for Frankfurt
- `au-syd` for Sydney
- `jp-tok` for Tokyo
- `eu-gb` for London
- `kr-seo` for Seoul

And `{instance_id}` is the unique identifier of the service instance.

> **Note:** The examples in the documentation abbreviate `wss://api.{location}.text-to-speech.watson.cloud.ibm.com/instances/{instance_id}` to `{ws_url}`. So all WebSocket examples call the method as `{ws_url}/v1/synthesize`.

A WebSocket client calls the `/v1/synthesize` method with the following query parameters to establish an authenticated connection with the service:

`access_token` (*required* string)

Pass a valid access token to authenticate with the service. You must use the access token before it expires.

- IBM Cloud Pass an Identity and Access Management (IAM) access token to authenticate with the service. You pass an IAM access token instead of passing an API key with the call. For more information, see Authenticating to IBM Cloud.
- IBM Cloud Pak for Data IBM Software Hub Pass an access token as you would with the `Authorization` header of an HTTP request. For more information, see Authenticating to IBM Cloud Pak for Data.

`voice` (*optional* string)

Specifies the voice in which the text is to be spoken in the audio. Use the `/v1/voices` method to get the current list of supported voices. Omit the parameter to use the default voice. For more information, see Languages and voices and Using the default voice.

`customization_id` (*optional* string)

Specifies the globally unique identifier (GUID) for a custom model that is to be used for the synthesis. A specified custom model must match the language of the voice that is used for the synthesis. If you include a customization ID, you must make the request with credentials for the instance of the service that owns the custom model. Omit the parameter to use the specified voice with no customization. For more information, see Understanding customization.

`rate_percentage` (*optional* integer)

Specifies the global speaking rate for the entire synthesis request. The speaking rate is the speed at which the service speaks the text that it synthesizes into speech. A higher rate causes the text to be spoken more quickly; a lower rate causes the text to be spoken more slowly. The parameter changes the per-voice default rate for an entire request. For more information, see Modifying the speaking rate.

`pitch_percentage` (*optional* integer)

Specifies the global speaking pitch for the entire synthesis request. The speaking pitch represents the tone of the speech that the service synthesizes. It represents how high or low the tone of the voice is perceived by the listener. A higher pitch results in speech that is spoken at a higher tone; a lower pitch results in speech that is spoken in a lower tone. The parameter changes the per-voice default pitch for an entire request. For more information, see Modifying the speaking pitch.

`spell_out_mode` (*optional* string)

For German voices, specifies how individual characters of a string are to be spelled out. By default, the service spells out individual characters at the same rate at which it synthesizes text for a language. You can use the parameter to direct the service to spell out individual characters more slowly, in groups of one (`singles`), two (`pairs`), or three (`triples`). For more information, see Specifying how strings are spelled out.

`x-watson-metadata` (*optional* string)

Associates a customer ID with data that is passed over the connection. The parameter accepts the argument `customer_id={id}`, where `id` is a random or generic string that is to be associated with the data. You must URL-encode the argument to the parameter, for example, `customer_id%3dmy_customer_ID`. By default, no customer ID is associated with the data. For more information, see Information security.

`x-watson-learning-opt-out` (*optional* boolean)

IBM Cloud Indicates whether the service logs requests and results that are sent over the connection. To prevent IBM from accessing your data for

general service improvements, specify `true` for the parameter. Opting out directs IBM to write to disk *no* user data (text or audio) for your request. You can also opt out at the account level. For more information, see Request logging.

The following snippet of JavaScript code opens a connection with the service. The call to the `/v1/synthesize` method passes the `voice` and `access_token` query parameters, the former to direct the service to use the US English Allison voice. Once the connection is established, the event listeners (`onOpen()`, `onClose()`, and so on) are defined to respond to events from the service.

```
var access_token = '{access_token}';
var wsURI = '{ws_url}/v1/synthesize'
  + '?access_token=' + access_token
  + '&voice=en-US_AllisonV3Voice';
var websocket = new WebSocket(wsURI);

websocket.onopen = function(evt) { onOpen(evt) };
websocket.onclose = function(evt) { onClose(evt) };
websocket.onmessage = function(evt) { onMessage(evt) };
websocket.onerror = function(evt) { onError(evt) };
```

Send input text

To synthesize text, the client passes a simple JSON text message to the service with the following parameters:

`text` (*required* string)

Provides the text that is to be synthesized. The client can pass plain text or text that is annotated with the Speech Synthesis Markup Language (SSML). The client can pass a maximum of 5 KB of input text with the request. The limit includes any SSML that you specify. For more information, see Specifying input text and the sections that follow it. (SSML input can also include the `<mark>` element. For more information, see Specifying an SSML mark.)

`accept` (*required* string)

Specifies the requested format (MIME type) of the audio. Use `*/*` to request the default audio format, `audio/ogg;codecs=opus`. For more information, see Using audio formats.

> ⚠ **Important:** The Ogg audio format is not supported with the Safari browser. If you are using the the Text to Speech service with the Safari browser, you must specify a different format in which you want the service to return the audio.

`timings` (*optional* string[ ])

Specifies that the service is to return word timing information for all strings of the input text. The service returns the start and end time of each token of the input. Specify `words` as the lone element of the array to request word timings. Specify an empty array or omit the parameter to receive no word timings. For more information, see Generating word timings. *Not supported for Japanese input text.*

The following snippet of JavaScript code passes a simple "Hello world" message as the input text and requests the default format for the audio. The calls are included in the `onOpen()` function that is defined for the client to ensure that they are sent only after the connection is established.

```
function onOpen(evt) {
  var message = {
    text: 'Hello world',
    accept: '*/*'
  };
  websocket.send(JSON.stringify(message));
}
```

The service responds to this message by sending a text message that confirms the format of the audio response. The following response confirms the default audio format.

```
{
  'binary_streams': [
    {
      content_type: 'audio/ogg;codecs=opus'
```

```
  }
 ]
}
```

Receive a response

After it confirms the audio format, the service sends the synthesized audio as a binary stream of data in the indicated format. For audio formats that include a header (for example, `audio/wav` and `audio/ogg`), the service returns the header before it sends the audio data. The header can span multiple binary responses. For all audio formats, the client needs to append all binary responses from the service to assemble the complete audio response.

In addition to sending a text message that confirms the requested audio format, the service can also send text messages with warnings or errors. The service also sends one or more text messages that include timing information if

- The input text includes one or more SSML `<mark>` elements.
- You specify the `timings` parameter with the request.

The client can handle text messages by responding to them, displaying them, or capturing them for use by the application (for example, if they contain mark locations).

When it finishes synthesizing the input text and sending all binary and text messages, the service automatically closes the WebSocket connection. The following simple `onMessage()` function appends text and binary messages that are received from the service to the appropriate variables based on their type. When the `onClose()` function executes, the entire audio stream has been received and the service sends no further binary or text messages.

```
var messages;
var audioStream;

function onMessage(evt) {
  if (typeof evt.data === string) {
    messages += evt.data;
  } else {
    console.log('Received ' + evt.data.size() + ' binary bytes');
    audioStream += evt.data;
  }
}

function onClose(evt) {
  // The service's response is complete.
}
```

WebSocket return codes

The service can send the following return codes to the client over the WebSocket connection:

- `1000` indicates normal closure of the connection, meaning that the purpose for which the connection was established has been fulfilled.
- `1002` indicates that the service is closing the connection due to a protocol error.
- `1006` indicates that the connection closed abnormally.
- `1009` indicates that the frame size exceeded the 4 MB limit.
- `1011` indicates that the service is terminating the connection because it encountered an unexpected condition that prevents it from fulfilling the request, such as an invalid argument. The return code can also indicate that the input text was too large.

If the socket closes with an error, the service sends the client an informative message of the form `{"error": "Specific error message"}` before closing. The service can also send non-fatal warning messages for unknown parameters. For more information about WebSocket return codes, see the Internet Engineering Task Force (IETF) Request for Comments (RFC) 6455.

Note: The WebSocket implementations of the SDKs can return different or additional response codes.

Example error and warning messages

The following examples show error responses. They include a JSON text message and a formatted message from the client's `onClose()` callback method. The formatted messages begin with the boolean `true` because the connection is closed. They also include the WebSocket error code that caused the closure.

- This example shows error messages for an invalid argument for the `accept` parameter:

  ```
  {
  ```

```
  "error": "Unsupported mimetype. Supported mimetypes are: ['application/json', 'audio/flac', ...]"
}
(True, 1011, u'see the previous message for the error details.')
```

- This example shows error messages for a missing `text` parameter:

```
{
  "error": "Required parameter \"text\" is missing."
}
(True, 1011, u'see the previous message for the error details.')
```

The following example shows a warning response, in this case for an unknown parameter named `invalid-parameter`. It does not include the second message because the connection is not closed by the warning.

```
{
  "warnings": "Unknown arguments: invalid-parameter."
}
```

Modifying speech synthesis characteristics

The IBM Watson® Text to Speech service includes query parameters that you can use to globally modify the characteristics of speech synthesis for an entire request: `rate_percentage`, `pitch_percentage`, and `spell_out_mode`. These parameters are available for both the HTTP and WebSocket interfaces.

Each of these parameters interacts with elements of the Speech Synthesis Markup Language (SSML). The descriptions of the parameters include information about how they interact with related SSML elements and attributes.

Modifying the speaking rate

The `rate_percentage` parameter is beta functionality.

The speaking rate is the speed at which the service speaks the text that it synthesizes into speech. A higher rate causes the text to be spoken more quickly; a lower rate causes the text to be spoken more slowly.

You can use the optional `rate_percentage` query parameter to modify the rate of the synthesized speech for a voice. Each voice has a default speaking rate that is optimized to represent a normal rate of speech. The parameter accepts an integer that represents the percentage change from the voice's default:

- Specify a signed negative integer to reduce the speaking rate by that percentage. For example, `-10` reduces the rate by ten percent.
- Specify an unsigned or signed positive integer to increase the speaking rate by that percentage. For example, `10` and `+10` increase the rate by ten percent.
- Specify `0` or omit the parameter to get the default speaking rate for the voice.

The best way to determine how the parameter affects the rate of a given voice is to experiment with different values. Increase or decrease the rate incrementally, by five or ten percent, before trying more significant changes. When decreasing the rate, values less than `-50` can begin to yield inadequate pronunciation. When increasing the rate, values greater than `100` might produce results that are not discernibly different from `100`.

Table 1 provides examples of how the service changes the speaking rate with different values for the `rate_percentage` parameter. The examples use the `en-US_AllisonV3Voice`.

Specification of the `rate_percentage` parameter	Audio sample
`rate_percentage=0` The service uses the default speaking rate. You can also omit the parameter to get the default behavior.	Your browser does not support the audio tag.
`rate_percentage=-20` The service uses a speaking rate that is 20 percent slower.	Your browser does not support the audio tag.
`rate_percentage=20` The service uses a speaking rate that is 20 percent faster.	Your browser does not support the audio tag.

Using the rate_percentage parameter

Interaction with SSML <prosody> element

The `rate_percentage` parameter changes the speaking rate for an entire request. You can also use `rate` attribute of the the SSML `<prosody>` element to

modify the rate of specific words of text. Although the `rate_percentage` parameter accepts only a percentage value, the `rate` attribute accepts values of different forms. For more information, see The rate attribute.

If you use the `rate_percentage` parameter and also use the `rate` attribute of the `<prosody>` tag to modify the rate for specific text of the same request, the effects are multiplicative. For example, suppose you specify the `rate_percentage` parameter with a value of `-10`, which slows the speaking rate by 10 percent. If you also use the `<prosody>` element to specify an equivalent `rate` of `-10%` for specific text of the request, that text is spoken at a rate that is 81 percent of the default rate for the voice.

Speaking rate examples

The following HTTP `POST /v1/synthesize` method uses the `rate_percentage` query parameter with a value of `-5`. The resulting audio is spoken at a rate that is five percent slower than it is by default. The example uses the US English voice `en-US_AllisonV3Voice`.

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output rate-percentage.wav \
--data "{\"text\":\"This text is spoken five percent slower than the default.\"}" \
"{url}/v1/synthesize?voice=en-US_AllisonV3Voice&rate_percentage=-5"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output rate-percentage.wav \
--data "{\"text\":\"This text is spoken five percent slower than the default.\"}" \
"{url}/v1/synthesize?voice=en-US_AllisonV3Voice&rate_percentage=-5"
```

The following WebSocket example shows inclusion of the parameter with the same value on the request to establish a connection:

```
var access_token = '{access_token}';
var wsURI = '{ws_url}/v1/synthesize'
  + '?access_token=' + access_token
  + '&voice=en-US_AllisonV3Voice'
  + '&rate_percentage=-5';
var websocket = new WebSocket(wsURI);
```

Modifying the speaking pitch

The `pitch_percentage` parameter is beta functionality.

The speaking pitch represents the pitch, or tone, of the speech that the service synthesizes, which also affects the intonation. It represents how high or low the tone of the voice is perceived by the listener. A higher pitch results in speech that is spoken at a higher tone and is perceived as a higher voice; a lower pitch results in speech that is spoken in a lower tone and is perceived as a lower, deeper voice.

You can use the optional `pitch_percentage` query parameter to modify the pitch of the synthesized speech for a voice. Each voice has a default, baseline pitch that reflects the intended tone of that voice. The parameter accepts an integer that represents the percentage change from the voice's default:

- Specify a signed negative integer to reduce the pitch by that percentage. For example, `-10` reduces the pitch by ten percent.
- Specify an unsigned or signed positive integer to increase the pitch by that percentage. For example, `10` and `+10` increase the pitch by ten percent.
- Specify `0` or omit the parameter to get the default pitch for the voice.

The best means of determining how the parameter affects the pitch of a given voice is to experiment with different values. Increase or decrease the rate incrementally, by five or ten percent, before trying more drastic changes. When decreasing the pitch, a value of `-50` halves the baseline pitch; lower values might not yield appreciable or positive results. Similarly, when increasing the pitch, a value of `100` doubles the pitch; higher values might produce results that are not discernibly different from `100`.

Table 2 provides examples of how the service changes the speaking pitch with different values for the `pitch_percentage` parameter. The examples use the `en-US_AllisonV3Voice`.

Specification of the `pitch_percentage` parameter	Audio sample

pitch_percentage=0 The service uses the default speaking pitch. You can also omit the parameter to get the default behavior.	Your browser does not support the audio tag.
pitch_percentage=-20 The service uses a speaking pitch that is 20 percent lower.	Your browser does not support the audio tag.
pitch-percentage=20 The service uses a speaking pitch that is 20 percent higher.	Your browser does not support the audio tag.

Using the pitch_percentage parameter

Interaction with SSML <prosody> element

The `pitch_percentage` parameter changes the pitch for an entire request. You can also use `pitch` attribute of the the SSML `<prosody>` element to modify the pitch of specific words of text. Although the `pitch_percentage` parameter accepts only a percentage value, the `pitch` attribute accepts values of different forms. For more information, see The pitch attribute.

If you use the `pitch_percentage` parameter and also use the `pitch` attribute of the `<prosody>` tag to modify the pitch for specific text of the same request, the effects are multiplicative. For example, suppose you specify the `pitch_percentage` parameter with a value of `10`, which increases the pitch by 10 percent. If you also use the `<prosody>` element to specify an equivalent `pitch` of `+10%` for specific text of the request, that text is spoken with a pitch that is 121 percent of the default baseline pitch for the voice.

Speaking pitch examples

The following HTTP `POST /v1/synthesize` method uses the `pitch_percentage` query parameter with a value of `5`. The resulting audio is spoken with a pitch that is five percent higher than it is by default. The example uses the US English voice `en-US_AllisonV3Voice`.

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output pitch-percentage.wav \
--data "{\"text\":\"This text is spoken five percent higher than the default.\"}" \
"{url}/v1/synthesize?voice=en-US_AllisonV3Voice&pitch_percentage=5"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output pitch-percentage.wav \
--data "{\"text\":\"This text is spoken five percent higher than the default.\"}" \
"{url}/v1/synthesize?voice=en-US_AllisonV3Voice&pitch_percentage=5"
```

The following WebSocket example shows inclusion of the parameter with the same value on the request to establish a connection:

```
var access_token = '{access_token}';
var wsURI = '{ws_url}/v1/synthesize'
  + '?access_token=' + access_token
  + '&voice=en-US_AllisonV3Voice'
  + '&pitch_percentage=5';
var websocket = new WebSocket(wsURI);
```

Specifying how strings are spelled out

The service's default pronunciation of alphabetic, numeric, and alphanumeric strings varies by language, with each language having its own rules. For any language, the pronunciation also depends on the length and context of the string, as well its composition in terms of the positions and grouping of its characters

- Alphabetic strings are pronounced as words if they are pronounceable in the language of the voice. Otherwise, the characters are spelled out individually. Whether a string is pronounced or spelled out can also depend on the case of the letters.
- Numeric strings can be pronounced as cardinal numbers, ordinal numbers, dates, or in other ways, including as individual digits.

- Alphanumeric strings that contain both letters and numbers can be pronounced in different ways depending on the characteristics of the string. For example, they might be pronounced as combinations of words, cardinal numbers, or digits. But as mentioned previously, the pronunciation depends largely on the string and its composition.

The best means of controlling how a specific string is pronounced is to use the `<say-as>` element of the SSML. The element allows you to specify exactly how a string is synthesized. You use the `interpret-as` attribute of the `<say-as>` element to indicate that a string of characters is to be pronounced as `letters`, `digits`, or in some other way. The available values differ by language.

If you use the `letters` or `digits` attributes, all letters and numbers of the string are pronounced individually. This is especially useful for spelling out the alphanumeric characters of a hexadecimal string or an identification or account number. For more information, including language-specific differences, see The say-as element.

Specifying the pace at which strings are spelled out

The `spell_out_mode` parameter is beta functionality that is supported only for German voices.

By default, the service reads characters that it spells out at the same rate at which it reads text for the language being synthesized. For German voices, however, you can use the optional `spell_out_mode` query parameter to indicate how the service is to spell out individual characters: one, two, or three at a time. The parameter applies globally to control the pace at which the service reads all strings that it spells out from a request. Use the `spell_out_mode` parameter only for longer strings. The parameter has no effect on a string that contains fewer than three characters.

Table 3 provides examples of how the service pronounces the following text with different values for the `spell_out_mode` parameter:

```
Die Nummer ist <say-as interpret-as='digits'>AB7234987FFA</say-as>.
```

Specification of the `spell_out_mode` parameter	Audio sample
`spell_out_mode=default` The service reads the characters in succession, typically with no pause between them, at the rate at which it synthesizes speech for the request. You can also omit the parameter to get the default behavior.	Your browser does not support the audio tag.
`spell_out_mode=singles` The service reads the characters one at a time, with a brief pause between each character.	Your browser does not support the audio tag.
`spell_out_mode=pairs` The service reads the characters two at a time, with a brief pause between each pair.	Your browser does not support the audio tag.
`spell_out_mode=triplets` The service reads the characters three at a time, with a brief pause between each triplet.	Your browser does not support the audio tag.

Using the spell_out_mode parameter for German

Note: Alphanumeric strings that mix letters and digits can yield different groupings of characters. If you do not use the `<say-as>` element, groups of letters and digits can be pronounced in ways that do not necessarily conform to the requested mode.

Spell-out mode examples

The following HTTP `POST /v1/synthesize` method uses the `spell_out_mode` query parameter with a value of `singles`. The example uses the German voice `de-DE_ErikaV3Voice`. The alphanumeric text to be synthesized is a hexadecimal string that is wrapped in a `<say-as>` element and interpreted as `digits`.

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output spell-out-mode.wav \
--data "{\"text\":\"Die Nummer ist <say-as interpret-as='digits'>AB7234987FFA</say-as>.\"}" \
"{url}/v1/synthesize?voice=de-DE_ErikaV3Voice&spell_out_mode=singles"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
```

```
--header "Accept: audio/wav" \
--output spell-out-mode.wav \
--data "{\"text\":\"Die Nummer ist <say-as interpret-as='digits'>AB7234987FFA</say-as>.\"}" \
"{url}/v1/synthesize?voice=de-DE_ErikaV3Voice&spell_out_mode=singles"
```

The following WebSocket example shows inclusion of the parameter with the same value on the request to establish a connection:

```
var access_token = '{access_token}';
var wsURI = '{ws_url}/v1/synthesize'
  + '?access_token=' + access_token
  + '&voice=de-DE_ErikaV3Voice'
  + '&spell_out_mode=singles';
var websocket = new WebSocket(wsURI);
```

Modifying speech synthesis with expressive neural voices

How to modify speech synthesis

The expressive neural voices that are available with the IBM Watson® Text to Speech service offer some additional features that are not available with other types of voices: *using speaking styles*, *emphasizing interjections*, and *emphasizing words*. These features are available for both the HTTP and WebSocket interfaces.

The features involve the use of elements of the Speech Synthesis Markup Language (SSML). The descriptions of the features provide information about how they interact with related SSML elements and attributes.

Using speaking styles

The expressive neural voices determine the sentiment of the text from the context of its words and phrases. The speech that they produce, in addition to having a very conversational style, reflects the mood of the text. The expressive voices naturally express gratitude, thankfulness, happiness, empathy, confusion, and other sentiments by default, with no explicit additional tagging.

However, you can embellish the voices' natural tendencies by using the `<express-as>` element with the required `style` attribute to indicate that all or some of the text is to emphasize specific characteristics. These characteristics are referred to as speaking styles:

- `cheerful` - Expresses happiness and good news. The style is upbeat, welcoming, and conveys a positive message.
- `empathetic` - Expresses empathy and compassion. The style has sympathetic undertones, but it is not excessively sorrowful.
- `neutral` - Expresses objectivity and evenness. The style strives for less emotion, and instead conveys a more even and instructional tone.
- `uncertain` - Expresses uncertainty and confusion. The style conveys the feeling of being unsure or in doubt.

In many cases, the effect of the styles is very subtle. In such cases, the differences might not be readily discernible from the default expressive tone. But there are instances where the style can audibly enhance the voices' inherent ability to detect and express emotion. The differences are often detectable only on certain words and phrases.

Expressing speaking styles with SSML

The following HTTP request uses the `<express-as>` element with a `style` of `cheerful` to modify the speaking style of the entire input text. The example uses the `en-US_EmmaExpressive` voice.

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output full-cheerful-style.wav \
--data "{\"text\":\"<express-as style='cheerful'>Oh, that&apos;s good news! I am very happy for you!</express-as>\"}" \
"{url}/v1/synthesize?voice=en-US_EmmaExpressive"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output full-cheerful-style.wav \
--data "{\"text\":\"<express-as style='cheerful'>Oh, that&apos;s good news! I am very happy for you!</express-as>\"}" \
"{url}/v1/synthesize?voice=en-US_EmmaExpressive"
```

The following example HTTP request uses the `cheerful` style for the first sentence of the request only. The second sentence is spoken with the default voice. The example again uses the `en-US_EmmaExpressive` voice.

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output partial-cheerful-style.wav \
--data "{\"text\":\"<express-as style='cheerful'>Oh, that&apos;s good news!</express-as> Do you need any further help?\"}" \
"{url}/v1/synthesize?voice=en-US_EmmaExpressive"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output partial-cheerful-style.wav \
--data "{\"text\":\"<express-as style='cheerful'>Oh, that&apos;s good news!</express-as> Do you need any further help?\"}" \
"{url}/v1/synthesize?voice=en-US_EmmaExpressive"
```

Emphasizing interjections

When you use expressive neural voices, the service automatically detects a number of common interjections based on context. In the resulting audio, it gives them the natural emphasis that a human would use in normal conversation. Expressive voices support a different set of interjections based on language. Please refer to the language specific pages for more details on which interjections are supported.

Enabling or disabling interjections with SSML

The service always interprets and pronounces most of the interjections as expected. However, there are some interjections that could be used in a different context. In such cases, the default behaviour of the service is to disable such interjections by default. You can use SSML to enable or disable such interjections. The `interpret-as` attribute of the SSML `<say-as>` element accepts an additional value, `interjection`. When you use the `interpret-as` attribute with the `interjection` value, you include the additional `enabled` attribute with a value of `true` or `false` to indicate whether the word is to be pronounced as an interjection. Every language has a different subset of interjections that can be enabled/disabled through SSML. Please refer to the language specific pages for more details on which interjections can be enabled/disabled using SSML.

The following example HTTP request directs the service not to pronounce `oh` and `aha` as interjections. The example uses the `en-US_AllisonExpressive` voice.

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output disabled-interjections.wav \
--data "{\"text\":\"<say-as interpret-as='interjection' enabled='false'>Oh</say-as>, in addition, the <say-as interpret-as='interjection' enabled='false'>aha</say-as> wasp is endemic to Australia.\"}" \
"{url}/v1/synthesize?voice=en-US_AllisonExpesssive"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output disabled-interjections.wav \
--data "{\"text\":\"<say-as interpret-as='interjection' enabled='false'>Oh</say-as>, in addition, the <say-as interpret-as='interjection' enabled='false'>aha</say-as> wasp is endemic to Australia.\"}" \
"{url}/v1/synthesize?voice=en-US_AllisonExpesssive"
```

Emphasizing words

All the expressive voices use a conversational style that naturally applies the correct intonation from context. But you can use the SSML `<emphasis>` element to indicate that one or more words are to be given more or less emphasis in the synthesized audio. The change in stress can be indicated by an increase or decrease in pitch, timing, volume, or other acoustic attributes.

The `<emphasis>` element supports an optional `level` attribute that accepts one of the following values:

- `none` - Prevents the service from emphasizing text that might otherwise be emphasized.
- `moderate` - Provides a noticeable amount of emphasis to the text. This value is the default if you omit the `level` attribute.
- `strong` - Provides a more significant amount of emphasis to the text than the `moderate` level provides.
- `reduced` - De-emphasizes the text by tending to reduce its significance in the audio. This level is the opposite of stressing the text.

Table 3 provides examples of each available `level` for the `emphasis` element. The samples use the `en-US_MichaelExpressive` voice. A request that uses the `emphasis` element fails if a specified level is not one of the supported values.

Level of emphasis	Example sentence	Audio sample
No emphasis	`"I am going to give her the book."`	Your browser does not support the audio tag.
none	`"I am going to give her the <emphasis level='none'>book</emphasis>."`	Your browser does not support the audio tag.
moderate	`"I am going to <emphasis level='moderate'>give</emphasis> her the book."`	Your browser does not support the audio tag.
strong	`"I am going to <emphasis level='strong'>give</emphasis> her the book."`	Your browser does not support the audio tag.
reduced	`"I <emphasis level='reduced'>am going to give</emphasis> her the book."`	Your browser does not support the audio tag.

Emphasizing words

Emphasizing a word with SSML

The following example HTTP request uses the `<emphasis>` element with a `level` of `moderate` (the default) to emphasize the word `give` in the sentence. The example uses the `en-US_MichaelExpressive` voice.

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output strong-emphasis.wav \
--data "{\"text\":\"I am going to <emphasis level='moderate'>give</emphasis> her the book.\"}" \
"{url}/v1/synthesize?voice=en-US_MichaelExpesssive"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--output strong-emphasis.wav \
--data "{\"text\":\"I am going to <emphasis level='moderate'>give</emphasis> her the book.\"}" \
"{url}/v1/synthesize?voice=en-US_MichaelExpesssive"
```

English expressive voices

Speaking styles

Table 1 provides examples of each available `style` for the `<express-as>` element. To demonstrate the effect of the styles, the table provides two samples for each example. The first sample speaks the text with the default `en-US_EmmaExpressive` voice. The second sample speaks the same text with the same voice but with the indicated style. A request that uses the `<express-as>` element fails if the `style` is not one of the supported values or is omitted from the element.

Style	Example input text	Audio sample
cheerful	`"Oh, that&apos;s good news. I am very happy for you!"`	Your browser does not support the audio tag.

	"<express-as style='cheerful'>Oh, that's good news. I am very happy for you!</express-as>"	Your browser does not support the audio tag.
empathetic	"Oh, I'm sorry to hear that. I know how difficult that can be."	Your browser does not support the audio tag.
	"<express-as style='empathetic'>Oh, I'm sorry to hear that. I know how difficult that can be.</express-as>"	Your browser does not support the audio tag.
neutral	"A five-alarm fire early this morning claimed the lives of more than a dozen residents."	Your browser does not support the audio tag.
	"<express-as style='neutral'>A five-alarm fire early this morning claimed the lives of more than a dozen residents.</express-as>"	Your browser does not support the audio tag.
uncertain	"That's strange. Hmm, I don't know if I've seen this before."	Your browser does not support the audio tag.
	"<express-as style='uncertain'>That's strange. Hmm, I don't know if I've seen this before.</express-as>"	Your browser does not support the audio tag.

Speaking styles

Emphasizing interjections

For English, we currently support the following interjections - `aha`, `hmm`, `huh`, `oh`, `uh`, `uh-huh` and `um`.

Table 2 lists the interjections that the service recognizes and provides examples of how they are pronounced in synthesized speech. The samples use the `en-US_AllisonExpressive` voice. The table shows the primary spelling of each interjection. The service recognizes alternative spellings for some of the interjections. For example, `oh` and `ohh` produce the same sound, as do `hmm` and `hmmm`. However, the service produces slightly different pronunciations for other alternative spellings, such as `ooh` and `uhm`,

Interjection	Example sentence	Audio sample
aha	"Aha. So that's the secret."	Your browser does not support the audio tag.
hmm	"Hmm. I'm not sure I understand."	Your browser does not support the audio tag.
huh	"Huh, I hadn't noticed that."	Your browser does not support the audio tag.
oh	"Oh, let me get that for you."	Your browser does not support the audio tag.
uh	"Uh, let me check."	Your browser does not support the audio tag.
uh-huh	"Uh-huh, that's right."	Your browser does not support the audio tag.
um	"That's, um, not quite right."	Your browser does not support the audio tag.

Interjections that are emphasized

Enabling or disabling interjections with SSML

For English, it is possible to enable/disable the following interjections using SSML - `aha` and `oh`.

Usage notes for interjections

Keep the following in mind when using interjections:

- The `<say-as>` element with the `interpret-as="interjection"` attribute has no effect if the word to be modified is not one of the interjections `aha` or `oh`.
- Including punctuation with the text that is to be modified can affect whether the interjection is enabled or disabled. Do not include any punctuation characters with the word inside of the `<say-as>` tag.
- The word `oh`, when used colloquially to indicate the number zero, is never treated as an interjection. For example, the word is not emphasized as an

interjection in the following sentence: `The number is one oh three.`

- The neutral style does not emphasize interjections to the extent of the default expressive voice or the other styles.

Spanish expressive voices

Speaking styles

Table 1 provides examples of each available `style` for the `<express-as>` element for Spanish. To demonstrate the effect of the styles, the table provides two samples for each example. The first sample speaks the text with the default `es-LA_DanielaExpressive` voice. The second sample speaks the same text with the same voice but with the indicated style. A request that uses the `<express-as>` element fails if the `style` is not one of the supported values or is omitted from the element.

Style	Example input text	Audio sample
cheerful	"¡Excelente! ¿Hay algo más en lo que pueda ayudarte?"	Your browser does not support the audio tag.
	"<express-as style="cheerful"> ¡Excelente! ¿Hay algo más en lo que pueda ayudarte? </express-as>"	Your browser does not support the audio tag.
empathetic	"Lamento escuchar que este asunto aún no se ha resuelto, y le pido disculpas."	Your browser does not support the audio tag.
	"<express-as style="empathetic"> Lamento escuchar que este asunto aún no se ha resuelto, y le pido disculpas. </express-as>"	Your browser does not support the audio tag.
neutral	"Durante el otoño, el clima es ventoso."	Your browser does not support the audio tag.
	"<express-as style="neutral"> Durante el otoño, el clima es ventoso. </express-as>"	Your browser does not support the audio tag.
uncertain	"Disculpe, pero ¿le importaría repetir lo que me decía? No logré oirlo bien."	Your browser does not support the audio tag.
	"<express-as style="uncertain"> Disculpe, pero ¿le importaría repetir lo que me decía? No logré oirlo bien. </express-as>"	Your browser does not support the audio tag.

Speaking styles

Emphasizing interjections

For Spanish, we currently support the following interjections - `aah`, `ajá`, `eh`, `oh` and `uf`.

Table 2 lists the interjections that the service recognizes and provides examples of how they are pronounced in synthesized speech. The samples use the `es-LA_DanielaExpressive` voice. The table shows the primary spelling of each interjection. The service recognizes alternative spellings for some of the interjections. For example, `oh` and `ohh` produce the same sound, as do `uf` and `uff`. However, the service produces slightly different pronunciations for other alternative spellings, such as `ah` and `eeh`.

Interjection	Example sentence	Audio sample
aah	"Aah, ese pago ya fue realizado la semana pasada, y su cuenta quedó saldada."	Your browser does not support the audio tag.
ajá	"¡Ajá, ya logré localizar su pedido! Está en camino y llegará dentro de 24 horas."	Your browser does not support the audio tag.
eh	"Eh, no sé si podremos enviar un técnico a su residencia antes del miércoles."	Your browser does not support the audio tag.
oh	"Oh, claro que podemos guardar su equipaje en consigna hasta la hora de su salida. No hay problemas."	Your browser does not support the audio tag.

uf	"Uf, no creo que pueda concertarle una cita hasta que recibamos sus documentos."	Your browser does not support the audio tag.

Interjections that are emphasized

Enabling or disabling interjections with SSML

For Spanish, all interjections are enabled by default, and it is not possible to enable/disable these interjections.

Usage notes for interjections

Keep the following in mind when using interjections:

- The neutral style does not emphasize interjections to the extent of the default expressive voice or the other styles.

Portuguese expressive voices

Speaking styles

Table 1 provides examples of each available `style` for the `<express-as>` element for Portuguese. To demonstrate the effect of the styles, the table provides two samples for each example. The first sample speaks the text with the default `pt-BR_LucasExpressive` voice. The second sample speaks the same text with the same voice but with the indicated style. A request that uses the `<express-as>` element fails if the `style` is not one of the supported values or is omitted from the element.

Style	Example input text	Audio sample
cheerful	"Claro! Que ótima notícia!"	Your browser does not support the audio tag.
	"<express-as style="cheerful"> Claro! Que ótima notícia! </express-as>"	Your browser does not support the audio tag.
empathetic	"Infelizmente, não vou conseguir te ajudar nisso."	Your browser does not support the audio tag.
	"<express-as style="empathetic"> Infelizmente, não vou conseguir te ajudar nisso. </express-as>"	Your browser does not support the audio tag.
neutral	"O clima é muito seco no outono."	Your browser does not support the audio tag.
	"<express-as style="neutral"> O clima é muito seco no outono. </express-as>"	Your browser does not support the audio tag.
uncertain	"Desculpe, poderia repetir seu número novamente?"	Your browser does not support the audio tag.
	"<express-as style="uncertain"> Desculpe, poderia repetir seu número novamente? </express-as>"	Your browser does not support the audio tag.

Speaking styles

Emphasizing interjections

For Portuguese, we currently support the following interjections - `ah`, `aham`, `ahn`, `eh`, `hum` and `uhum`.

Table 2 lists the interjections that the service recognizes and provides examples of how they are pronounced in synthesized speech. The samples use the `pt-BR_LucasExpressive` voice. The table shows the primary spelling of each interjection. The service recognizes alternative spellings for some of the interjections. For example, `eh` and `ehh` produce the same sound, as do `hum` and `humm`. However, the service produces slightly different pronunciations for other alternative spellings, such as `aah` and `eeh`.

Interjection	Example sentence	Audio sample
ah	"Ah, não tem problema algum!"	Your browser does not support the audio tag.
aham	"Aham, já encaminharemos para você imediatamente."	Your browser does not support the audio tag.
ahn	"Ahn, poderia aguardar na linha por favor?"	Your browser does not support the audio tag.

eh	"Eh, só um segundinho que vou checar com meu gerente."	Your browser does not support the audio tag.
hum	"Hum, não tenho muita certeza disso."	Your browser does not support the audio tag.
uhum	"Uhum, seu pacote será entregue até o final do dia!"	Your browser does not support the audio tag.

Interjections that are emphasized

Enabling or disabling interjections with SSML

For Portuguese, all interjections are enabled by default, and it is not possible to enable/disable these interjections.

Usage notes for interjections

Keep the following in mind when using interjections:

- The neutral style does not emphasize interjections to the extent of the default expressive voice or the other styles.

Generating word timings

You can use the WebSocket interface of the IBM Watson® Text to Speech service to obtain timing information for user-specified locations such as word boundaries or for all words of the input text:

- Include the SSML `<mark>` element in input text to identify the time at which the marker occurs in the audio.
- Specify the `timings` parameter of a JSON text message to obtain timing information for all strings of the input text.

Timing information is useful for synchronizing the audio and the input text. For example, you can coordinate an avatar's or robot's gestures with the content of the synthesized speech.

Note: The `<mark>` element and the `timings` parameter are available only with the WebSocket interface, not with the HTTP interface. Also, the `timings` parameter is not supported for Japanese input text.

How the service returns word timings

To return mark or word timing information, the service multiplexes independent binary and text streams to construct its response:

- For each `<mark` > element, the service returns a JSON text message. Each message indicates the exact time from the beginning of the synthesized audio at which the mark occurs.
- For word timings for all strings, the service returns one or more JSON text messages. Each message contains an array of words and their start and end times from the beginning of the synthesized audio.

The binary and text streams that the service sends are independent. So the service has little control over the number of audio chunks that it delivers and when the user receives the text and audio messages. For example, if the audio is synthesized more quickly than it is compressed, all text messages might arrive before any of the audio arrives.

In practical terms, the service can send an arbitrary number of audio chunks, including multiple chunks of audio before and after each text message. It is also possible for a single binary chunk to contain audio data that both precede and follow the timing information for a mark or word.

However, the text message that contains the timing information always arrives before the binary chunk that contains the corresponding audio. Moreover, the audio messages always arrive in order so that you can construct complete and accurate audio of the synthesized text from the binary results.

Specifying an SSML mark

The optional SSML `<mark>` element is an empty tag that places a marker into the text to be synthesized. The client is notified when all of the text that precedes the `<mark>` element has been synthesized.

The element accepts a single `name` attribute that specifies a string that uniquely identifies the mark. The name must begin with an alphanumeric character. The service returns the name along with the time at which the mark occurs from the beginning of the synthesized audio. You can include any number of marks in the input text.

The following snippet of JavaScript code includes an instance of the `<mark>` element with the name `here`:

```
function onOpen(evt) {
  var message = {
    text: 'Hello <mark name="here"/> world',
```

```
    accept: '*/*'
  };
  websocket.send(JSON.stringify(message));
}
```

When it finishes synthesizing the text that precedes the mark, the service sends a text message that identifies the name of the mark and the time in seconds at which the mark occurs in the audio:

```
{
  "marks": [
    ["here", 0.501]
  ]
}
```

The text message that contains the timing information always arrives before the audio chunk that contains the mark's location.

Requesting word timings for all words

The optional `timings` parameter of the JSON object that you pass to the service for a request returns timing information for all strings of the input text. This convenience eliminates the need to specify the SSML `<mark>` element for each word of the input. Pass an array that includes the string `words` to request word timings. Pass an empty array or omit the parameter to receive no timing information.

The service returns word timings over the WebSocket connection in the same way that it returns timing information for individual `<mark>` elements. It returns one or more JSON text messages. Each message contains an array of words and their start and end times in seconds from the beginning of the synthesized audio. For example, the following example requests word timing information:

```
function onOpen(evt) {
  var message = {
    text: 'I have a pet bird.',
    accept: '*/*',
    timings: ['words']
  };
  websocket.send(JSON.stringify(message));
}
```

In response, the service can return the following text messages:

```
{
  "words": [
    [
      "I", 0.0, 0.157
    ],
    [
      "have", 0.157, 0.321
    ],
    [
      "a", 0.321, 0.406
    ]
  ]
}
{
  "words": [
    [
      "pet", 0.406, 0.731
    ],
    [
      "bird.", 0.731, 1.049
    ]
  ]
}
```

The response is just an example. The service can return one or more text messages with timing information for the input. It can also return a separate text message for each word of the input. Moreover, the messages can be interspersed with responses that contain binary chunks of audio. But the text message that contains the timing information for a word always arrives before the audio chunk that contains that word.

Timings for plain text

The service's synthesis process involves a text normalization step that spells out numbers, dates, times, monetary amounts, acronyms, and abbreviations. The results correspond to how such strings are spoken. For example, the string *$200* is spoken as three words: *two*, *hundred*, and *dollars*. Because word timing information is used to synchronize the audio with the input text, the service returns timing information that corresponds to the non-normalized spelling of the input.

For example, consider the following input text:

```
The coldest recorded temperature is -89.2 degrees Celsius in Antarctica on July 21, 1983!
```

The service returns audio timings for the following strings:

"*The*", "*coldest*", "*recorded*", "*temperature*", "*is*", "*-89.2*", "*degrees*", "*Celsius*", "*in*", "*Antarctica*", "*on*", "*July*", "*21,*", "*1983!*"

Although "*-89.2*" is spoken in the audio as five separate words (*minus*, *eighty*, *nine*, *point*, *two*), the text message provides timing information for the string as a single unit with the start time of *minus* and the end time of *two*.

As in the previous example, non-normalized strings can also contain punctuation. The service includes the punctuation that precedes or follows a word in the text message that it returns with the timings. For instance, the strings "*21,*" and "*1983!*" include punctuation that the service returns in its text message. Although the punctuation results in silence, the audio timing for the word does *not* include that silence.

For example, consider input text that contains the following conditional statement:

```
If it is sunny, I will go to the beach.
```

The service returns timing information for all strings of the input, including " *sunny,*" and "*beach.*", both of which end in punctuation that produces silence. But the timing information for "*sunny,*" does not include the silence that is produced by the comma, and the timing information for "*beach.*" does not include the silence for the period. The information reflects only the timing of the spoken strings.

Timings for SSML text

When the service synthesizes plain text, it returns all input characters except for blank spaces as part of the strings in its word timing response. The same is not true of SSML, since some SSML elements do not generate audio. The following list summarizes the SSML elements that can impact word timing information:

- `<say-as>` indicates how the text that is enclosed between the opening and closing `<say-as>` tags is to be handled in the normalization step. Attributes specify how the embedded text is to be spoken. The following example indicates how the date is to be spoken:

  ```
  The baby was born on <say-as interpret-as="date" format="mdy">3/4/2016</say-as>.
  ```

  The service returns timing information for the following strings: " *The*", "*baby*", "*was*", "*born*", "*on*", "*3/4/2016.*" The service normalizes the string "*3/4/2016*" as "*march fourth two thousand sixteen*". The word timing information for the string reflects the start time of " *march*" and the end time of "*sixteen*".

  The following example indicates that the word `Hello` is to be spelled out:

  ```
  <say-as interpret-as="letters">Hello</say-as>.
  ```

  The service returns timing information for the string " *Hello.*". The service spells the word letter-by-letter during the normalization step. The word timing information in the response reflects the start time of the letter "*h*" and the end time of the letter " *o*".

- `<phoneme>` provides a pronunciation for the text that is enclosed in the opening and closing `<phoneme>` tags. But both the text and the closing tag are optional. The following example includes embedded text and a closing tag:

  ```
  The <phoneme alphabet="ibm" ph=".0tx.1me.0fo">tomato</phoneme> was ripe.
  ```

  The service returns timing information for the following strings: " *The*", "*tomato*", "*was*", "*ripe.*"

  Conversely, the following example provides a unary `<phoneme>` element with no embedded text and no closing tag:

  ```
  The <phoneme alphabet="ibm" ph=".0tx.1me.0fo"/> was ripe.
  ```

  In this case, the service returns timing information for the following strings: " *The*", "*<phoneme>*", "*was*", "*ripe.*"

- `<sub>` substitutes the text that is included in the element's `alias` attribute for the text that is enclosed between the opening and closing `<sub>` tags in the spoken audio. For example, the following input includes a single `<sub>` tag:

```
I work at <sub alias="International Business Machines">IBM</sub>.
```

The service produces timing information for the following strings: " *I*", "*work*", "*at*", "*IBM.*". The service normalizes the string " *IBM*" as "*International Business Machines*". The timing information for the string reflects the start time of "*International*" and the end time of " *Machines*".

- `<break>` inserts a pause in the spoken text. The service reflects the resulting silence in the word timings as a gap between the end time of the word that precedes the `<break>` element and the start time of the word that follows the element.
- `<paragraph>` (or `<p>`) can add silence to the audio. The service does not return timing information for the silence.
- `<sentence>` (or `<s>`) can add silence to the audio. The service does not return timing information for the silence.

SSML elements that are not mentioned in the list do not impact word timing information. For more information about the service's support for SSML, see Understanding SSML.

Examples with mark elements

The following examples show a simple WebSocket session between a client and the service. The examples focus on the exchange of data, not on opening the connection. The client sends a text message that includes two `<mark>` elements, named `SIMPLE` and `EXAMPLE`, and it requests the audio to be returned in WAV format:

```
{
  "text": "This is a <mark name=\"SIMPLE\"/>simple <mark name=\"EXAMPLE\"/> example.",
  "accept": "audio/wav"
}
```

The service first sends a message to confirm the audio format. It then sends multiple messages with the results. The service cannot guarantee the number of audio chunks that it sends to the client or the order in which the text and audio messages are delivered.

Both of the following responses are possible. In each case, the service sends two text messages that identify the locations of the marks in the binary stream. But it sends an arbitrary number of binary messages that contain the audio. The timing information for a mark always arrives before the audio chunk that contains the location of the mark.

- *In the first example response,* the text messages are interspersed with multiple audio messages:

```
{
  "binary_streams": [
    {
      "content_type": "audio/wav"
    }
  ]
}
... One or more chunks of binary audio.
    All audio precedes the SIMPLE mark....
{
  "marks": [
    [
      "SIMPLE", 0.784
    ]
  ]
}
... One or more chunks of binary audio audio can precede
    and follow the SIMPLE mark.
    All audio precedes the EXAMPLE mark....
{
  "marks": [
    [
      "EXAMPLE", 1.003
    ]
  ]
}
... One or more chunks of binary audio.
    Audio can precede and follow the EXAMPLE mark....
```

- *In the second example response,* the text messages arrive before any of the audio messages:

```
{
  "binary_streams": [
```

```
    "content_type": "audio/wav"}
  ]
}
{
  "marks": [
    [
      "SIMPLE", 0.784
    ]
  ]
}
{
  "marks": [
    [
      "EXAMPLE", 1.003
    ]
  ]
}
... One or more chunks of binary audio....
```

Customizing the service

Understanding customization

When you synthesize text with IBM Watson® Text to Speech, the service applies language-dependent pronunciation rules. The service applies the rules to convert the ordinary (orthographic) spelling of each word to a phonetic spelling. A word's phonetic spelling uses phoneme symbols to define how the word is pronounced. These symbols are the distinct units of sound that distinguish words in a language, the boundaries between syllables, and the stress marks for the syllables.

The service's regular pronunciation rules work well for common words. However, they can yield imperfect results for unusual words. Such words include domain-specific terms, words with foreign origins, personal or geographic names, and abbreviations or acronyms. If your application's lexicon includes such words, you can use the customization interface to specify how the service pronounces the words.

Status and support

The following status and support information applies to customization:

- Customization is available for all languages.
- IBM Cloud You must have the Standard or Premium pricing plan to use customization. Users of the Lite plan cannot use the customization interface. For more information, see the Text to Speech service in the IBM Cloud® Catalog.
- IBM Cloud Premium customers can work with IBM to train a new custom voice for their specific application needs. To request a custom voice or for more information, complete and submit this IBM Request Form.

How customization works

The customization interface of the Text to Speech service creates a dictionary of words and their translations for a specific language. This dictionary is referred to as a *custom model*. Each custom entry in a custom model consists of a *word*/*translation* pair. A word's translation tells the service how to pronounce the word when it occurs in input text.

The customization interface provides methods to create and manage your custom models, which the service stores permanently. After you create a custom model, you can use it during synthesis with any version of the `/v1/synthesize` method. When the service synthesizes input text, it determines the pronunciation of words that appear in the custom model by applying their translations either directly or indirectly. Because you create a custom model for a specific language, a custom model can be used with any voice that is available in that language.

You specify the translation for a word in a custom model as a *sounds-like translation* or a *phonetic translation*. You can use both methods for entries in the same custom model, and you can mix the two methods within the same translation. A number of rules and guidelines apply to custom entries. For more information, see Rules for creating custom entries.

Sounds-like translation

Sounds-like translation uses the service's regular pronunciation rules to represent the pronunciation of a target word indirectly. A sounds-like translation is formed from the regular pronunciations of one or more other words. The service first substitutes the specified translation for any occurrence of the word that appears in the input text. It then applies its regular pronunciation rules to the translation, converting the translation to its phonetic representation to obtain the pronunciation.

For example, the service's regular pronunciation rules properly translate many common abbreviations and acronyms. The service pronounces the abbreviation *cm* as *centimeter*. It pronounces less common abbreviations letter by letter. For instance, the service pronounces the string *Str* (an abbreviation for *street*) as *S T R*, with each letter pronounced individually. You can use the sounds-like method to specify the translation *street* for the string *Str*.

Another example of an acronym is the word *IEEE*, which stands for Institute of Electrical and Electronic Engineers. By default, the service pronounces this acronym as *I E E E*. But the acronym is commonly pronounced *I triple E*, which you can easily define by using the simple sounds-like translation *I triple E*. If the word *IEEE* appears in your custom model with this translation, the service substitutes each occurrence of the word with the translation. It then applies its regular pronunciation rules to the individual words *I*, *triple*, and *E* to yield the common pronunciation.

You can apply the sounds-like method to more than just abbreviations and acronyms. It works equally well for complex or unusual words. For example, the following pair of sounds-like translations yields correct pronunciations for unusual words that are handled imperfectly by the service's regular pronunciation rules. Finding proper translations for such words can be more challenging than for simple abbreviations. The following examples use the regular pronunciation rules to alter the words' spelling for translation:

- *Word*: `ayurvedic`, *Translation*: `aayervedic`
- *Word*: `gastroenteritis`, *Translation*: `gastro enteritis`

As these examples show, developing sounds-like translations can be more trial-and-error than formulaic. You create a candidate translation based on your

intuition and experience with the service. You then synthesize the word for the candidate translation as input text and listen to the resulting audio. If you are satisfied with the pronunciation, you can use the translation in your custom model; otherwise, you modify the translation and test it again.

Phonetic translation

The sounds-like method is a relatively simple and useful way of achieving a pronunciation. But it is not always possible to develop sounds-like translations. The direct alternative, the phonetic method, might appear to be more complicated and time-consuming, but it can achieve the pronunciation of any word.

Phonetic translation specifies a pronunciation in terms of phoneme symbols, syllable stress marks, and optional syllable boundaries that override the service's regular pronunciation rules. You specify a phonetic translation in one of the following formats:

- The standard International Phonetic Alphabet (IPA) representation
- The proprietary IBM Symbolic Phonetic Representation (SPR)

In either case, you specify a translation by using a specific phoneme format that is based on the Speech Synthesis Markup Language (SSML). SSML is an XML-based markup language that provides annotations of text for speech-synthesis applications. You specify the phonetic translation for a word by using the SSML `<phoneme>` element:

```
<phoneme alphabet="{ipa | ibm}" ph="{translation}"></phoneme>
```

The `alphabet` attribute specifies the phonetic representation type: `ipa` or `ibm`. The `ph` attribute specifies the phonetic translation string.

For example, consider the word `trinitroglycerin`. The service's regular pronunciation rules produce a pronunciation that differs from the one commonly used by chemists and physicians. The correct pronunciation can be achieved with a phonetic translation:

- *IPA*: `tɹaɪn'aɪtɹəglɪsəɹɨn`
- *SPR*: `trYn1YtrxglIsxrXn`

In these examples, the phonetic translation string is composed of phoneme symbols and a single primary stress mark. The primary stress mark is represented by `'` in IPA and by `1` in SPR. It is placed just before the symbol for the stressed vowel in both cases. Although the examples do not show it, you can also specify syllable boundaries and secondary stress positions in a phonetic translation. These elements are not required and normally are not needed to achieve a pronunciation. As with sounds-like translations, you can compose a phonetic translation from multiple strings that are delimited by spaces.

> **Note:** You can also specify IPA translations as IPA Unicode values. For more information, see Understanding phonetic symbols and the language-specific tables on the pages that are referred to from Phonetic symbols for supported languages. For an example translation that uses IPA Unicode values, see The phoneme element.

Working with an existing phonetic translation

Unless you are an expert in phonology, composing phonetic translations is not an easy task. It is always easier to edit an existing phonetic translation than to compose one from scratch. To help you create phonetic translations, the service's API includes a `GET /v1/pronunciation` method. The method returns the IPA or SPR representation that is generated by the service's regular pronunciation rules for a word in a specified language. You can also request the pronunciation for a word from a specified custom model to see the translation in the language of that model.

You can use the `/GET v/1/pronunciation` method to obtain an initial phonetic translation for a word. You can then modify the translation to achieve the pronunciation that you want. As with the sounds-like method, you follow a trial-and-error process. You submit your candidate translation to the service, synthesize the word as input text, listen to the resulting audio, and edit the candidate translation. You can repeat the process until you are satisfied with the pronunciation.

For more information, see Querying a word from a language.

More information about phonetic translation

The following resources provide information about phonetic translation:

- For more information about using SSML and its `<phoneme>` element, see Understanding SSML.
- For more information about specifying SPR and IPA symbols and translations, see Understanding phonetic symbols.

Mixed sounds-like and phonetic translation

You can mix the sounds-like and phonetic methods in the same translation. This feature can reduce the work that is involved in composing a translation.

For instance, assume that you used the sounds-like method to get part of a word pronounced satisfactorily. But you now need to fine-tune the remaining elements of the word. You can use the phonetic method to specify the difficult aspects of the word. The following example applies mixed translation to the word `trinitroglycerin`:

```
try<phoneme alphabet="ipa" ph="nˈaɪtʊəglɪsəɹɨn"></phoneme>
```

Rules for creating custom entries

The following rules and guidelines apply to populating a custom model with custom entries (word/translation pairs).

Maximum custom entries and limits

The following limits apply to all custom models and entries:

- A word in a custom entry can contain a maximum of 49 characters.
- A translation in a custom entry can contain a maximum of 499 characters.
- A custom model can include a maximum of 20,000 custom entries.
- A custom model can include a maximum of 1000 custom prompts.
- Do not use backslashes, slashes, colons, equal signs, ampersands, or question marks in the name.

Character encoding

The service accepts ASCII and UTF-8 character encoding for *word* and *translation* entries. For translations, use ASCII encoding for SPR notations and UTF-8 encoding for IPA notations.

White space

A word cannot include white space. The service uses white space to delineate individual words in the input text.

Case-sensitivity

A word is case-sensitive. For example, assume that a custom model contains the entry `{word='Sun', translation='Sunday'}`. The service applies its default pronunciation to the word `sun` but the custom translation to the word `Sun`, since only the latter has an initial capital letter.

To apply a custom translation to a word that might appear with or without initial capitalization, create two entries for both possible occurrences. Include both entries only if the translation is to be applied to both forms of the word.

Context sensitivity

The pronunciations of some words are context-sensitive. For example, consider the following example input sentence:

```
St. Anthony lives on Henry St.
```

The service's default pronunciation rules correctly synthesize this text as

```
Saint Anthony lives on Henry Street
```

However, if you override the default pronunciation rules for the string `St.` to translate it as `saint`, the service can no longer pronounce the word based on context. Applying a custom model that includes such a translation causes the service to pronounce the previous input sentence as

```
Saint Anthony lives on Henry saint
```

Consider such cases when you develop word/translation pairs.

Trailing periods

The service applies a word from a custom model only to those strings in the input text that match the word exactly. A trailing `.` (period) in a word entry changes how the word is synthesized:

- *A word that does not have a trailing period* can contain practically any character. Characters include letters, digits, punctuation (other than a trailing period), non-letter symbols (such as `%`, `&`, and `@`), quotation marks, parentheses, brackets, and so on. Its *translation* can include any legal input to the service, including white space and a phonetic representation in SSML format.
- *A word that has a trailing period* can contain only letters, periods, and internal apostrophes (not as the first or last character). The word's *translation* can contain only normal words with ordinary spelling that are separated by white space or hyphens. It cannot contain a phonetic representation.

An example of a word with a trailing period is "`div.`". Assume that a custom model includes the entry `{word='div.', translation='division'}`. The service does not apply the translation to the string "`div`" because it does not include a trailing period and therefore does not match the entry.

Phonetic translation for foreign words

One use of phonetic translation is to add pronunciations for words that are foreign to the base language of the custom model. For example, you might add a pronunciation for a French word to a custom model that is based on English. In this case, you must use the phonetic symbols for the language of the custom model, English.

The same phonetic symbol can produce different sounds for different languages. Also, not all phonetic symbols are supported for all languages. Make sure to use the phonetic symbols for the base language of the custom model when defining a translation.

Working with IBM SPR entries

Symbolic Phonetic Representation (SPR) is a proprietary, language-dependent format developed by IBM for specifying a word's pronunciation. For each supported language, SPR includes a phoneme alphabet, symbols for syllable boundaries, and symbols for levels of lexical stress. The following basic rules apply to creation of SPR entries:

- The default pronunciation that the customization interface returns for a word begins with a ``` ` ``` (back quote) and is enclosed in `[]` (square brackets). For example, the interface returns the following pronunciation for the word `tomato`:

  ```
  `[.0tx.1ma.0to]
  ```

  Omit the back quote and square brackets when you specify a word's translation with methods of the customization interface.

- You can use a period to indicate the beginning of a syllable in a translation, but periods are optional and do not influence the word's pronunciation. They appear in the pronunciation for a word only if you include them in the word's translation. Do not use spaces to indicate syllable boundaries.

- IBM recommends that you precede the vowel that has the primary stress for a word with a `1` symbol, though it is not strictly necessary. The service determines where stress occurs if you do not indicate it. You can also use a `2` symbol to indicate each secondary stress position, but the use of `2` symbols is also optional. They appear in the pronunciation for a word only if you include them in the word's translation.

For more information about working with SPR, see Understanding phonetic symbols.

Working with Japanese entries

Extra rules and a `part_of_speech` field apply to the creation of entries for words in a Japanese custom model:

- A sounds-like translation can contain only *Katakana* characters. *Kanji* and *Hiragana* characters are not allowed.

- When you create a translation (sounds-like or phonetic) for a word, you can also specify an optional `part_of_speech` field to identify the word's part of speech. The service uses the part of speech to produce the correct intonation for the word. For a complete list, see Japanese parts of speech.

- You can create only a single entry for any word, and you can specify only a single part of speech for any word. You cannot create multiple entries with different parts of speech (for instance, noun and verb) for the same word. Adding a translation for a word that exists in a model overwrites the word's existing translation, including its part of speech.

  > ⚠ **Important:** For improved naturalness of synthesized speech, do not create custom entries for long phrases. Create translations for single words or short phrases only. Note that other languages limit translation to single words only.

- The service applies the longest matching word from the word/translation pairs that are defined for a custom model. For example, consider the following three entries for a custom model.

  ```
  {
    "words": [
      {
        "word": "ＮＹ",
        "translation": "ニューヨーク",
        "part_of_speech": "Mesi"
      },
      {
        "word": "ＮＹＣ",
        "translation": "ニューヨークシティ",
        "part_of_speech": "Mesi"
      },
      {
        "word": "ＹＣ",
        "translation": "ヨコハマチューカガイ",
        "part_of_speech": "Mesi"
      }
  ```

```
 ]
}
```

With these entries, assume that the service receives the following input text: `一週間ＮＹＣを訪問した`. In this case, the service matches the word `ＮＹＣ` because `ＮＹＣ` is longer than `ＮＹ` and because `ＮＹＣ` matches before `ＹＣ`.

Japanese parts of speech

The following table lists the parts of speech that are supported for Japanese custom entries. For more information about specifying the part of speech for a Japanese custom entry, see Adding words to a Japanese custom model.

part_of_speech argument	Japanese meaning	English meaning
Dosi	*Doushi*	Verb
Fuku	*Fukishi*	Adverb
Gobi	*Gobi*	Inflection
Hoka	*Hoka*	Other (Words that have a special grammatical meaning of their own that does not fit into any other part of speech. For example, ありがとう for "thank you.")
Jodo	*Jodoushi*	Auxiliary verb
Josi	*Joshi*	Postpositional particle (For example, が の を for "of.")
Kato	*Kantoushi*	Interjection
Kedo	*Keiyodoushi*	Adjective verb
Keyo	*Keiyoshi*	Adjective (For example, 美し for "beautiful" or 明る for "bright.")
Kigo	*Kigou*	Symbol
Koyu	*Koyuumeishi*	Proper noun
Mesi	*Meishi*	Noun
Reta	*Rentaishi*	Determiner
Stbi	*Setsubiji*	Suffix
Stto	*Settoji*	Prefix
Stzo	*Setsuzokushi*	Conjunction
Suji	*Suuji*	Numeral

Japanese parts of speech

Creating and managing custom models

The first step in working with any custom model is to create it. Once it exists, you can manage the model by querying or updating its metadata and entries, and by deleting it if it is no longer needed. Before getting started, review the following general usage information about custom models.

Usage notes for customization

Consider the following guidelines when working with the customization interface.

Ownership of custom models

A custom model is owned by the instance of the Text to Speech service whose credentials are used to create it. You must use credentials for that service instance with methods of the customization interface to work with the custom model in any way.

All credentials obtained for the same instance of the Text to Speech service share access to all custom models created for that service instance. To restrict access to a custom model, create a separate instance of the service and use only the credentials for that service instance to create and work with the model. Credentials for other service instances cannot affect the custom model.

An advantage of sharing ownership across credentials is that you can cancel a set of credentials, for example, if they become compromised. You can then create new credentials for the same service instance and still maintain ownership of and access to custom models created with the original credentials.

Information security

The service allows you to associate a customer ID with data that is added or updated for custom models. You can associate a customer ID with custom words by passing the `X-Watson-Metadata` header with the following methods. If necessary, you can then delete the data associated with the customer ID by using the `DELETE /v1/user_data` method.

- `POST /v1/customizations/{customization_id}`
- `POST /v1/customizations/{customization_id}/words`
- `PUT /v1/customizations/{customization_id}/words/{word}`

In addition, if you delete an instance of the Text to Speech service from the IBM Cloud console, all data associated with that service instance is automatically deleted. This includes all custom models and word/translation pairs. This data is purged automatically and regardless of whether a customer ID is associated with the data.

For more information, see Information security.

Request logging and data privacy

IBM Cloud

How the service handles request logging for calls to the customization interface depends on the request:

- The service *does not* log data (words and translations) that are used to build custom models. You do not need to set the `X-Watson-Learning-Opt-Out` request header when using the customization interface to manage the words and translations in a custom model. Your training data is never used to improve the service's base models.
- The service *does* log data when a custom model is used with a synthesize request. You can opt out of request logging at the account level or by setting the `X-Watson-Learning-Opt-Out` request header to `true`.

For more information, see Request logging.

Creating a custom model

To create a new custom model, use the `POST /v1/customizations` method. A new model is always empty when you first create it. You must use other methods to populate it with word/translation pairs. The new custom model is owned by the service instance whose credentials are used to create it. For more information, see Ownership of custom models.

You pass the following attributes as a JSON object with the body of a `POST /v1/customizations` request:

`name` (*required* string)

A user-defined name for the new custom model. Use a localized name that matches the language of the custom model and describes the domain of the model, such as `Medical custom model` or `Legal custom model`.

- Include a maximum of 256 characters in the name.
- Do not use backslashes, slashes, colons, equal signs, ampersands, or question marks in the name.
- Use a name that is unique among all custom models that you own.

`language` (*optional* string)

An identifier for the language of the custom model. The default is `en-US` for US English. The custom model can be used with any voice that is available in the specified language. For example, a custom model that is created for the `en-US` language can be used with any US English voice. It cannot, however, be used with an `en-GB` voice.

`description` (*optional* string)

A recommended description of the new custom model.

- Use a localized description that matches the language of the custom model.
- Include a maximum of 128 characters in the description.

The following example example creates a new US English custom model named `Test`. The required `Content-Type` header identifies the type of the input as `application/json`.

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--data "{\"name\":\"Test\", \"language\":\"en-US\", \"description\":\"Customization test\"}" \
"{url}/v1/customizations"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--data "{\"name\":\"Test\", \"language\":\"en-US\", \"description\":\"Customization test\"}" \
"{url}/v1/customizations"
```

The method returns a JSON object that contains a globally unique identifier (GUID) for the new model. The GUID is used as the `customization_id` parameter in calls to access the model, such as those for querying, modifying, and using the model and its words.

```
{
  "customization_id": "64f4807f-a5f1-5867-924f-7bba1a84fe97"
}
```

Querying a custom model

To query information about an existing custom model, use the `GET /v1/customizations/{customization_id}` method. This is the most direct means of seeing all of the information about a model, including its metadata and the word/translation pairs and custom prompts that it contains.

IBM Cloud

```
$ curl -X GET -u "apikey:{apikey}" \
"{url}/v1/customizations/{customization_id}"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X GET \
--header "Authorization: Bearer {token}" \
"{url}/v1/customizations/{customization_id}"
```

The method returns its results as a JSON object of the following form:

```
{
  "customization_id": "64f4807f-a5f1-5867-924f-7bba1a84fe97",
  "owner": "297cfd08-330a-22ba-93ce-1a73f454dd98",
  "created": "2016-07-15T18:12:31.743Z",
  "name": "Test",
  "language": "en-US",
  "description": "Customization test",
  "last_modified": "2016-07-15T18:12:31.743Z",
  "words": [],
  "prompts": []
}
```

In addition to the information entered when the model was created, the output includes the credentials of the model's owner, the model's language, and the times at which the model was created and last modified. Because the model has not been modified since its creation, the two times in the example are the same.

The output also includes a `words` array that lists the model's custom words and a `prompts` array that lists the model's custom prompts. Because the model

has yet to be updated, the arrays in the example are empty.

Querying all custom models

To see information about all of the custom models that you own, use the `GET /v1/customizations` method:

IBM Cloud

```
$ curl -X GET -u "apikey:{apikey}" \
"{url}/v1/customizations"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X GET \
--header "Authorization: Bearer {token}" \
"{url}/v1/customizations"
```

The method returns a JSON array that includes an object for each custom model owned by the requester. The owner's credentials are shown in the `owner` field.


```
{
  "customizations": [
    {
      "customization_id": "64f4807f-a5f1-5867-924f-7bba1a84fe97",
      "owner": "297cfd08-330a-22ba-93ce-1a73f454dd98",
      "created": "2016-07-15T19:15:17.926Z",
      "name": "Test",
      "language": "en-US",
      "description": "Customization test",
      "last_modified": "2016-07-15T19:15:17.926Z"
    },
    {
      "customization_id": "63f5807f-a4f2-5766-914e-7abb1a84fe97",
      "owner": "297cfd08-330a-22ba-93ce-1a73f454dd98",
      "created": "2016-07-15T18:12:31.743Z",
      "name": "Test Two",
      "language": "en-US",
      "description": "Second customization test",
      "last_modified": "2016-07-15T18:23:50.912Z"
    }
  ]
}
```


The `created` and `last_modified` times for the first model are the same because it has yet to be updated. The times for the second model are different, indicating that it has been changed since its initial creation. The information does not include the custom entries defined for the models.

Updating a custom model

To update information about a custom model, use the `POST /v1/customizations/{customization_id}` method. You specify the updates as a JSON object. In addition to modifying its name and description, you can also use this method to add or update word/translation pairs in the model. You cannot change a model's language once it is created.

The following example updates the name and description of a custom model. An empty JSON array is sent with the `words` parameter to indicate that the model's entries are to remain unchanged.

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--data "{\"name\":\"Test Update\", \"description\":\"Customization test update\", \"words\":[]}" \
"{url}/v1/customizations/{customization_id}"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
```

```
--header "Content-Type: application/json" \
--data "{\"name\":\"Test Update\", \"description\":\"Customization test update\", \"words\":[]}" \
"{url}/v1/customizations/{customization_id}"
```

For information about updating the words in a model, see Adding multiple words to a custom model.

Deleting a custom model

To discard a custom model that you no longer need, use the `DELETE /v1/customizations/{customization_id}` method. Use this method only if you are sure that you no longer need the model, since deletion is permanent.

IBM Cloud

```
$ curl -X DELETE -u "apikey:{apikey}" \
"{url}/v1/customizations/{customization_id}"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X DELETE \
--header "Authorization: Bearer {token}" \
"{url}/v1/customizations/{customization_id}"
```

Creating and managing custom entries

Once a custom model exists, the next step is to add custom entries in the form of word/translation pairs to define how specified words are to be pronounced during synthesis. The definitions override the service's default regular pronunciation rules. You can add and query translations for one or more words at a time, and you can delete individual words that you no longer need. Once you are familiar with the customization interface, manipulating multiple words at once can be more convenient than working on a word-by-word basis. You must use credentials for the instance of the service that owns a custom model to use any method that requires its customization ID.

Note: For more information about the rules and limits that apply to custom entries, see Rules for creating custom entries.

Important: Do not use characters that need to be URL-encoded. For example, do not use spaces, slashes, backslashes, colons, ampersands, double quotes, plus signs, equals signs, question marks, etc. in the name. The service does not prevent the use of these characters, but because they must be URL-encoded wherever used, it is strongly discouraged.

Adding a single word to a custom model

To add a single word/translation pair to a custom model, use the `PUT /v1/customizations/{customization_id}/words/{word}` method. You specify the word to be added in the URL of the method. You provide the translation for the word as a JSON object with a single `translation` attribute. Adding a new translation for a word that already exists in a model overwrites the word's existing translation.

You can provide a translation by using the sounds-like or the phonetic method (or a combination of the two). For phonetic translations, you can use either the International Phonetic Alphabet (IPA) or the IBM Symbolic Phonetic Representation (SPR) format. The following examples use different approaches to add equivalent translations for the word `IEEE` to a custom model. The required `Content-Type` header identifies the type of the input as `application-json`.

- **Sounds-like:** For this example, the sounds-like method is the simplest approach:

  IBM Cloud

  ```
  $ curl -X PUT -u "apikey:{apikey}" \
  --header "Content-Type: application/json" \
  --data "{\"translation\":\"I triple E\"}" \
  "{url}/v1/customizations/{customization_id}/words/IEEE"
  ```

  IBM Cloud Pak for Data IBM Software Hub

  ```
  $ curl -X PUT \
  --header "Authorization: Bearer {token}" \
  --header "Content-Type: application/json" \
  --data "{\"translation\":\"I triple E\"}" \
  "{url}/v1/customizations/{customization_id}/words/IEEE"
  ```

- **Phonetic IPA:** IPA requires use of the `<phoneme>` element with the `alphabet` attribute set to `ipa` and the `ph` attribute defined in IPA format:

  IBM Cloud

  ```
  $ curl -X PUT -u "apikey:{apikey}" \
  --header "Content-Type: application/json" \
  --data "{\"translation\":\"<phoneme alphabet=\\\"ipa\\\" ph=\\\"aɪ.tʊˈɨpəl.ˈi\\\"></phoneme>\"}" \
  "{url}/v1/customizations/{customization_id}/words/IEEE"
  ```

  IBM Cloud Pak for Data IBM Software Hub

  ```
  $ curl -X PUT \
  --header "Authorization: Bearer {token}" \
  --header "Content-Type: application/json" \
  --data "{\"translation\":\"<phoneme alphabet=\\\"ipa\\\" ph=\\\"aɪ.tʊˈɨpəl.ˈi\\\"></phoneme>\"}" \
  "{url}/v1/customizations/{customization_id}/words/IEEE"
  ```

- **Phonetic IBM SPR:** SPR uses the `<phoneme>` element with the `alphabet` attribute set to `ibm` and the `ph` attribute defined in SPR format:

  IBM Cloud

  ```
  $ curl -X PUT -u "apikey:{apikey}" \
  --header "Content-Type: application/json" \
  --data "{\"translation\":\"<phoneme alphabet=\\\"ibm\\\" ph=\\\"1Y.tr1Ipxl.1i\\\"></phoneme>\"}" \
  "{url}/v1/customizations/{customization_id}/words/IEEE"
  ```

  IBM Cloud Pak for Data IBM Software Hub

  ```
  $ curl -X PUT \
  --header "Authorization: Bearer {token}" \
  --header "Content-Type: application/json" \
  --data "{\"translation\":\"<phoneme alphabet=\\\"ibm\\\" ph=\\\"1Y.tr1Ipxl.1i\\\"></phoneme>\"}" \
  "{url}/v1/customizations/{customization_id}/words/IEEE"
  ```

Adding multiple words to a custom model

To add one or more words to a custom model at one time, use the `POST /v1/customizations/{customization_id}/words` method. You specify the entries to be added to the custom model as a JSON array of word/translation pairs. The required `Content-Type` header identifies the type of the input as `application-json`.

The following example adds common sounds-like translations for the words `NCAA` and `iPhone` to a custom model:

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--data "{\"words\": [{\"word\":\"NCAA\", \"translation\":\"N C double A\"}, {\"word\":\"iPhone\", \"translation\":\"I phone\"}]}" \
"{url}/v1/customizations/{customization_id}/words"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--data "{\"words\": [{\"word\":\"NCAA\", \"translation\":\"N C double A\"}, {\"word\":\"iPhone\", \"translation\":\"I phone\"}]}" \
"{url}/v1/customizations/{customization_id}/words"
```

The JSON content sent in the request body equates to the following:

```
{
  "words": [
    {"word":"NCAA", "translation":"N C double A"},
    {"word":"iPhone", "translation":"I phone"}
  ]
}
```

As mentioned in Updating a custom model, you can also use the `POST /v1/customizations/{customization_id}` method to add words to a custom model. The following example uses this method to add the same two words as the previous example; it makes no changes to the model's metadata. With the exception of the URL, the two methods are identical.

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--data "{\"words\": [{\"word\":\"NCAA\", \"translation\":\"N C double A\"}, {\"word\":\"iPhone\", \"translation\":\"I phone\"}]}" \
"{url}/v1/customizations/{customization_id}"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--data "{\"words\": [{\"word\":\"NCAA\", \"translation\":\"N C double A\"}, {\"word\":\"iPhone\", \"translation\":\"I phone\"}]}" \
"{url}/v1/customizations/{customization_id}"
```

Adding words to a Japanese custom model

Additional considerations and an additional `part_of_speech` field apply when creating entries for words in a Japanese custom model; for more information, see Working with Japanese entries. Specify a part of speech for a Japanese custom entry as follows:

- For the `PUT /v1/customizations/{customization_id}/words/{word}` method, pass a JSON object of the following form:

```
{
  "translation": "value",
  "part_of_speech": "value"
}
```

- For the `POST /v1/customizations/{customization_id}/words` and `POST customizations/{customization_id}` methods, pass a JSON object like the following:

```
{
  "words": [
    {
      "word": "value",
      "translation": "value",
      "part_of_speech": "value"
    }
    . . .
  ]
}
```

The following examples of the `PUT /v1/customizations/{customization_id}/words/{word}` method translate the URL-encoded, double-byte string for `NY` to the noun (`Mesi`) `ニューヨーク` (`New York` in English). If this custom translation is not defined, the string is read as `enu wai`.

- **Sounds-like:**

  IBM Cloud

  ```
  $ curl -X PUT -u "apikey:{apikey}" \
  --header "Content-Type: application/json" \
  --data "{\"translation\":\"ニューヨーク\", \"part_of_speech\":\"Mesi\"}" \
  "{url}/v1/customizations/{customization_id}/words/%EF%BC%AE%EF%BC%B9"
  ```

  IBM Cloud Pak for Data IBM Software Hub

  ```
  $ curl -X PUT \
  --header "Authorization: Bearer {token}" \
  --header "Content-Type: application/json" \
  --data "{\"translation\":\"ニューヨーク\", \"part_of_speech\":\"Mesi\"}" \
  "{url}/v1/customizations/{customization_id}/words/%EF%BC%AE%EF%BC%B9"
  ```

- **Phonetic IPA:**

IBM Cloud

```
$ curl -X PUT -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--data "{\"translation\":\"<phoneme alphabet=\\\"ipa\\\" ph=\\\"ɲɯːjoːkɯ\\\"></phoneme>\", \"part_of_speech\":\"Mesi\"}" \
"{url}/v1/customizations/{customization_id}/words/%EF%BC%AE%EF%BC%B9"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X PUT \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--data "{\"translation\":\"<phoneme alphabet=\\\"ipa\\\" ph=\\\"ɲɯːjoːkɯ\\\"></phoneme>\", \"part_of_speech\":\"Mesi\"}" \
"{url}/v1/customizations/{customization_id}/words/%EF%BC%AE%EF%BC%B9"
```

- **Phonetic IBM SPR:**

IBM Cloud

```
$ curl -X PUT -u "apikey:{apikey}" \
--header "Content-Type: application/json" \
--data "{\"translation\":\"<phoneme alphabet=\\\"ibm\\\" ph=\\\"nyu:yo:ku\\\"></phoneme>\", \"part_of_speech\":\"Mesi\"}" \
"{url}/v1/customizations/{customization_id}/words/%EF%BC%AE%EF%BC%B9"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X PUT \
--header "Authorization: Bearer {token}" \
--header "Content-Type: application/json" \
--data "{\"translation\":\"<phoneme alphabet=\\\"ibm\\\" ph=\\\"nyu:yo:ku\\\"></phoneme>\", \"part_of_speech\":\"Mesi\"}" \
"{url}/v1/customizations/{customization_id}/words/%EF%BC%AE%EF%BC%B9"
```

Querying a single word from a custom model

To query the translation of a single word from a custom model, use the `GET /v1/customizations/{customization_id}/words/{word}` method. You specify the word in the URL of the method. The translation is returned as it is defined in the custom model (sounds-like or phonetic).

The following example queries a custom model for the translation of the word `IEEE`:

IBM Cloud

```
$ curl -X GET -u "apikey:{apikey}" \
"{url}/v1/customizations/{customization_id}/words/IEEE"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X GET \
--header "Authorization: Bearer {token}" \
"{url}/v1/customizations/{customization_id}/words/IEEE"
```

If the word has the sounds-like translation in the model, the example returns the following JSON output:

```
{
  "translation": "I triple E"
}
```

Querying all words from a custom model

To see the translations for all of the words defined in a custom model, use the `GET /v1/customizations/{customization_id}/words` method. The following example uses the method to list the entries from a custom model that contains sounds-like translations for three words:

IBM Cloud

```
$ curl -X GET -u "apikey:{apikey}" \
"{url}/v1/customizations/{customization_id}/words"
```

IBM Cloud Pak for Data | IBM Software Hub

```
$ curl -X GET \
--header "Authorization: Bearer {token}" \
"{url}/v1/customizations/{customization_id}/words"
```

The method returns a JSON array with the following data. For Japanese custom models, the output includes the part of speech for individual words.

```
{
  "words": [
    {
      "word": "IEEE",
      "translation": "I triple E"
    },
    {
      "word": "NCAA",
      "translation": "N C double A"
    },
    {
      "word": "iPhone",
      "translation": "I phone"
    }
  ]
}
```

As described in Querying a custom model, you can also use the `GET /v1/customizations/{customization_id}` method to see both the metadata and the words for a custom model:

IBM Cloud

```
$ curl -X GET -u "apikey:{apikey}" \
"{url}/v1/customizations/{customization_id}"
```

IBM Cloud Pak for Data | IBM Software Hub

```
$ curl -X GET \
--header "Authorization: Bearer {token}" \
"{url}/v1/customizations/{customization_id}"
```

The method returns JSON output of the following form. Because this and the previous example specify the same custom model, the array of words in their output is identical.

```
{
  "customization_id": "64f4807f-a5f1-5867-924f-7bba1a84fe97",
  "owner": "297cfd08-330a-22ba-93ce-1a73f454dd98",
  "created": "2016-07-15T19:15:17.926Z",
  "name": "Test",
  "language": "en-US",
  "description": "Customization test",
  "last_modified": "2016-07-15T19:46:01.387Z",
  "words": [
    {
      "word": "IEEE",
      "translation": "I triple E"
    },
    {
      "word": "NCAA",
      "translation": "N C double A"
    },
    {
      "word": "iPhone",
      "translation": "I phone"
    }
  ]
}
```

Querying a word from a language

To query the pronunciation of a word, use the `GET /v1/pronunciation` method. Specify a voice to get the pronunciation in the language of that voice. By default, the method returns the pronunciation based on the service's regular pronunciation rules, but you can also request the pronunciation for a specified custom model. The method includes four query parameters that let you specify the information that is to be returned:

- The required `text` parameter specifies the word whose pronunciation is to be returned.
- The optional `voice` parameter lets you specify the language of the pronunciation. You specify one of the voices (for example, `en-US_LisaV3Voice`) to indicate the desired language; all voices for the same language (for example, `en-US`) return the same pronunciation. By default, the pronunciation is returned for the language of the default voice. For more information, see Using the default voice.
- The optional `format` parameter lets you specify the phonetic format of the pronunciation, either `ipa` or `ibm`. By default, the pronunciation is returned in IPA format.
- The optional `customization_id` parameter lets you specify a custom model for which the pronunciation is to be returned. If the word is not defined in the specified custom model, the service returns the default pronunciation for the model's language. Omit the parameter to see the translation for the specified voice with no customization. Do not specify both a voice and a custom model.

This method is useful because it allows you to query a word from any language and, because it does not require a customization ID, places no restrictions on the words that you can see. It can prove especially helpful when composing a phonetic translation for a new word; you can use it to obtain the pronunciation for an existing word and then use that as the basis of your new translation, which is far more convenient than creating a translation from scratch.

The following example obtains the pronunciation for the word `IEEE` in the default IPA format:

IBM Cloud

```
$ curl -X GET -u "apikey:{apikey}" \
"{url}/v1/pronunciation?text=IEEE"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X GET \
--header "Authorization: Bearer {token}" \
"{url}/v1/pronunciation?text=IEEE"
```

The response shows the IPA symbols for the pronunciation:

```
{
  "pronunciation": ".'aɪ .'i .'i .'i"
}
```

The following example enters a sounds-like translation for the word `IEEE` and obtains the phonetic equivalent in IBM SPR format. Obtaining the phonetic pronunciation for a sounds-like translation is an especially interesting approach to composing a phonetic translation. The spaces of the word are URL-encoded in the example.

IBM Cloud

```
$ curl -X GET -u "apikey:{apikey}" \
"{url}/v1/pronunciation?text=i%20triple%20e&format=ibm"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X GET \
--header "Authorization: Bearer {token}" \
"{url}/v1/pronunciation?text=i%20triple%20e&format=ibm"
```

The response shows the SPR symbols for the pronunciation:

```
{
  "pronunciation": "1Y.tr1Ipxl.1i"
}
```

Deleting a word from a custom model

To delete a word from a custom model, use the `DELETE /v1/customizations/{customization_id}/words/{word}` method. You specify the word to be deleted in the

URL of the method. You can delete individual words only; you cannot delete multiple words with a single method.

The following example deletes the word `IEEE` from the specified custom model:

IBM Cloud

```
$ curl -X DELETE -u "apikey:{apikey}" \
"{url}/v1/customizations/{customization_id}/words/IEEE"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X DELETE \
--header "Authorization: Bearer {token}" \
"{url}/v1/customizations/{customization_id}/words/IEEE"
```

Using a custom model for speech synthesis

Once you create a custom model and populate it with custom entries, you use it by passing its customization ID (GUID) with the `customization_id` query parameter of the HTTP `GET` or `POST /v1/synthesize` method or the WebSocket `/v1/synthesize` method. When you include a customization ID, you must call a `synthesize` method with credentials for the instance of the service that owns the specified custom model.

Examples of using a custom model

The first two examples generate a custom pronunciation for `IEEE` that is based on entries from the indicated custom model. The custom pronunciation is used instead of the default pronunciation from the service's regular pronunciation rules.

- The HTTP `GET /v1/synthesize` method:

  IBM Cloud

  ```
  $ curl -X GET -u "apikey:{apikey}" \
  --header "Accept: audio/flac" \
  --output ieee.flac \
  "{url}/v1/synthesize?text=IEEE&customization_id={customization_id}"
  ```

  IBM Cloud Pak for Data IBM Software Hub

  ```
  $ curl -X GET \
  --header "Authorization: Bearer {token}" \
  --header "Accept: audio/flac" \
  --output ieee.flac \
  "{url}/v1/synthesize?text=IEEE&customization_id={customization_id}"
  ```

- The HTTP `POST /v1/synthesize` method:

  IBM Cloud

  ```
  $ curl -X POST -u "apikey:{apikey}" \
  --header "Content-Type: application/json" \
  --header "Accept: audio/flac" \
  --data "{\"text\":\"IEEE\"}" \
  --output ieee.flac \
  "{url}/v1/synthesize?customization_id={customization_id}"
  ```

  IBM Cloud Pak for Data IBM Software Hub

  ```
  $ curl -X POST \
  --header "Authorization: Bearer {token}" \
  --header "Content-Type: application/json" \
  --header "Accept: audio/flac" \
  --data "{\"text\":\"IEEE\"}" \
  --output ieee.flac \
  "{url}/v1/synthesize?customization_id={customization_id}"
  ```

The third example establishes a WebSocket connection with the `/v1/synthesize` method. The request uses the indicated custom model to synthesize text

that is passed over the connection.

```
var access_token = '{access_token}';
var wsURI = '{ws_url}/v1/synthesize'
  + '?access_token=' + access_token
  + '&voice=en-US_AllisonV3Voice'
  + '&customization_id=={customization_id}';
var websocket = new WebSocket(wsURI);
```

Using SSML

Understanding SSML

The Speech Synthesis Markup Language (SSML) is an XML-based markup language that provides annotations of text for speech-synthesis applications. It is a recommendation of the W3C Voice-Browser Working Group that has been adopted as the standard markup language for speech synthesis by the VoiceXML 2.0 specification. SSML provides developers of speech applications with a standard way to control aspects of the synthesis process by enabling them to specify pronunciation, volume, pitch, speed, and other attributes via markup. You can use SSML to control the synthesis of your text with all supported languages.

The IBM Watson® Text to Speech service bases its support on SSML version 1.1, which was recommended by W3C on 7 September 2010. For more information about the W3C SSML recommendation, see W3C Speech Synthesis Markup Language (SSML) Version 1.1 .

Introduction to SSML

SSML operates by augmenting the plain text that is passed to a synthesizer with a predefined set of elements, or tags. An XML parser first separates the plain input text from the markup specifications. The specifications are then processed and sent as a set of instructions in a form that can be understood by the synthesizer to produce the desired effects. For the XML parser to carry out this job, the markup needs to be well formed; for example, elements must be closed and multiple elements must be properly nested. For an introduction to basic XML concepts, see w3schools.com/xml/xml_whatis.asp.

An SSML element is anything contained within, and including, an opening tag and its matching closing tag. As shown in the following example, an element can contain a combination of other elements (tags can be nested) and text. Additionally, elements can require or optionally accept attributes set to particular values.

```
<tag1>
  <tag2 attributeName="attributeValue">
    ... some text ...
  </tag2>
</tag1>
```

A full legal SSML document consists of an XML prolog, which contains information such as encoding and the schema against which to validate the SSML document, followed by the root element, `<speak>`. Within the span of the `<speak>` element, you specify the text that is to be synthesized, augmented with additional elements.

```
<?xml version="1.0" encoding="UTF-8"?>


<speak version="1.1">
  ... the body that contains text to be synthesized plus markup ...
</speak>
```

Note: The XML prolog is not needed for text that you pass to the service. The service supports SSML fragments, which are SSML elements that do not include the full XML header and do not need to include their parent elements. For example, the `<?xml>` and `<speak>` elements are always optional for SSML that you send for synthesis.

SSML support

For more information about using SSML and related features with the service, see the following:

- The service implements most of the W3C specification and supports SSML fragments.
  - For complete information about the service's level of support for all SSML elements, see SSML elements.
  - For examples of using SSML elements with the `text` of a speech synthesis request, see Examples of input text.
- The service supports additional synthesis features for enhanced neural and expressive neural voices:
  - Modifying the speaking rate
  - Modifying the speaking pitch
- The service supports additional synthesis features for expressive neural voices:
  - Using speaking styles
  - Emphasizing interjections
  - Emphasizing words
- The service supports a synthesis feature that lets you control how alphanumeric strings are spelled out for German voices. For more information, see

Specifying how strings are spelled out.

- The service supports the use of the SSML `<mark>` element with the WebSocket interface to obtain timing information for words of the resulting audio. The WebSocket interface also allows you to request information for all strings of the input text. For more information, see
  - Specifying an SSML mark
  - Requesting word timings for all words
- The service's customization interface supports the use of the SSML `<phoneme>` element to specify the phonetic spelling that it uses to pronounce a word. The phonetic spelling represents the sounds of a word, how those sounds are divided into syllables, and which syllables receive stress.
  - For information about the customization interface, see Understanding customization.
  - For information about the valid symbols that you can use in an International Phonetic Alphabet (IPA) or the IBM Symbolic Phonetic Representation (SPR) specification for any supported language, see Understanding phonetic symbols.

SSML validation

The service validates all SSML elements that you submit in any content, either as input text for synthesis or as the definition of a word's translation for customization. The service cannot determine ahead of time whether text submitted for synthesis contains SSML elements. Therefore, it performs the same validation for all input text, regardless of whether it contains SSML.

Validation of elements and attributes

The service performs the following validation of SSML elements and attributes:

- *All SSML input must be correct and well formed.*
- *The service silently ignores unsupported SSML elements.* The service synthesizes the text that is contained within the tags of an unsupported element; only the element is ignored.
- *The service returns an HTTP 400 error code for invalid SSML elements or attributes.* When specifying SSML elements that require an attribute:
  - You must specify the required attribute.
  - You must specify a valid value for the required attribute.
  - If you include additional invalid attributes or values, they are ignored.

  If the request fails, the error response includes a descriptive message. For example, suppose you specify the following input text, which includes the `<say-as>` element:

  ```
  "text": "The price is <say-as interpret-as=\"currency\">$2,500.00</say-as>."
  ```

  In this example, the `interpret-as` attribute is required and is specified, but its value is invalid. A valid value for the attribute is `vxml:currency`, not `currency`. The error response includes the following message:

  ```
  The connection to the Watson Text to Speech service closed with the following error: Tag <say-as> has invalid attribute interpret-as=currency
  ```

Summary of SSML errors

Table 1 describes many common SSML validation errors. In each case, the request fails and the service returns a 400 error code.

Validation error	Description of problem
Invalid SSML element	For example, you specify a tag incorrectly, omit a required attribute, or include an opening tag but no matching closing tag.
Unescaped XML control characters	The input text itself contains a ", ', &, <, >, or / character instead of its equivalent escape string or character encoding. For more information, see Escaping XML control characters.
Invalid syllable stress	The `ph` attribute of a `<phoneme>` element for IBM SPR includes invalid syllable stress. For more information about indicating syllables and syllable stress, see Specifying syllables.
Invalid phonetic symbol	The `ph` attribute of a `<phoneme>` element includes an unsupported IPA or SPR symbol for the specified language.
No vowels	The `ph` attribute of a `<phoneme>` element specifies a word pronunciation that includes no vowels.

French liaison in invalid location	In the `ph` attribute of a `<phoneme>` element, the liaison character does not follow a consonant or it occurs in the middle of a word's pronunciation.
Japanese `:` symbol does not precede a vowel	In the `ph` attribute of a `<phoneme>` element in Japanese, a `:` character does not occur before a vowel (possibly with other symbols, such as syllable boundary, in between).
Invalid use of the SSML `<prosody>` element	You cannot use the `contour`, `duration`, `range`, and `volume` attributes of the `<prosody>` element with any voice.
Invalid use of the SSML `<express-as>` element	You can use the `<express-as>` element only with expressive neural voices.

SSML validation errors

SSML elements

With the IBM Watson® Text to Speech service, you can use most Speech Synthesis Markup Language (SSML) elements and attributes to control the synthesis of your text.

Supported elements and attributes

Table 1 summarizes the service's support for SSML elements and attributes:

- *Full* means that the service fully supports the element or attribute with its HTTP and WebSocket interfaces.
- *Partial* means that the service's support for the element or attribute is limited in one of the following ways:
  - The service supports only some aspects of the element or attribute.
  - The service supports the element or attribute with only some of its voices.
  - The service supports the element or attribute with only one of its interfaces, HTTP or WebSocket.
- *None* means that the service does not support the element or attribute.

The following sections provide descriptions of each element or attribute, including examples, restrictions, and whether the service's support differs from standard SSML. Support for some attributes and values differs slightly from the SSML specification. For more information, see W3C Speech Synthesis Markup Language (SSML) Version 1.1.

Element or attribute	Support	Element or attribute	Support
`<audio>` element	None	`<prosody>` element	Partial
`<break>` element	Full	• contour attribute	None
`<desc>` element	None	• duration attribute	None
`<emphasis>` element	Partial	• pitch attribute	Full
`<express-as>` element	Partial	• range attribute	None
`<lexicon>` element	None	• rate attribute	Full
`<mark>` element	Partial	• volume attribute	None
`<meta>` element	None	`<say-as>` element	Partial

<metadata> element	None	• interpret-as attribute	Partial
<paragraph> element	Full	<sentence> element	Full
<phoneme> element	Full	<speak> element	Full
		<sub> element	Full
		<voice> element	None

SSML elements and attributes

The <audio> element

This <audio> element inserts recorded elements into the service-generated audio. It is not supported.

The <break> element

The <break> element inserts a pause into the spoken text. It has the following optional attributes:

- strength specifies the length of the pause in terms of varying strength values:
  - none suppresses a break that might otherwise be produced during processing.
  - x-weak, weak, medium, strong, or x-strong insert increasingly stronger breaks.
- time specifies the length of the pause in terms of seconds or milliseconds. Valid value formats are {integer}s for seconds or {integer}ms for milliseconds.

```
Break size <break strength="none"/> no pause
Break size <break strength="x-weak"/> x-weak pause
Break size <break strength="weak"/> weak pause
Break size <break strength="medium"/> medium pause
Break size <break strength="strong"/> strong pause
Break size <break strength="x-strong"/> x-strong pause
Break size <break time="1s"/> one-second pause
Break size <break time="1500ms"/> 1500-millisecond pause
```

The <break> element is beta functionality for Natural voices.

Note: When used with the time attribute and Expressive or Natural voices, the <break> element introduces a pause of approximately the specified duration, though the exact timing may vary based on the voice and context.

The <desc> element

The <desc> element can occur only within an <audio> element. Because the <audio> element is not supported, neither is the <desc> element.

The <emphasis> element

Note: The <emphasis> element is supported for use only with the expressive neural voices.

With the expressive neural voices, you can use the <emphasis> element to emphasize or de-emphasize one or more words of the input text. The element supports an optional level attribute that accepts one of the following values:

- none - Prevents the service from emphasizing text that might otherwise be emphasized.
- moderate - Provides a noticeable amount of emphasis to the text. This level is the default if you omit the level attribute.
- strong - Provides a more significant amount of emphasis to the text than the moderate level provides.
- reduced - De-emphasizes the text by tending to reduce its significance in the audio. This level is the opposite of stressing the text.

The following example applies the moderate level to the word give:

```
I am going to <emphasis level="moderate">give</emphasis> her the book.
```

For more information, see Emphasizing words.

The <express-as> element

> **Note:** The <express-as> element is an SSML extension specific to the Text to Speech service. It is supported for use only with the expressive neural voices.

With the expressive neural voices, you can use the <express-as> element to apply speaking styles to enhance the service's emphasis of certain characteristics for all or part of the input text. The element supports a required style attribute that accepts one of the following speaking styles:

- cheerful - Expresses happiness and good news.
- empathetic - Expresses empathy and compassion.
- neutral - Expresses objectivity and evenness.
- uncertain - Expresses confusion and uncertainty.

The following example applies the cheerful style to the entire input text:

```
<express-as style="cheerful">Oh, that&apos;s good news! I&apos;m glad that we could help.</express-as>
```

For more information, see Using speaking styles.

The <lexicon> element

This <lexicon> element introduces pronunciation dictionaries for the given SSML document. It is not supported.

You can use the service's customization interface to define a dictionary of custom entries (word/translation pairs) for use during speech synthesis. For more information, see Understanding customization.

The <mark> element

> **Note:** The <mark> element is supported only by the service's WebSocket interface, not by its HTTP interface, which ignores the element. For more information, see Specifying an SSML mark.

> **Note:** The <mark> element is not supported for Natural voices.

The <mark> element is an empty element that places a marker into the text to be synthesized. The client is notified when all of the text that precedes the <mark> element has been synthesized. The element accepts a single name attribute that specifies a string that uniquely identifies the mark; the name must begin with an alphanumeric character. The name is returned along with the time at which the mark occurs in the synthesized audio.

```
Hello <mark name="here"/> world.
```

The <meta> and <metadata> elements

The <meta> and <metadata> elements are containers in which you can place information about the document. They are not supported.

The <paragraph> and <sentence> elements

The <paragraph> (or <p>) and <sentence> (or <s>) elements are optional elements that can be used to give hints about textual structure. If the text that is enclosed in a <paragraph> or <sentence> element does not end with an end-of-sentence punctuation character (like a period), the service adds a longer than normal pause to the synthesized audio.

The only valid attribute for either element is xml:lang, which allows for language switching. The attribute is not supported.

```
<paragraph>
  <sentence>Text within a sentence element.</sentence>
  <s>More text in another sentence.</s>
</paragraph>
```

The <phoneme> element

The <phoneme> element provides a phonetic pronunciation for the enclosed text. The phonetic spelling represents the sounds of a word, how the sounds

are divided into syllables, and which syllables receive stress. The element has two attributes:

- `alphabet` is an optional attribute that specifies the phonology to be used. The supported alphabets are
  - *The standard International Phonetic Alphabet (IPA):* `alphabet="ipa"`.
  - *The IBM Symbolic Phonetic Representation (SPR):* `alphabet="ibm"`.

  If no alphabet is specified, the service uses IBM SPR by default. For more information, see Understanding phonetic symbols.
- `ph` is a required attribute that provides the pronunciation in the indicated alphabet. The following examples show the pronunciation for the word *tomato* in both formats:
  - IPA format:

    ```
    <phoneme alphabet="ipa" ph="tə'meɪ.ɾoʊ">tomato</phoneme>
    ```

  - IPA format with Unicode symbols:

    ```
    <phoneme alphabet="ipa" ph="t&#x0259;&#x02C8;me&#x026A;.&#x027E;o&#x028A;">tomato</phoneme>
    ```

  - IBM SPR format:

    ```
    <phoneme alphabet="ibm" ph=".0tx.1me.0Fo">tomato</phoneme>
    ```

For more information about using SPR and IPA notations with the `<phoneme>` element, see Understanding phonetic symbols.

The <prosody> element

The `<prosody>` element controls the pitch and speaking rate of the text. All attributes are optional, but an error occurs if you do not specify at least one attribute with the element.

The service supports the following two attributes of the SSML specification:

- The `pitch` attribute
- The `rate` attribute

The SSML specification also offers four attributes that the service does not support:

- The `contour` attribute
- The `range` attribute
- The `duration` attribute
- The `volume` attribute

The service also supports query parameters that let you adjust the rate and pitch for all text of a speech synthesis request. For more information about the parameters and their interaction with the `pitch` and `rate` attributes of the `<prosody>` element, see

- Modifying the speaking rate
- Modifying the speaking pitch

Note: The `<prosody>` element is not supported for Natural voices.

Differences from the SSML version 1.1 specification

The Text to Speech service bases its SSML support on W3C Speech Synthesis Markup Language (SSML) Version 1.1. However, the SSML specification has evolved since the service was first released. To maintain backward-compatibility for users, the service continues to support some features of the `<prosody>` element that are different from the latest SSML specification.

- *For the `pitch` attribute,* the service supports the following additional features:
  - A relative change in percent indicated by a signed or unsigned number and followed by a `%` (percent sign). The default pitch for a voice is equivalent to passing a value of `0%`.
  - A relative change in semitones indicated by a signed or unsigned number and followed by the string `st`.
- *For the `rate` attribute,* the service supports the following additional features:
  - A relative change in percent indicated by a signed or unsigned number and followed by a `%` (percent sign). The default speaking rate for a

voice is equivalent to passing a value of `0%`.
    - A number with no unit designation specifies the number of words per minute. The number is absolute; you cannot specify a relative increase or decrease in words per minute.

- *For the expressive neural voices,* the `pitch` and `rate` attributes support only percentage values.
    - For the `pitch` attribute, do not use Hertz, semitones, or keywords.
    - For the `rate` attribute, do not use words per minute or keywords.

For more information about the features supported by SSML version 1.1, refer to section 3.2.4 prosody Element of the SSML specification.

The `pitch` attribute

The `pitch` attribute modifies the baseline pitch, or tone, for the text within the element. Accepted values are

- *A number followed by the `Hz` (Hertz) designation:* The baseline pitch is transposed (up or down) to the specified value. For example, `150Hz`.
- *A relative change in percent:* A number that causes a relative shift from the default baseline. The number is preceded by `+` (an increase) or `-` (a decrease) and followed by a `%` (percent sign). An unsigned number that is followed by a `%` is interpreted as a positive increase. For example, `+10%` or `10%`. The default pitch for a voice is equivalent to passing a value of `0%`.
- *A relative change in semitones:* A number that causes an absolute shift from the default baseline. The number is preceded by `+` (an increase) or `-` (a decrease) and followed by `st` (semitones). An unsigned number followed by `st` is interpreted as a positive increase. For example, `+5st` or `5st`.
- *A keyword:* One of the following six keywords, which modify the pitch to the corresponding predefined values:
    - `default` uses the service's default baseline pitch.
    - `x-low` shifts the pitch baseline down by 12 semitones.
    - `low` shifts the pitch baseline down by six semitones.
    - `medium` produces the same behavior as `default`.
    - `high` shifts the pitch baseline up by six semitones.
    - `x-high` shifts the pitch baseline up by 12 semitones.

Note: Expressive neural voices support only percentage values for the `pitch` attribute. They do not support the use of Hertz, semitones, or keywords.

The best way to determine what works for your application is to make adjustments based on percentages and experiment with different values. Try incremental changes of five or ten percent before making more significant modifications.

```
<prosody pitch="150Hz">Transpose pitch to 150 Hz</prosody>
<prosody pitch="-20Hz">Lower pitch by 20 Hz from baseline</prosody>
<prosody pitch="+20Hz">Increase pitch by 20 Hz from baseline</prosody>
<prosody pitch="-10%">Decrease pitch by 10 percent</prosody>
<prosody pitch="+10%">Increase pitch by 10 percent</prosody>
<prosody pitch="-12st">Lower pitch by 12 semitones from baseline</prosody>
<prosody pitch="+12st">Increase pitch by 12 semitones from baseline</prosody>
<prosody pitch="x-low">Lower pitch by 12 semitones from baseline</prosody>
```

The `rate` attribute

The `rate` attribute indicates a change in the speaking rate for the text within the element. Accepted values are

- *A number with no unit designation:* The rate is changed to the specified number of words per minute. For example, a value of `50` indicates a speaking rate of 50 words per minute. The number is absolute; you cannot specify a relative increase or decrease in words per minute.
- *A relative change in percent:* A number that causes a relative shift from the default speaking rate. The number is preceded by `+` (an increase) or `-` (a decrease) and followed by a `%` (percent sign). An unsigned number that is followed by a `%` is interpreted as a positive increase. For example, `+10%` or `10%`. The default speaking rate for a voice is equivalent to passing a value of `0%`.
- *A keyword:* One of the following six keywords, which modify the speaking rate to the corresponding predefined values:
    - `default` uses the service's default speaking rate.
    - `x-slow` decreases the rate by 50 percent.
    - `slow` decreases the rate by 25 percent.
    - `medium` produces the same behavior as `default`.
    - `fast` increases the rate by 25 percent.
    - `x-fast` increases the rate by 50 percent.

> **Note:** Expressive neural voices support only percentage values for the `rate` attribute. They do not support words per minute or keywords.

The best way to determine what works for your application is to make adjustments based on percentages and experiment with different values. Try incremental changes of five or ten percent before making more significant modifications.

```
<prosody rate="50">Set speaking rate to 50 words per minute</prosody>
<prosody rate="-5%">Decrease speaking rate by 5 percent</prosody>
<prosody rate="+5%">Increase speaking rate by 5 percent</prosody>
<prosody rate="slow">Decrease speaking rate by 25%</prosody>
<prosody rate="fast">Increase speaking rate by 25%</prosody>
```

The <say-as> element

The `<say-as>` element provides information about the type of text that is contained within the element and specifies the level of detail for rendering the text.

- The element has one required attribute, `interpret-as`, which indicates how the enclosed text is to be interpreted.
- The element has two optional attributes, `format` and `detail`, which are used only with particular values of the `interpret-as` attribute, as shown in the following examples.

The service supports the `<say-as>` element with the following languages:

- The service fully supports the `<say-as>` element for US English.
- For most other languages, the service supports only the `digits` and `letters` attributes of the element.
- For Japanese, the service supports only the `digits` attribute. The service ignores non-numeric characters that are included in the string of digits.

The service's default pronunciation of alphabetic, numeric, and alphanumeric strings varies by language, with each language having its own rules. You can use the `<say-as>` element to control how strings are pronounced, including whether they are to be spelled out as individual characters with the `letters` and `digits` elements.

For German, you can also control the pace at which the service pronounces the characters. For more information, see Specifying how strings are spelled out.

The interpret-as attribute

Acceptable values for the `interpret-as` attribute and examples of each value follow. The service supports the following values as arguments to the `interpret-as` attribute:

- cardinal
- date
- digits
- interjection
- letters
- number
- ordinal
- vxml:boolean
- vxml:currency
- vxml:date
- vxml:time
- vxml:digits
- vxml:phone

cardinal

The `cardinal` value speaks the cardinal number for the numeral within the element. The following examples say *Super Bowl forty-nine*. The first is superfluous, since it does not change the service's default behavior.

```
Super Bowl <say-as interpret-as="cardinal">49</say-as>
Super Bowl <say-as interpret-as="cardinal">XLIX</say-as>
```

date

The `date` value speaks the date within the element according to the format given in the associated `format` attribute. The `format` attribute is required for the `date` value. If no `format` is present, the service still attempts to pronounce the date. The following examples speak the indicated dates in the specified formats, where `d`, `m`, and `y` represent day, month, and year.

```
<say-as interpret-as="date" format="mdy">12/17/2005</say-as>
<say-as interpret-as="date" format="ymd">2005/12/17</say-as>
<say-as interpret-as="date" format="dmy">17/12/2005</say-as>
<say-as interpret-as="date" format="ydm">2005/17/12</say-as>
<say-as interpret-as="date" format="my">12/2005</say-as>
<say-as interpret-as="date" format="md">12/17</say-as>
<say-as interpret-as="date" format="ym">2005/12</say-as>
```

digits

The `digits` value speaks the digits in the number within the element. (The value also pronounces individually any alphabetic characters that are included in the enclosed string.) The following example speaks the individual digits *123456*.

```
<say-as interpret-as="digits">123456</say-as>
```

interjection

Note: The `interjection` attribute is an SSML extension specific to the Text to Speech service. It is supported for use only with the expressive neural voices.

With the expressive neural voices, the service automatically emphasizes the following interjections: `aha`, `hmm`, `huh`, `oh`, `uh`, `uh-huh`, and `um`. You can use the `interjection` value to enable or disable the service's emphasis of the interjections `aha` and `oh`. Include the additional `enabled` attribute with a value of `true` or `false` to enable or disable the interjection.

The following example disables emphasis of both the `aha` and `oh` interjections in the text:

```
<say-as interpret-as='interjection' enabled='false'>Oh</say-as>, in addition, the <say-as interpret-as='interjection' enabled='false'>aha</say-as> wasp is endemic to Australia.
```

For more information, see Emphasizing interjections.

letters

The `letters` value spells out the characters in the word within the element. (The value also pronounces individually any numeric characters that are included in the enclosed string.) The following example spells the letters of the word *hello*.

```
<say-as interpret-as="letters">Hello</say-as>
```

You can also specify the value `group` or `single` with the optional `format` attribute. These attributes help improve legibility of alphanumeric strings like confirmation of numbers and ID. The `single` format adds more silence while spelling-out characters one-by-one. The `group` format adds a longer silence when we switch from digits to letters and vice-versa, and after reading every 3 or 4 same type of characters.

```
<say-as interpret-as="letters" format=“single”>112A567B</say-as>
<say-as interpret-as="letters" format=“group”>3174A2W486</say-as>
```

number

The `number` value offers an alternative to the `cardinal` and `ordinal` values. You can use the optional `format` attribute to indicate how a series of numbers is to be interpreted. The first example omits the `format` attribute to pronounce the number as a cardinal value. The second example explicitly specifies that the number is to be pronounced as a `cardinal` value. The third example specifies that the number is to be pronounced as an `ordinal` value.

```
<say-as interpret-as="number">123456</say-as>
<say-as interpret-as="number" format="cardinal">123456</say-as>
<say-as interpret-as="number" format="ordinal">123456</say-as>
```

You can also specify the value `telephone` for the `format` attribute. The examples show two different ways of pronouncing a series of numbers as a telephone number. To pronounce the numbers with the punctuation included, specify the value `punctuation` for the optional `detail` attribute.

```
<say-as interpret-as="number" format="telephone">555-555-5555</say-as>
<say-as interpret-as="number" format="telephone" detail="punctuation">555-555-5555</say-as>
```

ordinal

The `ordinal` value speaks the ordinal value for the digit within the element. The following example says *second first*.

```
<say-as interpret-as="ordinal">2</say-as>
<say-as interpret-as="ordinal">1</say-as>
```

vxml:boolean

The `vxml:boolean` value speaks *yes* or *no* depending on the `true` or `false` value within the element.

```
<say-as interpret-as="vxml:boolean">true</say-as>
<say-as interpret-as="vxml:boolean">false</say-as>
```

vxml:currency

The `vxml:currency` value is used to control the synthesis of monetary values. The string must be written in the format `UUUmm.nn`, where `UUU` is the three-character currency indicator that is specified by ISO standard 4217 and `mm.nn` is the quantity. The following example says *forty-five dollars and thirty cents*.

```
<say-as interpret-as="vxml:currency">USD45.30</say-as>
```

If the specified number includes more than two decimal places, the amount is synthesized as a decimal number followed by the currency indicator. If the three-character currency indicator is not present, the amount is synthesized as a decimal number only and the currency type is not pronounced. The following example says *forty-five point three two nine US dollars* .

```
<say-as interpret-as="vxml:currency">USD45.329</say-as>
```

vxml:date

The `vxml:date` value works like the `date` value, but the format is predefined as `YYYYMMDD`. If a day, month, or year value is not known or if you do not want it to be spoken, replace the value with a `?` (question mark). The second and third examples include question marks.

```
<say-as interpret-as="vxml:date">20050720</say-as>
<say-as interpret-as="vxml:date">????0720</say-as>
<say-as interpret-as="vxml:date">200507??</say-as>
```

vxml:time

The 'vxml:time' value speaks the time within the element according to the format given in the associated format attribute. The format attribute is required for the time value. The format has to be four digits with either no suffix, "a", "p" or "h". The following examples speak the indicated time in the specified formats, where d, m, and y represent day, month, and year.

```
<say-as interpret-as="vxml:time">1230</say-as>
<say-as interpret-as="vxml:time">1230a</say-as>
<say-as interpret-as="vxml:time">1230p</say-as>
<say-as interpret-as="vxml:time">0100h</say-as>
```

vxml:digits

The `vxml:digits` value provides the same capabilities as the `digits` value.

vxml:phone

The `vxml:phone` value speaks a phone number with both digits and punctuation. It is equivalent to using the `number` value and specifying `telephone` for the `format` attribute and `punctuation` for the `detail` attribute.

```
<say-as interpret-as="vxml:phone">555-555-5555</say-as>
```

The <speak> element

> **Note:** The service supports SSML fragments, which are SSML elements that do not include the full XML header. The `<speak>` element is optional for SSML that you pass to the service.

The `<speak>` element is the root element for SSML documents. Valid attributes are

- `version` is a required attribute that specifies the SSML specification. The accepted value is `1.0`.
- `xml:lang` is not required by the service. Omit the attribute when you use this element. Note that you cannot use this attribute to change the language for a speech synthesis request.
- `xml:base` has no effect.
- `xmlns` is not required by the service. Omit the attribute when you use this element.

```
<speak version="1.1">
  The text to be spoken.
</speak>
```

The `<sub>` element

The `<sub>` element indicates that the text that is specified by the `alias` attribute is to replace the text that is enclosed within the element when speech is synthesized. The `alias` attribute is the only attribute of the element and is required.

```
<sub alias="International Business Machines">IBM</sub>
```

The `<voice>` element

This `<voice>` element requests a change in voice. It is not supported.

Using phonetic symbols

Understanding phonetic symbols

All languages and voices of the IBM Watson® Text to Speech service support both the standard International Phonetic Alphabet (IPA) and IBM Symbolic Phonetic Representation (SPR) notations to represent the sounds of words. Both notations provide phonetic encoding that represents the pronunciation of a word, the sounds that make up the word, how the sounds are divided into syllables, and which syllables are stressed. Phonetic symbols for supported languages provides links to topics that document the phonetic symbols for each language.

Defining a word pronunciation

To define the phonetic pronunciation for a word, either within input text or for a custom model, you use the `<phoneme>` element of the Speech Synthesis Markup Language (SSML) or equivalent method parameters. The `<phoneme>` element has two attributes:

- The `alphabet` attribute specifies the notation of the pronunciation. Use the value `ibm` to indicate that the pronunciation is defined in SPR. Use the value `ipa` to indicate that the pronunciation is defined in IPA.
- The `ph` attribute defines the pronunciation. It consists of a sequence of allowable symbols for a given language. The symbols define how the word that is enclosed in the `<phoneme>` element is to be pronounced.

Follow these rules when you define a pronunciation:

- Use only the documented SPR or IPA symbols. The service considers invalid any definition that contains phonetic symbols that are not allowed in a language. An SPR or IPA entry that does not conform to the required specification is invalid.
- When multiple IPA symbols (or symbol combinations) are documented for an SPR symbol, all of the IPA symbols are equivalent to the single SPR symbol. The service treats all of these IPA symbols the same and does not realize the subtle or regional differences that IPA is meant to describe.

For more information, see

- The phoneme element
- Rules for creating custom entries
- Creating and managing custom entries

Working with IBM SPR

IBM SPR is an alternative representation to standard IPA. The following examples of valid SPR notations define the words *through* and *shocking* in US English:

```
<phoneme alphabet="ibm" ph=".1Tru">through</phoneme>
<phoneme alphabet="ibm" ph=".1Sa.0kIG">shocking</phoneme>
```

In the definitions, the letters represent specific sounds of US English speech. A `.` signals the beginning of a new syllable, and the digits `1` and `0` indicate syllable stress. For more information, see Specifying syllables.

Speech sound symbols

Each language uses its own inventory of SPR symbols to represent the speech sounds of that language. The following rules apply to specifying an SPR symbol:

- Letters are case-sensitive, so `e` and `E`, for example, represent two different sounds.
- Two- and three-character symbols must be enclosed in single quotes when indicated in the symbol tables. The single quotes indicate that the multiple characters are actually a single symbol. For example, the symbol `'aj'` in the German word *heim* is specified as `"h'aj'm"`.
- Some three-character symbols include single quotes around only two of the characters. The single quotes indicate that the two characters are a single symbol. So the SPR consists of two symbols. For example, the symbol `'a:'n` in the Netherlands Dutch word **dependances** contains two symbols, `'a:'` and `n`, and is specified as `d'e:'.pEn.1d'a:'n.s@s`.

Also consider the following when defining a word's pronunciation in SPR format:

- The sounds of every language have specific distributional patterns within that language. For example, in all dialects of English, the sound `G` in *sing* (`".1sIG"`) does not occur at the beginning of a word. Other US English sounds that have a particularly narrow distribution are the glottal stop (`?`), the flap (`F`), and the syllabic nasal (`N`). If you enter a sound symbol in a context in which it does not normally occur, the resulting speech might sound unnatural.
- The service applies a sophisticated set of linguistic rules to its input to reflect the processes by which sounds change in specific contexts in natural language. For example, in US English, the sound `t` of the word *write* (`".1r1Yt"`) is pronounced as a flap (`F`) in *writer* (`".1rY.0FR"`). SPR input undergoes these modifications just as ordinary input text does. In this example, whether you enter `".1rY.0tR"` or `".1rY.0FR"` does not affect the

speech that is generated.

Working with IPA

You can define IPA pronunciations by using phonetic symbols or Unicode values. IPA is an industry standard notation. The following are examples of valid IPA notations for the word *tomato* in phonetic symbols and Unicode:

```
<phoneme alphabet="ipa" ph="təˈmeɪ.ɾoʊ">tomato</phoneme>
<phoneme alphabet="ipa" ph="t&#x0259;&#x02C8;me&#x026A;.&#x027E;o&#x028A;">tomato</phoneme>
```

Specifying syllables

You can specify syllable boundaries and stress in both SPR and IPA.

Syllable boundaries

You can use a `.` (period, IPA Unicode `002E`) to mark the beginning of each syllable in SPR or IPA. However, to preserve the valid phonetics of a language, the service can elect not to honor periods in some cases (for example, if a syllable boundary is placed at an illegal or unnatural position for a language). In general, in cases where you can indicate a valid preference for a syllable boundary or other aspect of a word's pronunciation, the service honors such requests.

Syllable stress

Table 1 identifies the symbols that you can use to indicate syllable stress for a pronunciation. IBM recommends that you indicate primary stress for pronunciations in either SPR or IPA. However, indicating syllable stress is optional for both formats; the service determines where stress occurs if you do not indicate it.

Stress	SPR symbol	IPA symbol	IPA Unicode
Primary stress	`1`	ˈ	`02C8`
Secondary stress	`2`	ˌ	`02CC`
No stress	`0`	No symbol	No value

Syllable stress

You must place a syllable stress marker within a syllable boundary but always to the left of the syllable's vowel. You can place a marker anywhere to the left of the stressed vowel. For example, each of the following SPR examples places the primary stress (`1`) on the correct vowel of the word *construction*:

```
<phoneme alphabet="ibm" ph="kXn1strHkSXn">construction</phoneme>
<phoneme alphabet="ibm" ph="kXns1trHkSXn">construction</phoneme>
<phoneme alphabet="ibm" ph="kXnst1rHkSXn">construction</phoneme>
<phoneme alphabet="ibm" ph="kXnstr1HkSXn">construction</phoneme>
```

Language-specific rules for using syllable stress

Table 2 lists language-specific considerations that apply to specifying syllable stress. Unless the table qualifies the rules for a language, you can use the syllable stress symbols described in the previous section.

Language	Notation	Language-specific rules
French and Canadian French	SPR	All syllable stress symbols are honored. But syllable stress must immediately precede the vowel of the syllable. Syllable stress for French is much stricter than for other languages. An error occurs if you place the stress symbol in an invalid location.
French and Canadian French	IPA	All syllable stress symbols are ignored.
Italian	SPR and IPA	You can specify only `1` (primary stress). An error occurs if you specify secondary or no stress.

Japanese	SPR and IPA	You can specify only `1` (primary stress) and `0` (no stress). An error occurs if you specify secondary stress.
Spanish	SPR and IPA	You can specify only `1` (primary stress). An error occurs if you specify secondary or no stress.

Language-specific rules for using syllable stress

Phonetic symbols for supported languages

Table 3 lists the languages that the service supports and provides links to topics that describe their SPR symbols, IPA symbols, and IPA Unicode values. The topics provide examples of each symbol in words from the language. Because of dialectal differences, the examples might not always match your pronunciation.

The *Availability* column indicates whether each voice is available for IBM Cloud, IBM Cloud Pak for Data, IBM Software Hub or all (*All versions*). For more information about the supported voices, see Languages and voices.

Language	Availability
Dutch (Netherlands) symbols	All versions
English (Australian) symbols	All versions
English (United Kingdom) symbols	All versions
English (United States) symbols	All versions
French (Canadian) symbols	All versions
French (France) symbols	All versions
German symbols	All versions
Italian symbols	All versions
Japanese symbols	All versions
Korean symbols	All versions
Portuguese (Brazilian) symbols	All versions
Spanish symbols	All versions

Phonetic symbols for supported languages

Dutch (Netherlands) symbols

IBM Cloud

The service supports the following symbols for Netherlands Dutch.

Regular vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
@	ə	0259	d**e**, uniek**e**

A	ɑ	0251	b**a**nk, p**a**d
E	ɛ	025B	h**e**ft, b**e**d
i	i	0069	v**ie**r, p**ie**t
I	ɪ	026A	k**i**p, r**i**ng
O	ɔ	0254	j**o**ng, r**o**b
u	u	0075	d**ou**che, k**oe**n
y	y	0079	**uu**r, d**uu**r
Y	**Y**	028F	dr**u**gs, dr**u**k, h**u**t

Regular vowels (Netherlands Dutch)

Long vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
'a:'	aː	0061+02D0	z**aa**d, v**aa**s
'e:'	eː	0065+02D0	p**ee**t, m**ee**rvoud
'E:'	ɛː	025B+02D0	fl**ai**r, **ai**rbag
'2:'	øː	00F8+02D0	n**eu**s, sh**i**rt, **eu**ro
'9:'	œː	0153+02D0	fr**eu**letjes
'o:'	oː	006F+02D0	n**o**dig, R**o**me
'Q:'	ɒː	0252+02D0	amaz**o**ne

Long vowels (Netherlands Dutch)

Nasalized vowels

SPR symbol [1]	IPA symbol	IPA Unicode	Example words
'a:'n	aːn	0061+02D0+006E	depend**an**ces
'e:'n	eːn	0065+02D0+006E	Chop**in**
'o:'n	oːn	006F+02D0+006E	**en**quête, g**en**re

Nasalized vowels (Netherlands Dutch)

Note:

1. In the SPR symbols, the single quotes that surround two-character symbols mean that the characters are interpreted as a single symbol. For example, the SPR `'a:'n` consists of two SPR symbols: `a:` and `n`. You must include the single quotes as shown when you use the symbols in an SPR pronunciation for a word.

Diphthongs

SPR symbol	IPA symbol	IPA Unicode	Example words

'aI'	ɛi	025B+0069	**aai**t, dr**aai**t
'Au'	ʌu	028C+0075	k**ou**d, f**au**n
'eI'	eɪ	0065+026A	l**ei**den, l**ei**dt
'eː'u	eːu	0065+02D0+0075	sn**eeuw**
'9y'	œy	0153+0079	m**ui**s
iu	iu	0069+0075	n**ieuw**
'oː'i	oː	006F+02D0	g**oo**it, f**oo**i, **oe**i
ui	ui	0075+0069	gr**oei**t
yu	yu	0079+0075	d**uw**en

Diphthongs (Netherlands Dutch)

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
b	b	0062	**b**ad
d	d	0064	**d**ak
'dZ'	dʒ **ʤ**	0064+0292 02A4	**j**azz
f	f	0066	**f**out
g	g	0067	**g**oal
G	ɣ	0263	re**g**en
h	ɦ	0266	**h**and, **h**oed
j	j	006A	les**j**e
k	k	006B	**k**at
l	l	006C	**l**and
m	m	006D	**m**at
n	n	006E	**n**ek
N	ŋ	014B	stori**n**g
p	p	0070	**p**ad
r	r	0072	**r**and
s	s	0073	**s**ap

S	ʃ	0283	pa**s**jes, **ch**ef
t	t	0074	**t**ak
v	v	0076	**v**ijf
w	ʋ	028B	d**w**alen, **w**ang
x	x	0078	**g**aan, to**ch**
z	z	007A	**z**at
Z	ʒ	0292	**j**ury

Consonants (Netherlands Dutch)

English (Australian) symbols

The service supports the following symbols for Australian (AU) English.

Regular vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
a	a	0061	p**a**th, f**a**ther, ch**a**nt
	aː	0061+02D0	
	ɑ	0251	
	ɑː	0251+02D0	
	ɐ	0250	
A	æ	00E6	b**a**ck, h**a**d
e	e	0065	c**a**ke, p**ai**n
	eɪ	0065+026A	
E	ɛ	025B	h**e**dge, l**e**t
i	iː	0069+02D0	s**ee**, sp**ea**k, bel**ie**ve
I	ɪ	026A	p**i**ck, **i**ll
o	o	006F	b**o**th, **oa**k
	oʊ	006F+028A	
	əʊ	0259+028A	
c	ɔ	0254	l**a**w, c**ou**rt, h**a**ll, w**a**ter
	ɔː	0254+02D0	
@	ɒ	0252	r**o**d, c**ou**gh
u	u	0075	z**oo**, tr**u**th
	uː	0075+02D0	

U	ʊ	028A	t**oo**k, p**u**t
H	ʌ	028C	b**u**t, m**u**g, s**o**n
R	ɜ	025C	butt**er**, h**u**rt
	ɜː	025C+02D0	
	ɚ	025A	
	ɝ	025D	

Regular vowels (AU English)

Diphthongs

SPR symbol	IPA symbol	IPA Unicode	Example words
O	ɔɪ	0254+026A	t**oi**l, b**oy**
W	aʊ	0061+028A	**ou**t, c**ow**
Y	aɪ	0061+026A	l**i**fe, f**i**ne
Ex	eɪ	0065+026A	sq**ua**re, b**a**red
	æɪ	00E6+026A	
Ix	ɪə	026A+0259	n**ea**r, h**e**re
Ux	ʊə	028A+0259	t**ou**r, l**u**red

Diphthongs (AU English)

Reduced vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
x	ə	0259	sof**a**, **a**lone, s**u**ppose, **A**meric**a**
X	i	0069	ros**e**s, hint**e**d
	ɨ	0268	

Reduced vowels (AU English)

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
b	b	0062	**b**ad, so**b**
C	tʃ	0074+0283	**ch**ip, wit**ch**, na**t**ure
	ʧ	02A7	
d	d	0064	**d**ip, ha**d**
D	ð	00F0	**th**is, brea**th**e
f	f	0066	**f**ield, i**f**, gra**ph**

g	g	0067	**g**ood, bu**g**
	ɡ	0261	
G	ŋ	014B	si**ng**, fi**ng**er
h	h	0068	**h**ot, **h**ero
	x	0078	
	χ	03C7	
	ɦ	0266	
J	dʒ	0064+0292	**J**ane, hu**g**e
	ʤ	02A4	
k	k	006B	**k**ill, ma**k**e, ba**ck**
l	l	006C	**l**ow, ha**ll**
L	l̩	006C+0329	candl**e**
m	m	006D	**m**an, hu**m**, su**mm**er
n	n	006E	**n**ever, su**n**, wi**nn**er
p	p	0070	**p**it, ri**p**
r	r	0072	bo**rr**ow, **r**ake
	ɹ	0279	
	ɾ	027E	
s	s	0073	**s**eal, mi**ss**, **c**eiling
S	ʃ	0283	**sh**ip, wi**sh**
t	t	0074	**t**ip, pe**t**
T	θ	03B8	**th**ing, Be**th**
v	v	0076	**v**ase, sa**v**e
w	w	0077	**w**ear, q**u**ick
y	j	006A	**y**es, Virgin**i**a
z	z	007A	**z**ip, pha**s**e
Z	ʒ	0292	trea**s**ure, gara**g**e

Consonants (AU English)

English (United Kingdom) symbols

The service supports the following symbols for United Kingdom (UK) English.

Regular vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
a	a	0061	p**a**th, f**a**ther, ch**a**nt
	aː	0061+02D0	
	ɑ	0251	
	ɑː	0251+02D0	
	ɐ	0250	
A	æ	00E6	b**a**ck, h**a**d
e	e	0065	c**a**ke, p**ai**n
	eɪ	0065+026A	
E	ɛ	025B	h**e**dge, l**e**t
i	iː	0069+02D0	s**ee**, sp**ea**k, bel**ie**ve
I	ɪ	026A	p**i**ck, **i**ll
o	o	006F	b**o**th, **oa**k
	oʊ	006F+028A	
	əʊ	0259+028A	
c	ɔ	0254	l**a**w, c**ou**rt, h**a**ll, w**a**ter
	ɔː	0254+02D0	
@	ɒ	0252	r**o**d, c**ou**gh
u	u	0075	z**oo**, tr**u**th
	uː	0075+02D0	
U	ʊ	028A	t**oo**k, p**u**t
H	ʌ	028C	b**u**t, m**u**g, s**o**n
R	ɜ	025C	butt**er**, h**u**rt
	ɜː	025C+02D0	
	ɚ	025A	
	ɝ	025D	

Regular vowels (UK English)

Diphthongs

SPR symbol	IPA symbol	IPA Unicode	Example words
O	ɔɪ	0254+026A	t**oi**l, b**oy**
W	aʊ	0061+028A	**ou**t, c**ow**
Y	aɪ	0061+026A	l**i**fe, f**i**ne

Diphthongs (UK English)

Reduced vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
x	ə	0259	sof**a**, **a**lone, s**u**ppose, **A**meric**a**
X	i	0069	ros**e**s, hint**e**d
	ɨ	0268	

Reduced vowels (UK English)

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
b	b	0062	**b**ad, so**b**
C	tʃ	0074+0283	**ch**ip, wit**ch**, na**t**ure
	ʧ	02A7	
d	d	0064	**d**ip, ha**d**
D	ð	00F0	**th**is, brea**th**e
f	f	0066	**f**ield, i**f**, gra**ph**
g	g	0067	**g**ood, bu**g**
	ɡ	0261	
G	ŋ	014B	si**ng**, fi**ng**er
h	h	0068	**h**ot, **h**ero
	x	0078	
	χ	03C7	
	ɦ	0266	
J	dʒ	0064+0292	**J**ane, hu**g**e
	ʤ	02A4	
k	k	006B	**k**ill, ma**k**e, ba**ck**
l	l	006C	**l**ow, ha**ll**
L	l̩	006C+0329	candl**e**
m	m	006D	**m**an, hu**m**, su**mm**er
n	n	006E	**n**ever, su**n**, wi**nn**er
p	p	0070	**p**it, ri**p**

r	r	0072	bo**rr**ow, **r**ake
	ɹ	0279	
	ɾ	027E	
s	s	0073	**s**eal, mi**ss**, **c**eiling
S	ʃ	0283	**sh**ip, wi**sh**
t	t	0074	**t**ip, pe**t**
T	θ	03B8	**th**ing, Be**th**
v	v	0076	**v**ase, sa**v**e
w	w	0077	**w**ear, q**u**ick
y	j	006A	**y**es, Virgin**i**a
z	z	007A	**z**ip, pha**s**e
Z	ʒ	0292	trea**s**ure, gara**g**e

Consonants (UK English)

English (United States & Canada) symbols

The service supports the following symbols for English (United States & Canada).

Regular vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
a	a	0061	f**a**ther, l**o**t
	aː	0061+02D0	
	ɑ	0251	
	ɑː	0251+02D0	
	ɐ	0250	
A	æ	00E6	b**a**ck, h**a**d
e	e	0065	c**a**ke, p**ai**n
	eɪ	0065+026A	
E	ɛ	025B	h**e**dge, l**e**t
i	i	0069	s**ee**, sp**ea**k, bel**ie**ve
	iː	0069+02D0	
I	ɪ	026A	p**i**ck, **i**ll
o	o	006F	b**o**th, **o**ak
	oʊ	006F+028A	

c	ɔ	0254	l**a**w, c**ou**gh
	ɔː	0254+02D0	
	ɒ	0252	
u	u	0075	z**oo**, tr**u**th
	uː	0075+02D0	
U	ʊ	028A	t**oo**k, p**u**t
	ɯ	026F	
H	ʌ	028C	b**u**t, m**u**g, s**o**n
R	ɚ	0259+02DE	butt**er**, h**ur**t
	ɝ	025A	
	ɜː	025C+02D0	
	ɜ˞	025D	

Regular vowels (US English)

Diphthongs

SPR symbol	IPA symbol	IPA Unicode	Example words
O	ɔɪ	0254+026A	t**oi**l, b**oy**
W	aʊ	0061+028A	**ou**t, c**ow**
Y	aɪ	0061+026A	l**i**fe, f**i**ne

Diphthongs (US English)

Reduced vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
x	ə	0259	sof**a**, **a**lone, s**u**ppose, tedi**ou**s, **A**meric**a**
X	ɨ	0268	ros**e**s, c**o**nnect, mel**o**dy, symph**o**ny, hint**e**d
	ɪ	0131	

Reduced vowels (US English)

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
b	b	0062	**b**ad, so**b**
C	tʃ	0074+0283	**ch**ip, wit**ch**, na**t**ure
	ʧ	02A7	
d	d	0064	**d**ip, ha**d**
D	ð	00F0	**th**is, brea**th**e

f	f	0066	**f**ield, i**f**, gra**ph**
F (flap)	ɾ	027E	wri**t**er, fi**dd**le, li**tt**le
g	g	0067	**g**ood, bu**g**
	ɡ	0261	
G	ŋ	014B	si**ng**, fi**ng**er
h	h	0068	**h**ot, **h**ero, **ch**allah
	ɦ	0266	
	x	0078	
	χ	03C7	
J	dʒ	0064+0292	**J**ane, hu**g**e
	ʤ	02A4	
k	k	006B	**k**ill, **c**at, ma**k**e, ba**ck**
l	l	006C	**l**ow, ha**ll**
m	m	006D	**m**an, hu**m**, su**mm**er
M	m̩	006D+0329	h**mmm**
n	n	006E	**n**ever, su**n**, wi**nn**er
N (syllabic nasal)	n̩	006E+0329	butt**on**, sat**in**, eat**en**, burd**en**
p	p	0070	**p**it, ri**p**
r	r	0072	bo**rr**ow, **r**ake
	ɹ	0279	
s	s	0073	**s**eal, mi**ss**, **c**eiling
S	ʃ	0283	**sh**ip, wi**sh**
t	t	0074	**t**ip, pe**t**
T	θ	03B8	**th**ing, Be**th**
v	v	0076	**v**ase, sa**v**e
w	w	0077	**w**ear, q**ui**ck
	ʍ	028D	
y	j	006A	**y**es, Virgin**i**a
z	z	007A	**z**ip, pha**s**e
Z	ʒ	0292	trea**s**ure, gara**g**e

? (glottal stop)	ʔ	0294	ki**tt**en, La**t**in

Consonants (US English)

French (Canadian) symbols

The service supports the following symbols for Canadian French.

Vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
a	a	0061	p**a**ttes, l**a**c, c**a**ve, inform**a**tion
A	ɑ	0251	ch**a**r, m**â**le
e	e	0065	caf**é**, d**e**former, **é**t**é**
E	ɛ	025B	**fait**e, dr**e**sser, p**è**re, annu**ai**re, f**ê**te
	ɜ	025C	
i	i	0069	s**i**, t**y**pique
I	ɪ	026A	s**i**te, plast**i**que, r**i**de
o	o	006F	p**au**le, b**eau**, t**ôt**, c**ô**té, v**au**devilliste
c	ɔ	0254	p**au**l, n**o**te, échal**o**tte, l**o**ge, enc**o**re
	ɒ	0252	
u	u	0075	r**ou**e, **où**, t**ou**r, f**ou**r, d**ou**ze
U	ʊ	028A	f**ou**le, m**ou**sse
y	y	0079	**u**tile, p**u**re, Br**u**no, d**u**r, b**u**se
Y	ʏ	028F	autob**u**s, ch**u**te
x	ə	0259	l**e**, qu**e**
'eu'	ø	00F8	j**eû**ne, ém**eu**te, m**eu**glement
	ɵ	0275	
'oe'	œ	0153	p**eu**r, j**eu**ne, déj**eu**ner, c**e**pendant, ch**e**val
	ɶ	0276	
'a~'	ã	0061+0303	b**an**c, **en**, t**em**ps
	ɑ̃	0251+0303	
'E~'	ẽ	0065+0303	f**in**, pl**ein**, f**aim**
	ɛ̃	025B+0303	
'o~'	õ	006F+0303	b**on**, p**on**t, m**on**
	ɔ̃	0254+0303	

'oe~'	œ̃	0153+0303	**un**, auc**un**

Vowels (Canadian French)

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
b	b	0062	**b**é**b**é, **b**alle, ro**b**e
d	d	0064	**d**ort, **d**olmen
D	dz	0064+007A	**d**uque, **d**ire
	ʣ	02A3	
f	f	0066	che**f**, **f**aim, **ph**are
g	g	0067	**gu**erre, ba**gu**e, **g**arer
	ɡ	0261	
H	ɥ	0265	s**u**is, l**u**i
j	j	006A	h**i**érarchie, pa**ill**e, **y**oga
	ʎ	028E	
k	k	006B	**k**ilo, **c**aler, **qu**ai
l	l	006C	**l**itre, i**ll**isible, pâ**l**e
m	m	006D	**m**a**m**an, fe**mm**e, **m**iser
n	n	006E	A**nn**e, **n**i, ma**n**iaque
'ng'	ŋ	014B	parki**ng**, campi**ng**
'nj'	ɲ	0272	a**gn**eau, campa**gn**e
p	p	0070	**p**orte, **p**rêt, guê**p**e
r	r	0072	pa**r**er, **r**a**r**e, ca**rr**eau
	ɹ	0279	
	ʀ	0280	
	ʁ	0281	
s	s	0073	**s**ans, ambi**t**ion, fa**ç**on
S	ʃ	0283	**ch**eval, lâ**ch**e, **sch**éma
t	t	0074	**t**on, pa**tt**e, théâ**t**re
T	ʦ	0074+0073	pe**t**it, **t**uque
	ʦ	02A6	
v	v	0076	la**v**er, **w**agon, **v**isiter

w	w	0077	**ou**i, **w**att
z	z	007A	ja**s**er, ré**s**eau, **z**ig**z**aguer
Z	ʒ	0292	ra**g**e, **g**îte, **j**ouer

Consonants (Canadian French)

Liaison

In Canadian French, the `_` (underscore) can be used immediately following the final consonant of a word (but within the double-quotes that enclose the SPR) to indicate that it is a liaison consonant. A liaison consonant is pronounced only if the word that follows begins with a vowel.

SPR symbol	IPA symbol	IPA Unicode
_	‿	203F

Liaison (Canadian French)

Examples of words with and without the liaison symbol follow:

- `"p0'oe't1it_"`: The `t` is pronounced only if the following word begins with a vowel.
- `"nEt"`: The `t` is always pronounced.

A dictionary entry *petit* with the translation value `"p'oe't1it_"` has the final `t` pronounced in the input string *un petit ami* but not in the input string *un petit chien*. An entry with the translation value `"nEt"` has the final `t` pronounced regardless of context.

French (France) symbols

The service supports the following symbols for French.

Vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
a	a	0061	p**a**ttes, l**a**c, c**a**ve
	ɑ	0251	
	ɐ	0250	
e	e	0065	caf**é**, d**é**form**e**r, **été**
E	ɛ	025B	f**ai**te, m**ai**, h**e**rb
	ɜ	025C	
i	i	0069	f**i**lm, t**y**p**i**que
	ɪ	026A	
o	o	006F	**eau**, **au**x, g**au**che
c	ɔ	0254	P**au**l, n**o**te, échal**o**tte
	ɒ	0252	
u	u	0075	r**ou**e, **où**, t**ou**r
	ʊ	028A	

y	y	0079	**u**tile, p**u**re, Br**u**no
	Y	028F	
x **1**	ə	0259	litr**e**s, marbr**e**
'eu'	ø	00F8	m**eu**gle, p**eu**, joy**eu**x
	ɵ	0275	
'oe'	œ	0153	p**eu**r, c**oeu**r, j**eu**ne
	Œ	0276	
'a~'	ã	0061+0303	b**an**c, **en**, t**em**ps
	ɑ̃	0251+0303	
'E~'	ẽ	0065+0303	f**i**n, pl**e**in, f**ai**m
	ɛ̃	025B+0303	
	æ̃	00E6+0303	
'o~'	õ	006F+0303	b**on**, p**on**t, m**on**
	ɔ̃	0254+0303	
'oe~'	œ̃	0153+0303	**un**, auc**un**, br**un**

Vowels (French)

Note:

1. The `x` is elided in certain contexts.

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
b	b	0062	**b**é**b**é, **b**alle, ro**b**e
d	d	0064	**d**ort, **d**olmen
f	f	0066	che**f**, **f**aim, **ph**are
g	g	0067	**gu**erre, ba**gu**e, **g**arer
	ɡ	0261	
H	ɥ	0265	s**u**is, l**u**i, n**u**ée
j	j	006A	h**i**érarchie, pa**ill**e, **y**oga
	ʎ	028E	
k	k	006B	**k**ilo, **c**aler, **qu**ai
l	l	006C	**l**itre, i**ll**isible, pâ**l**e
m	m	006D	**m**a**m**an, fe**mm**e, **m**iser

n	n	006E	A**nn**e, **n**i, ma**n**iaque
'ng'	ŋ	014B	parki**ng**, campi**ng**
'nj'	ɲ	0272	a**gn**eau, campa**gn**e
p	p	0070	**p**orte, **p**rêt, guê**p**e
r	r	0072	pa**r**er, **r**a**r**e, ca**rr**eau
	ɹ	0279	
	ʀ	0280	
	ʁ	0281	
	x	0078	
	χ	03C7	
s	s	0073	**s**ans, ambi**t**ion, fa**ç**on
S	ʃ	0283	**ch**eval, lâ**ch**e, **sch**éma
t	t	0074	**t**on, pa**tt**e, théâ**t**re
v	v	0076	la**v**er, **w**agon, **v**isiter
w	w	0077	**ou**i, b**ou**ée, **w**att
z	z	007A	ja**s**er, ré**s**eau, **z**ig**z**aguer
Z	ʒ	0292	ra**g**e, **g**îte, **j**ouer

Consonants (French)

Liaison

In French, the `_` (underscore) can be used immediately following the final consonant of a word (but within the double-quotes that enclose the SPR) to indicate that it is a liaison consonant. A liaison consonant is pronounced only if the word that follows begins with a vowel.

SPR symbol	IPA symbol	IPA Unicode
_	‿	203F

Liaison (French)

Examples of words with and without the liaison symbol follow:

- `"p0'oe't1it_"`: The `t` is pronounced only if the following word begins with a vowel.
- `"nEt"`: The `t` is always pronounced.

A dictionary entry *petit* with the translation value `"p'oe't1it_"` has the final `t` pronounced in the input string *un petit ami* but not in the input string *un petit chien*. An entry with the translation value `"nEt"` has the final `t` pronounced regardless of context.

German symbols

The service supports the following symbols for German.

Regular vowels

SPR symbol	IPA symbol	IPA Unicode	Example words

i	i	0069	l**ie**ben, T**i**tel, t**ie**f
	iː	0069+02D0	
I	ɪ	026A	b**i**tte, T**i**sch, L**i**cht
e	e	0065	g**e**ben, **Eh**re, S**ee**
	eː	0065+02D0	
E	ɛ	025B	tr**e**ffen, G**e**ld, k**ä**mmen
'E:'	ɛː	025B+02D0	K**ä**se, M**ä**dchen, w**ä**gen
a	aː	0061+02D0	H**aa**r, h**a**ben, f**a**hren
	ɑ	0251	
A	a	0061	l**a**ssen, m**a**tt, **A**pfel
u	u	0075	g**u**t, **Uh**r, **U**we
	uː	0075+02D0	
U	ʊ	028A	H**u**nd, Fl**u**ss, M**u**tter
o	o	006F	**O**ber, **oh**ne, B**oo**t
	oː	006F+02D0	
O	ɔ	0254	K**o**pf, St**o**pp
y	y	0079	B**ü**cher, f**üh**len, T**ü**r, k**ü**hn
	yː	0079+02D0	
Y	Y	028F	f**ü**nf, f**ü**llen, K**ü**nstler
'oe'	ø	00F8	L**ö**we, h**ö**ren, S**öh**ne
	øː	00F8+02D0	
'OE'	œ	0153	k**ö**nnen, h**ö**lzern, **ö**stlich
	Œ	0276	

Regular vowels (German)

Diphthongs

SPR symbol	IPA symbol	IPA Unicode	Example words
'aj'	ai	0061+0069	h**ei**m, W**ai**se, M**ai**
	aɪ	0061+026A	
'aw'	au	0061+0075	H**au**s, M**au**l, Fr**au**
	aʊ	0061+028A	

'oj'	ɔi	0254+0069	h**eu**te, Geb**äu**de, H**äu**ser
	ɔy	0254+0079	
	ɔɪ	0254+026A	
	ɔ**Y**	0254+028F	

Diphthongs (German)

Nasalized vowels

Nasalized vowels in German occur mostly in foreign loan words.

SPR symbol	IPA symbol	IPA Unicode	Example words
'a~'	ã	0061+0303	Ch**an**ce
	ɐ̃	0250+0303	
'E~'	ẽ	0065+0303	T**ein**t
	ɛ̃	025B+0303	
'o~'	õ	006F+0303	Pard**on**
	ɔ̃	0254+0303	
'oe~'	ø̃	00F8+0303	Parf**um**
	œ̃	0153+0303	

Nasalized vowels (German)

Reduced vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
@	ə	0259	bitt**e**, Kam**e**ra, Bod**e**n
	ɵ	0275	
	ɘ	0258	

Reduced vowels (German)

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
b	b	0062	**B**oden, **B**ett, o**b**en
C	tʃ	0074+0283	deuts**ch**, **Ch**ile, **C**ello
	t͡ʃ	02A7	
d	d	0064	**d**unkel, kin**d**isch, Hel**d**en
	ð [1]	00F0	
f	f	0066	**f**ast, ho**ff**en, **V**ater

g	g	0067	**g**eben, **g**rau, Ta**g**e
	g	0261	
	ɣ	0263	
G	ŋ	014B	Fi**ng**er, lä**ng**s, Anfa**ng**
h	h	0068	**h**och, **H**and, A**h**orn
	ɦ	0266	
j	j	006A	**J**unge, **j**a, **J**ahr, Ministeri**i**um
	ɟ	029D	
J	dʒ	0064+0292	**J**ob, **Dsch**ungel
	ʤ	02A4	
k	k	006B	**K**atze, E**ck**e, S**k**ulptur, la**g**, **q**uitt
l	l	006C	**l**esen, fa**ll**en, Pu**l**t
m	m	006D	**M**ann, ko**mm**en, Ate**m**
n	n	006E	**N**acht, kö**nn**en, Ki**n**d
p	p	0070	**P**apier, Li**pp**e, Gra**b**
P	pf	0070+0066	**Pf**lanze, Stum**pf**en
r	r	0072	**R**ad, füh**r**en
	ɹ	0279	
	ʀ	0280	
	ʁ	0281	
	ɾ	027E	
R	ɐ	0250	Wied**er**, üb**er**
	ʌ	028C	
	ɚ	025A	
	ɝ	025D	
s	s	0073	Ma**ß**, la**ss**en, La**s**t, H**au**s
S	ʃ	0283	**sch**on, **s**pielen, **S**til, wä**sch**t
t	t	0074	**T**ag, bi**tt**e, Ra**d**
T	ts	0074+0073	**Z**auber, Poli**z**ei, Glan**z**
	ʦ	02A6	
	θ [2]	03B8	

v	v	0076	**W**agen, **v**iskös, **V**olum, o**v**al
w	w	0077	Ed**u**ard, akt**u**ell, Jan**u**ar
	ʊ	028B	
x	x	0078	Bu**ch**, Ba**ch**, Wo**ch**en
	χ	03C7	
X	ç	00E7	i**ch**, **Ch**emie, Kel**ch**, man**ch**er
z	z	007A	**S**ee, **S**atz, le**s**en
Z	ʒ	0292	Gara**g**e, **G**enie
? (glottal stop)	**ʔ**	0294	**e**r**o**bern (IPA ʔɛɐ'ʔoːbɐn) **e**rinnern (IPA ʔɛɐ'ʁɪnɐn)

Consonants (German)

Notes:

1. The IPA symbol `ð` (IPA Unicode `00F0`) occurs only in loanwords from foreign languages, mostly English (for example, *The New York Times*). However, the service realizes that sound as IPA symbol `d` (IPA Unicode `0064`).
2. The IPA symbol `θ` (IPA Unicode `03B8`) occurs only in loanwords from foreign languages, mostly English (for example, *thriller*). However, the service realizes that sound as IPA symbol `ʦ` (IPA Unicode `02A6`).

Italian symbols

The service supports the following symbols for Italian.

Regular vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
a	a	0061	l**a**s**a**gn**a**, **a**llegro
e	e	0065	n**e**ro, du**e**tto
E	ɛ	025B	**e**cco, lic**e**o
i	i	0069	**i**sola, form**i**ca
o	o	006F	padr**o**ne, att**o**re
c	ɔ	0254	c**o**sta, m**o**sse
u	u	0075	l**u**na, **u**fficio

Regular vowels (Italian)

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
b	b	0062	**b**occa, **b**ere
C	tʃ	0074+0283	**c**e**c**e, **c**iao
	ʧ	02A7	

d	d	0064	**d**are, **d**ata
D	dz	0064+007A	**z**abaione, **z**ero, **z**ona
	ʣ	02A3	
f	f	0066	**f**are, **f**orte
g	g	0067 0261	**g**rande, re**g**alo
J	dʒ	0064+0292	**Gi**ovanni, con**g**elare
	ʤ	02A4	
k	k	006B	**c**asa, ve**cch**io
l	l	006C	**l**ento, pa**l**ma
L	ʎ	028E	**gl**ielo, **gl**i
m	m	006D	**m**a**mm**a, **m**ano
n	n	006E	**n**ie**n**te, **n**otte
N	ɲ	0272	**gn**occhi, lasa**gn**a
p	p	0070	**p**artire, **p**oco
r	r	0072	ca**r**o, se**r**eno
	ɾ	027E	
R	rr	0072+0072	te**rr**a, to**rr**e
	rː	0072+02D0	
s	s	0073	pe**s**to, **s**tare
S	ʃ	0283	**sc**egliere, la**sc**iare
t	t	0074	**t**occare, len**t**o
T	ts	0074+0073	**z**ampa, **z**uppa
	ʦ	02A6	
v	v	0076	**v**ano, **v**i**v**ere
w	w	0077	n**u**ovo, q**u**ando
y	j	006A	**i**eri, raso**i**o
z	z	007A	pae**s**e, **s**baglio

Consonants (Italian)

Geminates

SPR symbol	IPA symbol	IPA Unicode	Example words

CC	tʧ ttʃ	0074+02A7 0074+0074+0283	ghia**cc**io, fe**cc**ia
DD	ddʒ ddz	0064+02A3 0064+0064+007A	a**zz**urro, me**zz**o
JJ	ddʒ **dʤ**	0064+0064+0292 0064+02A4	Chio**gg**ia, ma**gg**io
TT	tts tʦ	0074+0074+0073 0074+02A6	ta**zz**a, chia**zz**a

Geminates (Italian)

Allophones

If an allophone is not used in SPR, the Text to Speech service automatically generates the appropriate variant for the given context.

SPR symbol	IPA symbol	IPA Unicode	Example words
'ng' **1**	ŋg	014B+0067 014B+0261	a**ng**olo, lu**ng**o
'nk' **1**	ŋk	014B+006B	a**nc**ora, bia**nc**o

Allophones (Italian)

Note:

1. The `ng` and `nk` allophones are supported in the IPA specification but cannot be used in SPR.

Japanese symbols

The service supports the following symbols for Japanese. The significant symbols in example Japanese words are shown in **bold**.

Vowels and marks

SPR symbol	IPA symbol	IPA Unicode	Example words
a	a ɐ ɑ	0061 0250 0251	相手(アイテ)
i	i ɪ	0069 026A	いたずら(イタズラ)
u	ɯ u	026F 0075	宇宙(ウチュウ)
e	e ɛ	0065 025B	絵本(エホン)

o	o	006F	お菓子(オカシ)
	ɒ	0252	
	ɔ	0254	
: **1**	ː	02D0	レース
^	↓	A71C	コグ'ニティブ
	^	005E	
'_n'	ɴ	0274	満月(マンゲツ)
	ŋ	014B	

Vowels and marks (Japanese)

Note:

1. The `:` symbol must be followed by a vowel, possibly with other symbols, such as syllable boundary, in between.

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
b	b	0062	バナナ
	v	0076	
'by'	bj	0062+006A	白夜(ビャクヤ)
'ch'	ʧ	02A7	地図(チズ)
	t͡ɕ	02A8	
d	d	0064	団子(ダンゴ)
'dy'	dj	0064+006A	デュエット
f	ɸ	0278	不夜城(フヤジョウ)
	f	066	
g	g	0067	群青(グンジョウ)
	ɣ	0263	
'gy'	gj	0067+006A	牛丼(ギュウドン)
h	h	0068	花柄(ハナガラ)
'hy'	ç	00E7	百年(ヒャクネン)
j	ʤ	02A4	ジュゴン(ジュゴン)
	d͡ʑ	02A5	
k	k	006B	公園(コウエン)
'ky'	kj	02B7+006A	休日(キュウジツ)

m	m	006D	魔法(マホウ)
'my'	mj	006D+006A	陰陽師(オンミョウジ)
n	n	006E	二輪車(ニリンシャ)
'ny'	ɲ	0272	牛乳(ギュウニュウ)
p	p	0070	パンダ
'py'	pj	0070+006A	ぴょこん(ピョコン)
'Qb'	bb	0062+0062	ばっばーん(バッバーン)
'Qch'	ʧʧ	02A7+02A7	おっちゃん(オッチャン)
'Qd'	dd	0064+0064	ベッド
'Qf'	ɸɸ	0278+0278	フッフール
'Qg'	gg	0067+0067	タッグマッチ
'Qh'	hh	0068+0068	コッヘル(コッヘル)
'Qj'	ʤʤ	02A4+02A4	じゃっじゃめ(ジャッジャメ)
'Qk'	kk	02B7+02B7	月光(ゲッコウ)
'Qky'	kkj	02B7+02B7+006A	特許(トッキョ)
'Qp'	pp	0070+0070	ハッピー
'Qpy'	ppj	0070+0070+006A	突拍子(トッピョウシ)
'Qs'	ss	0073+0073	疾走(シッソウ)
'Qsh'	ɕɕ	0255+0255	八尺(ハッシャク)
'Qt'	tt	0074+0074	雪駄(セッタ)
'Qts'	ʦʦ	02A6+02A6	鉄槌(テッツイ)
'Qz'	ʣʣ	02A3+02A3	キッズ
r	ɾ	027E	ライ麦(ライムギ)
	r	0072	
	ɹ	0279	
	l	006C	
	ɽ	027D	
'ry'	ɾj	027E+006A	流星群(リュウセイグン)
s	s	0073	寿司(スシ)

'sh'	ɕ	0255	四季(シキ)
	ʃ	0283	
t	t	0074	大河(タイガ)
'ts'	ʦ	02A6	釣り(ツリ)
w	w	0077	悪だくみ(ワルダクミ)
	ɯ̟	0270	
y	j	006A	陽気(ヨウキ)
z	ʣ	02A3	全部(ゼンブ)
	z	007A	

Consonants (Japanese)

Korean symbols

IBM Cloud

The service supports the following symbols for Korean.

Monophthongs

SPR symbol	IPA symbol	IPA Unicode	Example words
a	a	0061	한 /h**a**n/
	ɐ	0250	
	ɑ	0251	
	ɞ	025E	
	ɐ	029A	
A	ɛ	025B	새로 /sɛɾo/
	æ	00E6	
	ɜ	025C	
E	ʌ	028C	것 /kʌt/
	ə	0259	
	ɘ	0258	
	ɵ	0275	
e	e	0065	세 /s**e**/
i	i	0069	시 /s**i**/
	ɪ	0131	
	ɨ̞	0268	
	ɩ	0269	
	ɪ	026A	
o	o	006F	사고 /sag**o**/
	ɒ	0252	
	ɔ	0254	
	ɤ	0264	
u	u	0075	수 /s**u**/
	ɯ	0277	
	ʉ	0289	
	ʊ	028A	

U	ɯ	026F	그 /kɯ/

Monophthongs (Korean)

Diphthongs

SPR symbol	IPA symbol	IPA Unicode	Example words
ya	ja	U+006A + U+0061	약 /**ja**k/
yA	jɛ	U+006A + U+025B	얘기 /**jɛ**gi/
yE	jʌ	U+006A + U+028C	여섯 /**jʌ**sʌt/
ye	je	U+006A + U+0065	예금 /**je**gɯm/
yo	jo	U+006A + U+006F	요즘 /**jo**dʑɯm/
yu	ju	U+006A + U+0075	유독 /**ju**dok/
wa	wa	0077+0061	봐 /p**wa**/
wA	wɛ	0077+025B	왜 /**wɛ**/
wE	wʌ	0077+028C	원인 /**wʌ**nin/
we	we	0077+0065	외부 /**we**bu/
wi	wi ɥi y	0077+0069 0265+0069 0079	행위 /hɛŋ**wi**/
I	ɯi ɯi	0270+0069 026F+0069	의사 /ɯ**i**sa/

Diphthongs (Korean)

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
c	dʑ d͡ʑ tɕ t͡ɕ	0064+0291 02A5 0074+0255 02A8	여자 /jʌ**dʑ**a/
'cc'	t̬ɕ tɕ̬ t͡ɕ= t͡ɕ̬	0074+0348+0255 0074+0255+0348 02A8+02ED 02A8+0348	약점 /jakt̬ɕʌm/
'ch'	tɕʰ t͡ɕʰ	0074+0255+02B0 02A8+02B0	부착 /pu**tɕʰ**ak/
h	h ɦ	0068 0266	한 /**h**an/
k	g g̬ ɡ̬	006B 0067+0325 0261+0325	육 /ju**k**/

'kh'	kʰ	006B+02B0	약한 /ja**k**ʰan/
'kk'	k͈ k̟̈	006B+02ED 006B+0348	꼭 /**k̟̈**ok/
l	l ɭ	006C 026D	오늘 /onɯ**l**/
m	m	006D	마치 /**m**atɕʰi/
n	n	006E	안 /a**n**/
'ng'	ŋ	014B	사상 /sasa**ŋ**/
p	b b̥ p v	0062 0062+0325 0070 0076	보고 /**p**ogo/
'ph'	f pʰ	0066 0070+02B0	판매 /**p**ʰanmɛ/
'pp'	p͈ p̟̈	0070+02ED 0070+0348	뿐 /**p̟̈**un/
r	r ɹ ɾ ʀ ʁ	0072 0279 027E 0280 0281	주로 /tɕuɾo/
s	sʰ s z **ɕ** **ɕ**ʰ	0073+02B0 0073 007A 0255 0255+02B0	삼 /**s**am/
'ss'	s͈ s̟̈ **ɕ͈** **ɕ̟̈**	0073+02ED 0073+0348 0255+02ED 0255+0348	쓰고 /**s̟̈**ɯgo/
t	d̥ d t	0064+0325 0064 0074	도 /**t**o/
'th'	tʰ	0074+02B0	같은 /ka**t**ʰɯn/
'tt'	t͈ ẗ̟	0074+02ED 0074+0348	때 /**ẗ̟**ɛ/
w	w	0077	왜 /**w**ɛ/
y	j	006A	야 /**j**a/

Consonants (Korean)

Portuguese (Brazilian) symbols

The service supports the following symbols for Brazilian Portuguese.

Regular vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
a **1**	a	0061	vir**ar**, m**a**r
	ɐ	0250	Terr**a**, muit**a**
e	e	0065	d**e**do, portugu**ê**s
E	ɛ	025B	**é**s, b**e**lo, **e**co
i	i	0069	l**i**gar, bod**e**
o	o	006F	c**o**r, b**o**lha
c	ɔ	0254	pr**ó**ximo, p**o**rta
u	u	0075	l**u**gar, per**u**
'a~'	ɐ̃	0250+0303	b**a**nco, l**ã**
'e~'	ẽ	0065+0303 1EBD	inc**en**so, ag**en**te
'i~'	ĩ	0069+0303 0129	ass**im**, **in**censo
'o~'	õ	006F+0303 00F5	c**ôn**sul, t**om**
'u~'	ũ	0075+0303 0169	alg**un**s, **um**

Regular vowels (Brazilian Portuguese)

Note:

1. The SPR symbol `a` maps to two different IPA symbols with different pronunciations: `a` and `ɐ`. In general, the SPR symbol maps to `a` for synthesis of stressed syllables and to `ɐ` for synthesis of unstressed syllables.

Semi-vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
y	j	006A	do**i**s, no**i**va
Y	j̃	006A+0303	mã**e**, mu**i**to
w	w	0077	me**u**, q**u**arto
W	w̃	0077+0303	capitã**o**, Sã**o**

Semi-vowels (Brazilian Portuguese)

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
b	b	0062	a**b**raço, **B**rasil

C	tʃ ʧ	0074+0283 02A7	se**t**e, bo**t**e
d	d	0064	**d**ar, **d**ente
f	f	0066	**f**lauta, **f**aixa
g	g	0067 0261	**g**ato, **g**uarda
J	dʒ ʤ	0064+0292 02A4	bo**d**e, **d**iz
k	k	006B	**c**ama, **qu**eda
l	l	006C	**l**eite, cava**l**o
'ly'	ʎ	028E	**lh**e, bagu**lh**o
m	m	006D	for**m**a, **m**acaco
n	n	006E	do**n**o, **n**ovo
N	ɲ	0272	cu**nh**a, ni**nh**o
p	p	0070	**p**luma, **p**rimo
r	ɾ	027E	ca**r**o, t**r**em
R	ʁ	0281	ca**rr**o, **r**io
s	s	0073	**c**erto, avan**ç**ar, **s**ete
S	ʃ	0283	**ch**ave, bai**x**a
t	t	0074	**t**rono, por**t**a
v	v	0076	**v**ila, bre**v**e
z	z	007A	**z**ero, ca**s**o
Z	ʒ	0292	**g**eral, **j**ogo

Consonants (Brazilian Portuguese)

Spanish symbols

The service supports the following symbols for Spanish. The information applies to the Castilian, Latin American, and North American dialects. Except where noted, the dialects are identical.

Regular vowels

SPR symbol	IPA symbol	IPA Unicode	Example words
a	a	0061	**a**gua, c**a**s**a**

e	e	0065	**e**st**e**, b**e**so
i	i	0069	**i**gual, h**i**jo
o	o	006F	**o**s**o**, cant**o**
u	u	0075	**u**va, l**u**gar

Regular vowels (Spanish)

Consonants

SPR symbol	IPA symbol	IPA Unicode	Example words
b	b	0062	**b**asta, **b**eber, **v**aca
C	ʧ	02A7	**ch**alupa, mu**ch**o
	tʃ	0074+0283	
d	d	0064	**d**ar, **d**edo
f	f	0066	**f**laco, a**f**uera
g	g	0067 0261	**g**oma, **g**orra
j	h **1**	0068	**j**unco, re**j**a, **g**ente
	x **2**	0078	
k	k	006B	**c**uen**c**o, **c**anto
l	l	006C	**l**oco, a**l**go
L **3**	ʎ	028E	**ll**over, po**ll**o
	ʝ **4**	029D	**y**egua, pla**y**a
m	m	006D	**m**ano, a**m**or
n	n	006E	**n**ada, ma**n**o
N	ɲ	0272	pi**ñ**a, ni**ñ**o
p	p	0070	**p**arte, a**p**agar
r	ɾ	027E	pa**r**a, pe**r**o
R	r	0072	**r**opa, pe**rr**o
s	s	0073	**s**illa, ca**s**a
t	t	0074	**t**oma, a**t**ar
T **5**	θ	03B8	**z**apato, so**c**io
w	w	0077	f**u**era, de**u**da

y	j	006A	medio, oigo
Y	ɟ 6	029D	yegua, playa

Consonants (Spanish)

Notes:

1. The IPA symbol `h` applies to North American and Latin American Spanish only.
2. The IPA symbol `x` applies to Castilian Spanish only.
3. The SPR symbol `L` maps to two different IPA symbols with different pronunciations for both North American and Latin American Spanish: `ʎ` and `ɟ`. Specifying this SPR symbol with these dialects might yield either of the two variants. The difference between the pronunciations is often indistinguishable to native speakers.
4. The IPA symbol `ɟ` maps to the SPR symbol `L` in North American and Latin American Spanish.
5. The SPR symbol `T` is realized only in Castilian Spanish. It is replaced internally with the phonetic symbol `s` in North American and Latin American Spanish, even when present in the SPR input.
6. The IPA symbol `ɟ` maps to the SPR symbol `Y` in Castilian Spanish.

Allophones

Each allophone is a variation of the phoneme indicated in parentheses. If an allophone is not used in SPR, the Text to Speech service automatically generates the appropriate variant for the given context.

SPR symbol	IPA symbol	IPA Unicode	Example words
B (b)	β	03B2	bobo, nueve
D (d)	ð	00F0	hada, dedo
G (g)	ɣ	0263	segar, lugar
z (s)	z	007A	mismo, desde
'ng' **1**	ŋg	014B+0067	tango, engaño
'nk' **1**	ŋk	014B+006B	ancla, cinco

Allophones (Spanish)

Note:

1. The `ng` and `nk` allophones are supported in the IPA specification but cannot be used in SPR.

Obsolete phonetic symbols

Dutch (Netherlands) symbols (obsolete)

The service supported the following symbols for the obsolete neural voices for Netherlands Dutch. These voices were replaced with an alternative enhanced neural voice and removed from the service on 31 March 2023. The symbols are available to help users migrate from the obsolete Netherlands Dutch neural voices to the new Netherlands Dutch enhanced neural voice.

Regular vowels

IPA symbol	IPA Unicode	Example words
ɑ	0251	bad
ɛ	025B	bed
ɪ	026A	vis

ɔ	0254	p**o**t
Y	028F	h**u**t
u	0075	h**oe**d
ø	00F8	b**eu**k
ə	0259	d**e**
ʌ	028C	gl**o**ve

Regular vowels (Netherlands Dutch)

Long vowels

IPA symbol	IPA Unicode	Example words
a	0061	n**aa**m
e	0065	h**ee**t
i	0069	d**ie**p
ɪː	026A+02D0	k**ee**l
o	006F	p**oo**t
y	0079	**uu**r
iː	0069+02D0	ch**ea**p
ɛː	0258+02D0	g**ê**ne
ɔː	0254+02D0	r**o**ze
uː	0075+02D0	bl**ue**
yː	0079+02D0	b**üh**ne
œː	0153+02D0	**oeu**vre
œ	Not available	

Long vowels (Netherlands Dutch)

Nasalized vowels

IPA symbol	IPA Unicode	Example words
ɑ̃	0251+0303	ch**an**son
ɛ̃	025B+0303	t**im**bre
ɔ̃	0254+0303	b**on**b**on**
œ̃	0153+0303	parf**um**

Nasalized vowels (Netherlands Dutch)

Diphthongs

IPA symbol	IPA Unicode	Example words
ɛi	025B+0069	f**ij**n
ɑu	0251+0075	k**ou**
œy	0153+0079	h**ui**s

Diphthongs (Netherlands Dutch)

Semi vowels

IPA symbol	IPA Unicode	Example words
w	0077	**w**indows
j	006A	**j**as

Semi vowels (Netherlands Dutch)

Consonants

IPA symbol	IPA Unicode	Example words
p	0070	**p**ad
b	0062	**b**ad
t	0074	**t**ak
d	0064	**d**ak
k	006B	**k**at
g	0067	**g**oal
f	0066	**f**out
s	0073	**s**ap
z	007A	**z**at
ʃ	0283	**sj**aal
ʒ	0292	baga**g**e
x	0078	**g**aan, to**ch**
ɣ	0263	re**g**en
h	0068	**h**and
v	0076	le**v**en
ʋ	028B	**w**ang
l	006C	**l**and

r	0072	**r**and
ʁ	0281	abbatoi**r**
ɹ	0279	da**r**k
m	006D	**m**at
n	006E	**n**ek
ŋ	014B	e**ng**
ʧ	02A7	**ch**eck
ʤ	02A4	**j**azz
θ	03B8	**th**umbnail
ð	00F0	bro**th**er

Consonants (Netherlands Dutch)

English (Australian) symbols (obsolete)

The service supported the following symbols for the obsolete neural voices for Australian English. These voices were replaced with alternative expressive neural voices and removed from the service on 31 March 2023. The symbols are available to help users migrate from the obsolete Australian English neural voices to the new Australian English expressive neural voices.

Monophthongs

IPA symbol	IPA Unicode	Example words
ɒ	0252	n**o**t
æ	00E6	c**a**t
e	0065	p**e**t
ɛ	Not available	
I	026A	c**i**ty
i	0069	happ**y**
ʊ	028A	p**u**t
u	0075	infl**u**enza
ʌ	028C	r**u**n
ə	0259	**a**bout

Monophthongs (Australian English)

Diphthongs

IPA symbol	IPA Unicode	Example words
aI	0061+026A	r**i**se

eI	0065+026A	r**ai**se
ɔI	0254+026A	n**oi**se
aʊ	0061+028A	r**ou**se
əʊ	0259+028A	n**o**se
eə	0065+0259	st**ai**rs
ɛə	Not available	
Iə	026A+0259	f**ea**r
ʊə	028A+0259	c**u**re

Diphthongs (Australian English)

Long vowels

IPA symbol	IPA Unicode	Example words
ɑ:	0251+02D0	f**a**ther
i:	0069+02D0	s**ee**
ɔ:	0254+02D0	l**aw**
u:	0075+02D0	s**oo**n
ɜ:	025C+02D0	b**ir**d

Long vowels (Australian English)

Consonants

IPA symbol	IPA Unicode	Example words
p	0070	**p**en
b	0062	**b**ut
t	0074	**t**wo
d	0064	**d**o
tʃ	02A7	**ch**air
dʒ	02A4	**j**oy
k	006B	**c**at
g	0067	**g**et
f	0066	**f**ool
v	0076	**v**oice
θ	03B8	**th**ink

ð	00F0	**th**is
s	0073	**s**ee
z	007A	**z**oo
ʃ	0283	**sh**e
ʒ	0292	plea**s**ure
h	0068	**h**am
x	Not available	lo**ch**
ɬ	Not available	**Ll**antilio
m	006D	**m**an
m̩	Not available	Wilsh**am**
n	006E	**n**o
n̩	Not available	mort**en**
ŋ	014B	ri**ng**
l	006C	**l**eft
l̩	Not available	turt**le**
ɹ	0279	**wr**ong
ɻ	Not available	
r	Not available	
j	006A	**y**es
w	0077	**w**e

Consonants (Australian English)

Korean symbols (obsolete)

The service supported the following symbols for the obsolete neural voices for Korean. These voices were replaced with an alternative enhanced neural voice and removed from the service on 31 March 2023. The symbols are available to help users migrate from the obsolete Korean neural voices to the new Korean enhanced neural voice.

Monophthongs

IPA symbol	IPA Unicode	Example words
a	0061	아기
e	0065	에누리
i	0069	이발
o	006F	오리

u	0075	우산
ɛ	025B	애교
ø	00F8	외국
ʌ	028C	어머니
ɯ	026F	드디어

Monophthongs (Korean)

Diphthongs

IPA symbol	IPA Unicode	Example words
ja	006A+0061	야구
je	006A+0065	예술
jo	006A+006F	요리
ju	006A+0075	유자
jɛ	006A+025B	얘기
jʌ	006A+028C	여기
wa	0077+0061	와우
we	0077+0065	웨딩
wi	0077+0069	위기
wɛ	0077+025B	왜
wʌ	0077+028C	워낭
ɯi	026F+0069	의사

Diphthongs (Korean)

Onset consonants

IPA symbol	IPA Unicode	Example words
P	0070	바다
pʰ	0070+02B0	포도
b	0062	시비
t	0074	다음
tʰ	0074+02B0	토요일
d	0064	기도
t͡ɕ	02AB	저울

tɕʰ	02AB+02B0	치과
dʑ	02A3	기자
k	006B	구두
kʰ	006B+02B0	카드
g	0067	지구
s	0073	사과
h	0068	효과
x	0078	이해
m	006D	메기
n	006E	나무
l	006C	라면
r	0072	고리

Onset consonants (Korean)

Double onset consonants

IPA symbol	IPA Unicode	Example words
bb	0062+0062	뻐꾸기
dd	0064+0064	딸기
tɕtɕ	02AB+02AB	쭈꾸미
gg	0067+0067	꼬마
ss	0073+0073	싸움

Double onset consonants (Korean)

Coda consonants

IPA symbol	IPA Unicode	Example words
p	0070	집
t	0074	받침
k	006B	목
m	006D	힘
n	006E	손
ŋ	014B	장기
l	006C	날개

Coda consonants (Korean)

IBM Cloud security

Information security

IBM® is committed to providing our clients and partners with innovative data privacy, security, and governance solutions.

> ⚠ **Important:** Clients are responsible for ensuring their own compliance with various laws and regulations, including the European Union General Data Protection Regulation. Clients are solely responsible for obtaining advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulations that might affect the clients' business and any actions the clients might need to take to comply with such laws and regulations.

The products, services, and other capabilities described herein are not suitable for all client situations and might have restricted availability. IBM does not provide legal, accounting or auditing advice or represent or warrant that its services or products will ensure that clients are in compliance with any law or regulation.

If you need to request GDPR support for IBM Cloud® Watson resources that are created

- In the European Union (EU), see Requesting support for IBM Cloud Watson resources created in the European Union .
- Outside of the EU, see Requesting support for resources outside the European Union .

European Union General Data Protection Regulation (GDPR)

IBM is committed to providing our clients and partners with innovative data privacy, security and governance solutions to assist them on their journey to GDPR compliance.

Learn more about IBM's own GDPR readiness journey and our GDPR capabilities and offerings to support your compliance journey here.

Health Insurance Portability and Accountability Act (HIPAA)

IBM Cloud

US Health Insurance Portability and Accountability Act (HIPAA) support is available for Premium plans that are hosted in the Washington, DC, (`us-east`) and Dallas (`us-south`) locations. For more information, see Enabling HIPAA support for your account .

Do not include personal health information (PHI) in data that is to be added to custom models. Be sure to remove any PHI from data that you use for custom models.

Labeling and deleting data in the Text to Speech service

The IBM Watson® Text to Speech service enables you to delete all data that is associated with speech synthesis requests and with custom models. To delete data, you must do the following:

1. Use the `X-Watson-Metadata` header to associate a customer ID with data that is passed by a request to the service; see Specifying a customer ID .
2. Use the `DELETE /v1/user_data` method to delete all data that is associated with a specified customer ID; see Deleting data.

By default, no customer ID is associated with data.

> **Note:** Experimental and beta features are not intended for use with a production environment and therefore are not guaranteed to function as expected when labeling and deleting data. Experimental and beta features should not be used when implementing a solution that requires the labeling and deletion of data.

Specifying a customer ID

To associate a customer ID with data, include the `X-Watson-Metadata` header with the request that passes the information. You pass the string `customer_id={id}` as the argument of the header. The following example associates the customer ID `my_customer_ID` with the data passed with a `POST /v1/synthesize` request:

IBM Cloud

```
$ curl -X POST -u "apikey:{apikey}" \
--header "X-Watson-Metadata: customer_id=my_customer_ID" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--data "{\"text\":\"hello world\"}" \
```

```
--output hello_world.wav \
"{url}/v1/synthesize"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X POST \
--header "Authorization: Bearer {token}"\
--header "X-Watson-Metadata: customer_id=my_customer_ID" \
--header "Content-Type: application/json" \
--header "Accept: audio/wav" \
--data "{\"text\":\"hello world\"}" \
--output hello_world.wav \
"{url}/v1/synthesize"
```

A customer ID can include any characters except for the `;` (semicolon) and `=` (equals sign). Specify a random or generic string for the customer ID; do not specify a personally identifiable string, such as an email address or Twitter ID. You can specify different customer IDs with different requests. A customer ID that you specify is associated with the instance of the service whose credentials are used with the request; only credentials for that instance of the service can delete data associated with the ID.

Use the `X-Watson-Metadata` header with the following methods:

- With HTTP requests:
  - `POST /v1/synthesize`
  - `GET /v1/synthesize`

  The customer ID is associated with data that is sent with the request.
- With WebSocket requests:
  - `/v1/synthesize`

  You specify the customer ID with the `x-watson-metadata` query parameter to associate the ID with data that is sent with the request. You must URL-encode the argument to the query parameter, for example, `customer_id%3dmy_customer_ID`.
- With requests to add custom words to custom models:
  - `POST /v1/customizations/{customization_id}`
  - `POST /v1/customizations/{customization_id}/words`
  - `PUT /v1/customizations/{customization_id}/words/{word}`

  The customer ID is associated with the custom words that are added or updated by the request.

Deleting data

To delete all data that is associated with a customer ID, use the `DELETE /v1/user_data` method. You pass the string `customer_id={id}` as a query parameter with the request. The following example deletes all data for the customer ID `my_customer_ID`:

IBM Cloud

```
$ curl -X DELETE -u "apikey:{apikey}" \
"{url}/v1/user_data?customer_id=my_customer_ID"
```

IBM Cloud Pak for Data IBM Software Hub

```
$ curl -X DELETE \
--header "Authorization: Bearer {token}" \
"{url}/v1/user_data?customer_id=my_customer_ID"
```

The `/v1/user_data` method deletes all data that is associated with the specified customer ID, regardless of the method by which the information was added. The method has no effect if no data is associated with the customer ID. You must issue the request with credentials for the same instance of the service that was used to associate the customer ID with the data.

Deletion of all data for a Text to Speech service instance

IBM Cloud

If you delete an instance of the Text to Speech service from the IBM Cloud console, all data associated with that service instance is automatically deleted. This includes all custom models and word/translation pairs, and all data related to speech synthesis requests.

This data is purged automatically and regardless of whether a customer ID is associated with the data. Once you delete a service instance, you can no longer restore any of the deleted data.

Activity Tracking events for Text to Speech

IBM Cloud

IBM Cloud services, such as Text to Speech, generate activity tracking events.

Activity tracking events report on activities that change the state of a service in IBM Cloud. You can use the events to investigate abnormal activity and critical actions and to comply with regulatory audit requirements.

You can use IBM Cloud Activity Tracker Event Routing, a platform service, to route auditing events in your account to destinations of your choice by configuring targets and routes that define where activity tracking events are sent. For more information, see About IBM Cloud Activity Tracker Event Routing.

You can use IBM Cloud Logs to visualize and alert on events that are generated in your account and routed by IBM Cloud Activity Tracker Event Routing to an IBM Cloud Logs instance.

Locations where activity tracking events are generated

Locations where activity tracking events are sent to IBM Cloud Activity Tracker Event Routing

IBM Watson® Text to Speech sends activity tracking events by IBM Cloud Activity Tracker Event Routing in the regions that are indicated in the following table.

Dallas (us-south)	Washington (us-east)	Toronto (ca-tor)	Sao Paulo (br-sao)
Yes	Yes	No	No

Regions where activity tracking events are sent in Americas locations

Tokyo (jp-tok)	Sydney (au-syd)	Osaka (jp-osa)	Chennai (in-che)
Yes	Yes	No	No

Regions where activity tracking events are sent in Asia Pacific locations

Frankfurt (eu-de)	London (eu-gb)	Madrid (eu-es)
Yes	Yes	No

Regions where activity tracking events are sent in Europe locations

Viewing activity tracking events for Text to Speech

You can use IBM Cloud Logs to visualize and alert on events that are generated in your account and routed by IBM Cloud Activity Tracker Event Routing to an IBM Cloud Logs instance.

Launching IBM Cloud Logs from the Observability page

For information on launching the IBM Cloud Logs UI, see Launching the UI in the IBM Cloud Logs documentation..

Customization events

The following tables list the Text to Speech actions for model customization that generate events.

Note: The names of the actions for the customization events have changed. The old names are deprecated. Use the names in the following tables instead. For more information, see Updates to Activity Tracker actions for customization in the release notes for IBM Cloud.

Create events

Action	Description
text-to-speech.custom-model.create	Create a custom model (POST /v1/customizations).
text-to-speech.custom-model-word-list.create	Create a word list for a custom model (POST /v1/customizations/{customization_id}/words).
text-to-speech.custom-model-word.create	Create a word for a custom model (PUT /v1/customizations/{customization_id}/words/{word}).

Customization .create actions that generate events

Read events

Action	Description
text-to-speech.custom-model-list.read	Read a list of custom models created by a user (GET /v1/customizations).
text-to-speech.custom-model.read	Read a custom model (GET /v1/customizations/{customization_id}).
text-to-speech.custom-model-word-list.read	Read a word list for a custom model (GET /v1/customizations/{customization_id}/words).
text-to-speech.custom-model-word.read	Read a word for a custom model (GET /v1/customizations/{customization_id}/words/{word}).

Customization .read actions that generate events

Update event

Action	Description
text-to-speech.custom-model.update	Update a custom model (POST /v1/customizations/{customization_id}).

Customization .update action that generates an event

Delete events

Action	Description
text-to-speech.custom-model.delete	Delete a custom model (DELETE /v1/customizations/{customization_id}).
text-to-speech.custom-model-word.delete	Delete a word from a custom model (DELETE /v1/customizations/{customization_id}/words/{word}).

Customization .delete actions that generate events

GDPR event

The following table lists the Text to Speech action for General Data Protection Regulation (GDPR) that generates an event.

Action	Description
text-to-speech.gdpr-user-data.delete	Delete information for a user (DELETE /v1/user_data).

GDPR .delete action that generates an event

Public and private network endpoints

IBM Cloud

IBM Cloud® supports both public and private network endpoints for certain plans. Connections to private network endpoints do not require public internet access.

Private network endpoints support routing services over the IBM Cloud private network instead of the public network. A private network endpoint provides a unique IP address that is accessible to you without a VPN connection.

Enabling your account

> ⚠ **Important:** Private network endpoints are supported for paid plans. Check the plan information for your service to learn about the plans that support private network endpoints.

Your account must be configured before you can use private endpoints. To use private network endpoints, the following account features must be enabled for your account.

- Virtual routing and forwarding (VRF).
- Service endpoints. Enabling service endpoints means that all users in the account can connect to private network endpoints.

To enable VRF, you create a support case. To enable service endpoints, you use the IBM Cloud CLI. For more information about how to enable your account, see Enabling VRF and service endpoints.

Setting a private endpoint

After your account is enabled for VRF and service endpoints, you can add a private network endpoint to a service instance.

A service instance can have a private network endpoint, a public network endpoint, or both.

- Public: A service endpoint on the IBM Cloud public network.
- Private: A service endpoint that is accessible only on the IBM Cloud private network with no access from the public internet.
- Both public and private: Service endpoints that allow access over both networks.

Adding a private network endpoint

You add a private endpoint to a paid service instance from the service details page if you have a Manager or Writer service access role.

1. Go to your Resource list.
2. Click the name of a service instance that is on a paid plan. Lite plans do not support private network endpoints.
3. In the service details page, click the **Manage** tab.
4. Click **Add private network endpoint**.

Viewing your endpoint URL

The service endpoint URLs are different for private and public network endpoints. You can view the URL for an endpoint from the service details page.

1. Go to your Resource list.
2. Click the name of a service instance that has a private network endpoint.
3. In the service details page, click the **Manage** tab, and then click **Private Network Endpoint**.

What to do next

- Configure your account for VRF and Service endpoints.
- Modify your applications to use the new service endpoint URL.
- Read more about service endpoints.

Virtual Private Endpoints

IBM Cloud

IBM Cloud® Virtual Private Endpoints (VPE) for VPC enables you to connect to supported IBM Cloud® services from your VPC network by using the IP addresses of your choosing, allocated from a subnet within your VPC. See more details here.

> **Note:** This document applies to watsonx Assistant, Discovery, Speech to Text, and Text to Speech. Virtual Private Endpoints (VPEs) are available for these services in the Dallas, Washington, Frankfurt, London, Sydney, and Tokyo locations.

Prerequisites

- Have a Virtual Private Cloud (VPC)
- Have private endpoints enabled for your service instance

Instructions

- Create a VPE Gateway (VPEG) through the UI, CLI, or API.
- Creation of the VPEG is confirmed when an IP address is set in the details view of the VPEG page in the UI, CLI output of the `endpoint-gateway` command, or API details call.

You can verify by running `nslookup <endpoint>` on the private service endpoint of the Watson service from your VPC, for example:

```
# nslookup api.private.us-south.assistant.watson.cloud.ibm.com
Server:   127.0.0.53
Address:  127.0.0.53#53

Non-authoritative answer:
Name:   api.private.us-south.assistant.watson.cloud.ibm.com
Address: 10.240.0.9   <---- your VPE IP address
```

To make requests using the assigned IP address instead of just the private service endpoint as suggested here, you must do your own hostname resolution. For example:

```
$ curl -X POST "https://api.private.us-south.assistant.watson.cloud.ibm.com/v2/assistants" --connect ::10.240.0.9
```

```
$ curl -X POST "https://api.private.us-south.assistant.watson.cloud.ibm.com/v2/assistants" --resolve api.private.us-south.assistant.watson.cloud.ibm.com:443:10.240.0.9
```

Additional links

- IBM Cloud VPC
- VPE FAQ
- Accessing the VPE after setup
- Viewing Details of a VPE Gateway
- VPE Limitations
- Full VPC CLI reference

Service background

The science behind the service

The IBM Watson® Text to Speech service offers voices that rely on natural voices, expressive voices and enhanced neural voices. A brief overview of each of these voices follows.

Natural voices

The natural voices in the portfolio use an encoder-decoder architecture that disentangles timbre and prosody characteristics to better guide the synthesis. That characteristics are able to provide state-of-the-art performance in terms of naturalness and expressiveness. These novel natural voices use various techniques to provide an edge over the existing expressive voices, including:

- use of diffusion denoising models to better predict style and timbre features [1].
- use of model pre-trained by using large amounts of data, followed by model refinement with smaller, dedicated expressive corpora.
- minimizing modular approaches, and fully integrating the encoder and decoder into an end-to-end pipeline that is jointly trained.

On the encoder side, the architecture comprises:

- phonetic encoders to process the linguistic inputs, which consist of a sequence of phonemes (augmented with orthographic punctuation) generated from raw text by a rules-based front-end responsible for text normalization and phonetization.
- a diffusion model that is responsible for predicting latent representations for timbre and prosody, which is guided by a combination of a global speaker embedding and an optional reference prompt embedding.
- a prosody prediction model that generates explicit duration and normalized (speaker-agnostic) pitch and volume targets.
- a prosody de-normalization model that corrects the normalized pitch and volume targets based on a timbre latent representation

The decoder absorbs the information that is produced by the encoder modules to generate waveforms, which is guided by perceptual adversarial losses (involving mel-spectra and WavLM-based losses), deploying an advanced vocoder with a novel streaming support for improved latency.

Expressive voices

The expressive voices work a style-sensitive, prosody-controllable architecture based on non-attentive Tacotron2 acoustic model, augmented with a set of Hierarchical Prosodic Controls (HPCs) [2],[3]. At a high level, it contains the following components:

- an encoder module, which embeds the full inputs to the model, and which comprise the phoneme sequence (generated by the same modules that are deployed in the Natural Voices architecture), phrase-level linguistic features, and a style-vector.
- a prosodic module that use the output of the encoder to predict the HPCs: a nested sequence of speaker-agnostic prosodic descriptors that contain various statistics related to pitch, energy, and duration. These HPCs provide the fine-grained conditioning (for example, at the phone, word, and utterance levels) to help realize the distinct prosodic patterns associated with different styles.
- a non-auto-regressive decoder that takes the output of the encoder and HPC modules, plus a speaker embedding, to generate a sequence of spectral and periodicity features that are finally fed into a separately trained neural vocoder (an LPC Network) to generate high-quality audio.

Enhanced neural voices

Enhanced neural voices represent the oldest technology in the catalog, and use a modular, cascaded, fully Deep-Neural-Network (DNN) based approach to provide the back-end for speech synthesis [4]. Just like natural and expressive voices, a separate module is responsible for processing the text to normalize and extract phonetic sequences that are then input into:

- a prosody-prediction DNN that predicts the pitch and phoneme duration from textual features
- an acoustic feature DNN that uses these predicted prosody targets, plus phonetic information, to generate spectral and periodicity features
- a neural vocoder that uses these spectral features and generates an output waveform.

This modular approach has the advantage of enabling fast and simple training, and independent control of each component and fast run time performance.

References

[1]: Yinghao Aaron Li, Cong Han, Vinay Raghavan, Gavin Mischler, Nima Mesgarani -- StyleTTS 2: Towards Human-Level Text-to-Speech through Style Diffusion and Adversarial Training with Large Speech Language Models. Proc. NeurIPS 2023.

[2]: Slava Shechtman, Raul Fernandez, Alexander Sorin, and David Haws -- Synthesis of Expressive Speaking Styles with Limited Training Data in a Multi-Speaker, Prosody-Controllable Sequence-to-Sequence Architecture. Proc. Interspeech 2021, pp. 4693-4697.

[3]: Raul Fernandez, David Haws, Guy Lorberbom, Slava Shechtman, and Alexander Sorin -- Transplantation of Conversational Speaking Style with Interjections in Sequence-to-Sequence Speech Synthesis. Proc. Interspeech 2022, pp. 5488-5492.

[4]: Zvi Kons, Slava Shechtman, Alex Sorin, Carmel Rabinovitz, and Ron Hoory -- High Quality, Lightweight and Adaptable TTS Using LPCNet. Proc. Interspeech 2019, pp. 176-180.

High availability and disaster recovery

IBM Cloud

The IBM Watson® Text to Speech service is highly available within any IBM Cloud location (for example, Dallas or Washington, DC). However, recovering from potential disasters that affect an entire location requires planning and preparation.

You are responsible for understanding your configuration, customization, and usage of the service. You are also responsible for being ready to re-create an instance of the service in a new location and to restore your data in any location.

High availability

The Text to Speech service supports high availability with no single point of failure. The service achieves high availability automatically and transparently by means of the multi-zone region (MZR) feature provided by IBM Cloud.

IBM Cloud enables multiple zones that do not share a single point of failure within a single location. It also provides automatic load balancing across the zones within a region.

Disaster recovery

Disaster recovery can become an issue if an IBM Cloud location experiences a significant failure that includes the potential loss of data. Because MZR is not available across locations, you must wait for IBM to bring a location back online if it becomes unavailable. If underlying data services are compromised by the failure, you must also wait for IBM to restore those data services.

In the event of a catastrophic failure, IBM might not be able to recover data from database backups. In this case, you need to restore your data to return your service instance to its most recent state. You can restore the data to the same or to a different location.

For the Text to Speech service, only data for custom models, custom words, speaker models, and custom prompts is stored on IBM Cloud. Your disaster recovery plan includes knowing, preserving, and being prepared to restore your customization information.

Backing up custom models and speaker models

Preserve the following information about your custom models, custom entries, speaker models, and custom prompts:

- A list of all of your custom models and their definitions. To list information about your custom models:
  - Use the `GET /v1/customizations` method to list information about all custom models. For more information, see Querying all custom models.
  - Use the `GET /v1/customizations/{customization_id}` method to list information about a specified custom model. For more information, see Querying a custom model.
- Information about all custom entries (word/translation pairs) in your custom models:
  - Use the `GET /v1/customizations/{customization_id}/words` method to list information about all word/translation pairs from a custom model. For more information, see Querying all words from a custom model.
  - Use the `GET /v1/customizations/{customization_id}/words/{word}` method to list information about a specified word/translation pair from a custom model. For more information, see Querying a single word from a custom model.

It is a best practice to preserve this information in a format that you can use to re-create your customized resources in the event of a failure. Actively maintaining the information, and preparing the calls listed in the following section ahead of time, can enable you to recover as quickly as possible.

Restoring custom models and speaker models

If you need to recover from a disaster, you can use your backup information to re-create your custom models, custom entries, speaker models, and custom prompts:

1. To re-create your custom models, use the `POST /v1/customizations` method. For more information, see Creating a custom model.
2. To add multiple word/translation pairs to a custom model, use the `POST /v1/customizations/{customization_id}/words` method. For more information, see Adding multiple words to a custom model.
3. To add a single word/translation pair to a custom model, use the `POST /v1/customizations/{customization_id}/words/{word}` method. For more information, see Adding a single word to a custom model.

You need to re-create your custom models, speaker models, and custom prompts individually. You can add all of your custom entries to a custom model at once, in groups, or one at a time.

Watson SDKs

SDKs abstract much of the complexity associated with application development. By providing programming interfaces in languages that you already know, they can help you get up and running quickly with IBM Watson services.

Supported SDKs

The following Watson SDKs are supported by IBM:

- Java SDK
- Node.js SDK
- Python SDK

Tip: The API reference for each service includes information and examples for these SDKs.

Community SDKs

The following SDKs are available from the Watson community of developers:

- ABAP SDK for IBM Watson, using SAP NetWeaver
- Android SDK
- Go SDK
- Ruby SDK
- Salesforce SDK
- Swift SDK
- Unity SDK

SDK updates and deprecation

The supported Watson SDKs are updated according to the following guidelines.

Semantic versioning

Supported Watson SDKs adhere to semantic versioning with releases labeled as `{major}.{minor}.{patch}`.

Release frequency

SDKs are released independently and might not update on the same schedule.

- The current releases of the Watson SDKs are updated on a 2- to 6-week schedule. These releases are either minor updates or patches that do not include breaking changes. You can update to any version of the SDK with the same major version number.
- Major updates that might include breaking changes are released approximately every 6 months.

Deprecated release

When a major version is released, support continues on the previous major release for 12 months in a deprecation period. The deprecated release might be updated with bug fixes, but no new features will be added and documentation might not be available.

Obsolete release

After the 12-month deprecation period, a release is obsolete. The release might be functional but is unsupported and not updated. Update to the current release.

Usage FAQs

FAQs for IBM Watson® Text to Speech include questions about speech synthesis, supported languages, audio formats, and other topics. To find all FAQs for IBM Cloud®, see the FAQ library.

How do I access my service credentials?

How you access your service credentials depends on whether you are using Text to Speech with IBM Cloud® or IBM Cloud Pak® for Data. For more information about obtaining your credentials for both versions, see Before you begin in the getting started tutorial.

Once you have your service credentials, see the following topics for information about authenticating to the service:

- Authenticating to IBM Cloud
- Authenticating to IBM Cloud Pak for Data

What languages does the service support?

The Text to Speech service supports male and female voices in various spoken languages:

- The service offers *expressive neural voices* for English (Australian and United States).
- The services offers *enhanced neural voices* for Dutch Netherlands, English (United Kingdom and United States), French (Canadian and France), German, Italian, Japanese, Korean, Portuguese (Brazilian), and Spanish (Castilian, Latin American, and North American).

Some languages and voices are available only for IBM Cloud®, not for IBM Cloud Pak® for Data. For more information about the available voices for all languages, see Languages and voices.

How does the service synthesize audio?

The Text to Speech service offers voices that rely on neural technology to synthesize text to speech. The topic of synthesizing text to speech is inherently complex. For more information, see

- Languages and voices
- The science behind the service

What are the output audio formats?

By default, the Text to Speech service returns audio in Ogg format with the Opus codec (`audio/ogg;codecs=opus`). The service supports many other audio formats to suit your application needs. For more information, see Supported audio formats.

How do I convert my text to speech?

To submit text to the service for synthesized audio output, you make an HTTP or WebSocket request. You can use the API directly or use one of the Watson SDKs. Getting started offers examples of both the HTTP `POST /v1/synthesize` and `GET /v1/synthesize` methods. The API & SDK reference shows examples of all interfaces and methods.

There is no graphical user interface for submitting text. See the Text to Speech demo to try an example of the service in action. The demo accepts a small amount of your text as input to generate speech with different voices.

Can I change how the service interprets input text and produces synthesized audio?

You can use the Speech Synthesis Markup Language (SSML) to control aspects of the synthesis process such as pronunciation, volume, pitch, speed, and other attributes.

- For general information about SSML, see Understanding SSML.
- For information about the supported SSML elements, see SSML elements.

What programming languages can I use?

The service supports SDKs in many popular programming languages and platforms.

- For more information about the SDKs and links to them on GitHub, see Watson SDKs.
- For more information about all methods of the SDKs for the Text to Speech service, see the API & SDK reference.

What is the maximum amount of text that I can submit for synthesis?

You can submit the following maximum amount of text for a speech synthesis request with each of the service's method:

- HTTP `GET /v1/synthesize` method - Maximum of 8 KB of total input, which includes the input text, SSML, and the URL and headers.
- HTTP `POST /v1/synthesize` method - Maximum of 8 KB for the URL and headers. Maximum of 5 KB for the input text, including SSML.
- WebSocket `/v1/synthesize` method - Maximum of 5 KB of input text, including SSML.

All characters of the input, including whitespace and those that are part of SSML elements, are counted toward the data maximum. For billing purposes, whitespace characters are not counted. For more information, see Data limits.

How does customization work?

The customization interface of the Text to Speech service creates a dictionary of words and their translations for a specific language. This dictionary is referred to as a custom model. For more information, see Understanding customization.

How do I create a custom model?

Review the guidelines for working with the customization interface before you begin. Then, see the steps and examples for creating, querying, updating, and deleting custom models in Creating and managing custom models. Also review Creating and managing custom entries for examples and guidance about adding relevant training data.

Can I create a custom voice?

IBM Cloud

As a premium customer, you can work with IBM to train a new custom voice for your specific use case and target market. Creating a custom voice is different from customizing one of the service's existing voices. A custom voice is a unique new voice that is based on audio training data that the customer provides. IBM can train a custom voice with as little as one hour of training data.

To request a custom voice or for more information, complete and submit this IBM Request Form.

What limits exist for a custom model?

The following limits apply to all custom models:

- A word in a custom entry can contain a maximum of 49 characters.
- A translation in a custom entry can contain a maximum of 499 characters.
- A custom model can include a maximum of 20,000 custom entries.

For more information, see Rules for creating custom entries.

Where can I find plans and pricing information?

IBM Cloud

The Text to Speech service offers multiple pricing plans. For more information about pricing, see the Text to Speech service in the IBM Cloud Catalog.

IBM Cloud accessibility features and support

Learn about IBM Cloud® accessibility features, including keyboard navigation, screen reader support, and assistive technology compatibility for users with disabilities.

Accessibility features assist users who have a disability, such as restricted mobility or limited vision, to use information technology content successfully.

Accessibility features

IBM Cloud® includes the following major accessibility features:

- Keyboard-only operation
- Operations that use a screen reader

IBM Cloud uses the W3C Standard, WAI-ARIA 1.2, to ensure compliance to ensure compliance to US Section 508, Web Content Accessibility Guidelines (WCAG) 2.2, and EN 301 549. To take advantage of accessibility features, use the latest release of your screen reader in combination with the latest Chrome web browser that is supported by this product.

Interface information

Review the following information about the IBM Cloud user interface:

- If you are using a screen reader with the IBM Cloud web user interface or product documentation, use the latest version of Chrome with the latest release of the screen reader.
- If you are using keyboard operation only, ensure that your MacOS setting is enabled for `Keyboard navigation: Use keyboard navigation to move focus between controls`.
- The IBM Cloud user interfaces do not have content that flashes 2 - 55 times per second.
- The IBM Cloud web user interfaces rely on cascading style sheets to render content properly and to provide a usable experience. The application provides an equivalent way for low-vision users to use a user's system display settings, including high-contrast mode. You can control font size by using the device or web browser settings.
- The IBM Cloud web user interface includes WAI-ARIA navigational landmarks that you can use to quickly navigate to functional areas in the application.

Related accessibility information

The IBM Cloud platform accessibility compliance status is specifically for the IBM Cloud platform and platform documentation. There are subsections of the user interface that are owned by third-party products or services that host content within the platform, for which the IBM Cloud compliance record does not maintain or own the accessibility compliance status. If you are accessing any user interface or documentation for a service, you must request the compliance statements for that service. For example, if you are using an interface for IBM Cloud Kubernetes Service, you must request product accessibility information for that interface or documentation.

IBM and accessibility

For more information about the commitment that IBM has to accessibility, see IBM Accessibility.

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
2026-05-19