Pronunciation with SSML

Article • 09/24/2024

You can use Speech Synthesis Markup Language (SSML) with text to speech to specify how the speech is pronounced. For example, you can use SSML with phonemes and a custom lexicon to improve pronunciation. You can also use SSML to define how a word or mathematical expression is pronounced.

Refer to the following sections for details about how to use SSML elements to improve pronunciation. For more information about SSML syntax, see SSML document structure and events.

phoneme element

The `phoneme` element is used for phonetic pronunciation in SSML documents. Always provide human-readable speech as a fallback.

Phonetic alphabets are composed of phones, which are made up of letters, numbers, or characters, sometimes in combination. Each phone describes a unique sound of speech. The phonetic alphabet is in contrast to the Latin alphabet, where any letter might represent multiple spoken sounds. Consider the different `en-US` pronunciations of the letter "c" in the words "candy" and "cease" or the different pronunciations of the letter combination "th" in the words "thing" and "those."

> **Note**
>
> For a list of locales that support phonemes, see footnotes in the **language support** table.

Usage of the `phoneme` element's attributes are described in the following table.

Expand table

Attribute	Description	Required or optional
`alphabet`	The phonetic alphabet to use when you synthesize the pronunciation of the string in the `ph` attribute. The string that specifies the alphabet must	Optional

Attribute	Description	Required or optional
	be specified in lowercase letters. The following options are the possible alphabets that you can specify: - `ipa` – See SSML phonetic alphabets - `sapi` – See SSML phonetic alphabets - `ups` – See Universal Phone Set - `x-sampa` – See SSML phonetic alphabets The alphabet applies only to the `phoneme` in the element.	
`ph`	A string containing phones that specify the pronunciation of the word in the `phoneme` element. If the specified string contains unrecognized phones, text to speech rejects the entire SSML document and produces none of the speech output specified in the document. For `ipa`, to stress one syllable by placing stress symbol before this syllable, you need to mark all syllables for the word. Or else, the syllable before this stress symbol is stressed. For `sapi`, if you want to stress one syllable, you need to place the stress symbol after this syllable, whether or not all syllables of the word are marked.	Required

phoneme examples

The supported values for attributes of the `phoneme` element were described previously. In the first two examples, the values of `ph="tə.ˈmeɪ.toʊ"` or `ph="təmeɪˈtoʊ"` are specified to stress the syllable `meɪ`.

XML

```xml
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">
    <voice name="en-US-AvaNeural">
        <phoneme alphabet="ipa" ph="tə.ˈmeɪ.toʊ"> tomato </phoneme>
    </voice>
</speak>
```

XML

```xml
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">
```

```xml
    <voice name="en-US-AvaNeural">
        <phoneme alphabet="ipa" ph="təmeɪˈtoʊ"> tomato </phoneme>
    </voice>
</speak>
```

XML

```xml
<speak version="1.0" xmlns="https://www.w3.org/2001/10/synthesis" xml:lang="en-
US">
    <voice name="en-US-AvaNeural">
        <phoneme alphabet="sapi" ph="iy eh n y uw eh s"> en-US </phoneme>
    </voice>
</speak>
```

XML

```xml
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">
    <voice name="en-US-AvaNeural">
        <s>His name is Mike <phoneme alphabet="ups" ph="JH AU"> Zhou </phoneme>
</s>
    </voice>
</speak>
```

XML

```xml
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">
     <voice name="en-US-AvaNeural">
        <phoneme alphabet='x-sampa' ph='he."lou'>hello</phoneme>
    </voice>
</speak>
```

Custom lexicon

You can define how single entities (such as company, a medical term, or an emoji) are read in SSML by using the phoneme and sub elements. To define how multiple entities are read, create an XML structured custom lexicon file. Then you upload the custom lexicon XML file and reference it with the SSML `lexicon` element.

ⓘ **Note**

For a list of locales that support custom lexicon, see footnotes in the language support table.

The `lexicon` element is not supported by the Long Audio API. For long-form text to speech, use the batch synthesis API (Preview) instead.

Usage of the `lexicon` element's attributes are described in the following table.

Expand table

Attribute	Description	Required or optional
`uri`	The URI of the publicly accessible custom lexicon XML file with either the `.xml` or `.pls` file extension. Using Azure Blob Storage is recommended but not required. For more information about the custom lexicon file, see Pronunciation Lexicon Specification (PLS) Version 1.0 .	Required

Custom lexicon examples

The supported values for attributes of the `lexicon` element were described previously.

After you publish your custom lexicon, you can reference it from your SSML. The following SSML example references a custom lexicon that was uploaded to `https://www.example.com/customlexicon.xml`. We support lexicon URLs from Azure Blob Storage, Advanced Media Services (AMS) Storage, and GitHub. However, note that other public URLs may not be compatible.

XML

```xml
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
          xmlns:mstts="http://www.w3.org/2001/mstts"
          xml:lang="en-US">
    <voice name="en-US-AvaNeural">
        <lexicon uri="https://www.example.com/customlexicon.xml"/>
        BTW, we will be there probably at 8:00 tomorrow morning.
        Could you help leave a message to Robert Benigni for me?
    </voice>
</speak>
```

Custom lexicon file

To define how multiple entities are read, you can define them in a custom lexicon XML file with either the `.xml` or `.pls` file extension.

> ⓘ **Note**
>
> The custom lexicon file is a valid XML document, but it cannot be used as an SSML document.

Here are some limitations of the custom lexicon file:

- **File size**: The custom lexicon file size is limited to a maximum of 100 KB. If the file size exceeds the 100-KB limit, the synthesis request fails. You can split your lexicon into multiple lexicons and include them in SSML if the file size exceeds 100 KB.
- **Lexicon cache refresh**: The custom lexicon is cached with the URI as the key on text to speech when it's first loaded. The lexicon with the same URI isn't reloaded within 15 minutes, so the custom lexicon change needs to wait 15 minutes at the most to take effect.

The supported elements and attributes of a custom lexicon XML file are described in the Pronunciation Lexicon Specification (PLS) Version 1.0 . Here are some examples of the supported elements and attributes:

- The `lexicon` element contains at least one `lexeme` element. Lexicon contains the necessary `xml:lang` attribute to indicate which locale it should be applied for. One custom lexicon is limited to one locale by design, so if you apply it for a different locale, it doesn't work. The `lexicon` element also has an `alphabet` attribute to indicate the alphabet used in the lexicon. The possible values are `ipa` and `x-microsoft-sapi`.
- Each `lexeme` element contains at least one `grapheme` element and one or more `grapheme`, `alias`, and `phoneme` elements. The `lexeme` element is case sensitive in the custom lexicon. For example, if you only provide a phoneme for the `lexeme` "Hello", it doesn't work for the `lexeme` "hello".
- The `grapheme` element contains text that describes the orthography .
- The `alias` elements are used to indicate the pronunciation of an acronym or an abbreviated term.
- The `phoneme` element provides text that describes how the `lexeme` is pronounced. The syllable boundary is '.' in the IPA alphabet. The `phoneme` element can't contain white space when you use the IPA alphabet.

- When the `alias` and `phoneme` elements are provided with the same `grapheme` element, `alias` has higher priority.

Microsoft provides a validation tool for the custom lexicon that helps you find errors (with detailed error messages) in the custom lexicon file. Using the tool is recommended before you use the custom lexicon XML file in production with the Speech service.

Custom lexicon file examples

The following XML example (not SSML) would be contained in a custom lexicon `.xml` file. When you use this custom lexicon, "BTW" is read as "By the way." "Benigni" is read with the provided IPA "bɛˈniːnji."

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
      xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
        http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
      alphabet="ipa" xml:lang="en-US">
    <lexeme>
        <grapheme>BTW</grapheme>
        <alias>By the way</alias>
    </lexeme>
    <lexeme>
        <grapheme>Benigni</grapheme>
        <phoneme>bɛˈniːnji</phoneme>
    </lexeme>
    <lexeme>
        <grapheme>😃</grapheme>
        <alias>test emoji</alias>
    </lexeme>
</lexicon>
```

You can't directly set the pronunciation of a phrase by using the custom lexicon. If you need to set the pronunciation for an acronym or an abbreviated term, first provide an `alias`, and then associate the `phoneme` with that `alias`. For example:

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
```

```
      xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
        http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
      alphabet="ipa" xml:lang="en-US">
    <lexeme>
        <grapheme>Scotland MV</grapheme>
        <alias>ScotlandMV</alias>
    </lexeme>
    <lexeme>
        <grapheme>ScotlandMV</grapheme>
        <phoneme>ˈskɒtlənd.ˈmiːdiəm.weɪv</phoneme>
    </lexeme>
</lexicon>
```

You could also directly provide your expected `alias` for the acronym or abbreviated term. For example:

XML

```
  <lexeme>
    <grapheme>Scotland MV</grapheme>
    <alias>Scotland Media Wave</alias>
  </lexeme>
```

The preceding custom lexicon XML file examples use the IPA alphabet, which is also known as the IPA phone set. We suggest that you use the IPA because it's the international standard. For some IPA characters, they're the "precomposed" and "decomposed" version when they're being represented with Unicode. The custom lexicon only supports the decomposed Unicode.

The Speech service defines a phonetic set for these locales: `en-US`, `fr-FR`, `de-DE`, `es-ES`, `ja-JP`, `zh-CN`, `zh-HK`, and `zh-TW`. For more information on the detailed Speech service phonetic alphabet, see the Speech service phonetic sets.

You can use the `x-microsoft-sapi` as the value for the `alphabet` attribute with custom lexicons as demonstrated here:

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
      xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
        http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
      alphabet="x-microsoft-sapi" xml:lang="en-US">
  <lexeme>
    <grapheme>BTW</grapheme>
    <alias> By the way </alias>
  </lexeme>
  <lexeme>
    <grapheme> Benigni </grapheme>
    <phoneme> b eh 1 - n iy - n y iy </phoneme>
  </lexeme>
</lexicon>
```

say-as element

The `say-as` element indicates the content type, such as number or date, of the element's text. This element provides guidance to the speech synthesis engine about how to pronounce the text.

Usage of the `say-as` element's attributes are described in the following table.

Expand table

Attribute	Description	Required or optional
`interpret-as`	Indicates the content type of an element's text. For a list of types, see the following table.	Required
`format`	Provides additional information about the precise formatting of the element's text for content types that might have ambiguous formats. SSML defines formats for content types that use them. See the following table.	Optional
`detail`	Indicates the level of detail to be spoken. For example, this attribute might request that the speech synthesis engine pronounce punctuation marks. There are no standard values defined for `detail`.	Optional

The following content types are supported for the `interpret-as` and `format` attributes. Include the `format` attribute only if `format` column isn't empty in this table.

ⓘ Note

The `characters` and `spell-out` values for the `interpret-as` attribute are supported for all **text to speech locales**. Other `interpret-as` attribute values are supported for all locales of the following languages: Arabic, Catalan, Chinese, Danish, Dutch, English, French, Finnish, German, Hindi, Italian, Japanese, Korean, Norwegian, Polish, Portuguese, Russian, Spanish, and Swedish.

Expand table

interpret-as	format	Interpretation
`characters`, `spell-out`		The text is spoken as individual letters (spelled out). The speech synthesis engine pronounces: `<say-as interpret-as="characters">test</say-as>` As "T E S T."
`cardinal`, `number`	None	The text is spoken as a cardinal number. The speech synthesis engine pronounces: `There are <say-as interpret-as="cardinal">10</say-as> options` As "There are ten options."
`ordinal`	None	The text is spoken as an ordinal number. The speech synthesis engine pronounces: `Select the <say-as interpret-as="ordinal">3rd</say-as> option` As "Select the third option."
`number_digit`	None	The text is spoken as a sequence of individual digits. The speech synthesis engine pronounces: `<say-as interpret-as="number_digit">123456789</say-as>` As "1 2 3 4 5 6 7 8 9."
`fraction`	None	The text is spoken as a fractional number. The speech synthesis engine pronounces: `<say-as interpret-as="fraction">3/8</say-as> of an inch` As "three eighths of an inch."

interpret-as	format	Interpretation

`date`	dmy, mdy, ymd, ydm, ym, my, md, dm, d, m, y	The text is spoken as a date. The `format` attribute specifies the date's format (*d=day, m=month, and y=year*). The speech synthesis engine pronounces:
		`Today is <say-as interpret-as="date">10-12-2016</say-as>`
		As "Today is October twelfth two thousand sixteen."
		Pronounces:
		`Today is <say-as interpret-as="date" format="dmy">10-12-2016</say-as>`
		As "Today is December tenth two thousand sixteen."

`time`	hms12, hms24	The text is spoken as a time. The `format` attribute specifies whether the time is specified by using a 12-hour clock (hms12) or a 24-hour clock (hms24). Use a colon to separate numbers representing hours, minutes, and seconds. Here are some valid time examples: 12:35, 1:14:32, 08:15, and 02:50:45. The speech synthesis engine pronounces: `The train departs at <say-as interpret-as="time" format="hms12">4:00am</say-as>` As "The train departs at four A M."

`duration`	hms, hm, ms	The text is spoken as a duration. The `format` attribute specifies the duration's format (*h=hour, m=minute, and s=second*). The speech synthesis engine pronounces:
		`<say-as interpret-as="duration">01:18:30</say-as>`
		As "one hour eighteen minutes and thirty seconds".
		Pronounces:
		`<say-as interpret-as="duration" format="ms">01:18</say-as>`
		As "one minute and eighteen seconds".
		This tag is only supported on English and Spanish.

`telephone`	None	The text is spoken as a telephone number. The speech synthesis engine pronounces: `The number is <say-as interpret-as="telephone">(888) 555-1212</say-as>`

interpret-as	format	Interpretation
		As "My number is area code eight eight eight five five five one two one two."
`currency`	None	The text is spoken as a currency. The speech synthesis engine pronounces: `<say-as interpret-as="currency">99.9 USD</say-as>` As "ninety-nine US dollars and ninety cents."
`address`	None	The text is spoken as an address. The speech synthesis engine pronounces: `I'm at <say-as interpret-as="address">150th CT NE, Redmond, WA</say-as>` As "I'm at 150th Court Northeast Redmond Washington."

`name`	None	The text is spoken as a person's name. The speech synthesis engine pronounces:
		`<say-as interpret-as="name">ED</say-as>`
		As [æd].
		In Chinese names, some characters pronounce differently when they appear in a family name. For example, the speech synthesis engine says 仇 in
		`<say-as interpret-as="name">仇先生</say-as>`
		As [qiú] instead of [chóu].

say-as examples

The supported values for attributes of the `say-as` element were described previously.

The speech synthesis engine speaks the following example as "Your first request was for one room on October nineteenth twenty ten with early arrival at twelve thirty five PM."

XML

```xml
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
    <voice name="en-US-AvaMultilingualNeural">
        <p>
        Your <say-as interpret-as="ordinal"> 1st </say-as> request was for <say-as interpret-as="cardinal"> 1 </say-as> room
        on <say-as interpret-as="date" format="mdy"> 10/19/2010 </say-as>, with early arrival at <say-as interpret-as="time" format="hms12"> 12:35pm </say-as>.
        </p>
    </voice>
</speak>
```

sub element

Use the `sub` element to indicate that the alias attribute's text value should be pronounced instead of the element's enclosed text. In this way, the SSML contains both a spoken and written form.

Usage of the `sub` element's attributes are described in the following table.

Expand table

Attribute	Description	Required or optional
`alias`	The text value that should be pronounced instead of the element's enclosed text.	Required

sub examples

The supported values for attributes of the `sub` element were described previously.

The speech synthesis engine speaks the following example as "World Wide Web Consortium."

XML

```xml
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
    <voice name="en-US-AvaMultilingualNeural">
        <sub alias="World Wide Web Consortium">W3C</sub>
    </voice>
</speak>
```

Pronunciation with MathML

The Mathematical Markup Language (MathML) is an XML-compliant markup language that describes mathematical content and structure. The Speech service can use the MathML as input text to properly pronounce mathematical notations in the output audio.

> ⓘ **Note**
>
> The MathML elements (tags) are currently supported in the following locales: `de-DE`, `en-AU`, `en-GB`, `en-US`, `es-ES`, `es-MX`, `fr-CA`, `fr-FR`, `it-IT`, `ja-JP`, `ko-KR`, `pt-BR`, and `zh-CN`.

All elements from the MathML 2.0 and MathML 3.0 specifications are supported, except the MathML 3.0 Elementary Math elements.

Take note of these MathML elements and attributes:

- The `xmlns` attribute in `<math xmlns="http://www.w3.org/1998/Math/MathML">` is optional.
- The `semantics`, `annotation`, and `annotation-xml` elements don't output speech, so they're ignored.
- If an element isn't recognized, it's ignored, and the child elements within it are still processed.

The XML syntax doesn't support the MathML entities, so you must use the corresponding unicode characters to represent the entities, for example, the entity `&copy;` should be represented by its unicode characters `&#x00A9;`, otherwise an error occurs.

MathML examples

The text to speech output for this example is "a squared plus b squared equals c squared".

XML

```xml
<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis'
xmlns:mstts='http://www.w3.org/2001/mstts' xml:lang='en-US'>
```

```
    <voice name='en-US-JennyNeural'>
        <math xmlns='http://www.w3.org/1998/Math/MathML'>
            <msup>
                <mi>a</mi>
                <mn>2</mn>
            </msup>
            <mo>+</mo>
            <msup>
                <mi>b</mi>
                <mn>2</mn>
            </msup>
            <mo>=</mo>
            <msup>
                <mi>c</mi>
                <mn>2</mn>
            </msup>
        </math>
    </voice>
</speak>
```

Next steps

- SSML overview
- SSML document structure and events
- Language support: Voices, locales, languages

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A