

The Wayback Machine - <https://web.archive.org/web/20070929040443/http://www.w3.org:80/T...>



Speech Synthesis Markup Language (SSML) Version 1.1

W3C Working Draft 4 September 2007

This version:

<http://www.w3.org/TR/2007/WD-speech-synthesis11-20070904/>

Latest version:

<http://www.w3.org/TR/speech-synthesis11/>

Previous version:

<http://www.w3.org/TR/2007/WD-speech-synthesis11-20070611/>

Editors:

Daniel C. Burnett, Nuance Communications

双志伟 (Zhi Wei Shuang), IBM

Authors:

Paolo Baggia, Loquendo

Paul Bagshaw, France Telecom

Michael Bodell, Tellme

黄德智 (De Zhi Huang), France Telecom

黄力行 (Lixing Huang), Chinese Academy of Sciences

康永国 (Yongguo Kang), Panasonic

楼晓雁 (Lou Xiaoyan), Toshiba

Scott McGlashan, HP

蒙美玲 (Helen Meng), Chinese University of Hong Kong

陶建华 (JianHua Tao), Chinese Academy of Sciences

吴志勇 (Zhiyong Wu), Chinese University of Hong Kong

严峻 (Yan Jun), iFLYTEK

[Copyright](#) © 2007 [W3C](#)® ([MIT](#), [ERCIM](#), [Keio](#)), All Rights Reserved. W3C [liability](#), [trademark](#) and [document use](#) rules apply.

Abstract

The Voice Browser Working Group has sought to develop standards to enable access to the Web using spoken interaction. The Speech Synthesis Markup Language Specification is one of these standards and is designed to provide a rich, XML-based markup language for assisting the generation of synthetic speech in Web and other applications. The essential role of the markup language is to provide authors of synthesizable content a standard way to control aspects of speech such as pronunciation, volume, pitch, rate, etc. across different synthesis-capable platforms.

Status of this Document

Samsung v. Cerence
IPR2025-00460
Exhibit 1008

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the [W3C technical reports index](http://www.w3.org/TR/) at <http://www.w3.org/TR/>.

This is the 4 September 2007 Working Draft of "Speech Synthesis Markup Language (SSML) Version 1.1". Changes from the previous Working Draft can be found in [Appendix G](#).

This document enhances SSML 1.0 [[SSML](#)] to provide better support for a broader set of natural (human) languages. To determine in what ways, if any, SSML is limited by its design with respect to supporting languages that are in large commercial or emerging markets for speech synthesis technologies but for which there was limited or no participation by either native speakers or experts during the development of SSML 1.0, the W3C held three workshops on the Internationalization of SSML. The first workshop [[WS](#)], in Beijing, PRC, in October 2005, focused primarily on Chinese, Korean, and Japanese languages, and the second [[WS2](#)], in Crete, Greece, in May 2006, focused primarily on Arabic, Indian, and Eastern European languages. The third workshop [[WS3](#)], in Hyderabad, India, in January 2007, focused heavily on Indian and Middle Eastern languages. Information collected during these workshops was used to develop a requirements document [[REQS11](#)]. Changes from SSML 1.0 are motivated by these requirements.

This document is a [W3C Working Draft](#). It has been produced as part of the [Voice Browser Activity](#). The authors of this document are participants in the [Voice Browser Working Group](#). For more information see the [Voice Browser FAQ](#). The Working Group expects to advance this Working Draft to Recommendation status.

Comments are welcome on www-voice@w3.org ([archive](#)). See [W3C mailing list and archive usage guidelines](#). Please send comments by **4 October 2007**.

Publication as a Working Draft does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the [5 February 2004 W3C Patent Policy](#). W3C maintains a [public list of any patent disclosures](#) made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains [Essential Claim\(s\)](#) must disclose the information in accordance with [section 6 of the W3C Patent Policy](#).

Table of Contents

1. [Introduction](#)
 - 1.1 [Design Concepts](#)
 - 1.2 [Speech Synthesis Process Steps](#)
 - 1.3 [Document Generation, Applications and Contexts](#)
 - 1.4 [Platform-Dependent Output Behavior of SSML Content](#)
 - 1.5 [Terminology](#)
2. [SSML Documents](#)
 - 2.1 [Document Form](#)
 - 2.2 [Conformance](#)
 - 2.2.1 [Conforming Speech Synthesis Markup Language Fragments](#)
 - 2.2.2 [Conforming Stand-Alone Speech Synthesis Markup Language Documents](#)
 - 2.2.3 [Using SSML With Other Namespaces](#)
 - 2.2.4 [Conforming Speech Synthesis Markup Language Processors](#)
 - 2.2.5 [Conforming User Agent](#)
 - 2.3 [Integration With Other Markup Languages](#)
 - 2.3.1 [SMIL](#)
 - 2.3.2 [ACSS](#)

- 2.3.3 [VoiceXML](#)
- 2.4 [Fetching SSML Documents](#)
- 3. [Elements and Attributes](#)
 - 3.1 [Document Structure, Text Processing and Pronunciation](#)
 - 3.1.1 ["speak" Root Element](#)
 - 3.1.1.1 [Trimming Attributes](#)
 - 3.1.2 [Language: "xml:lang" Attribute](#)
 - 3.1.3 [Base URI: "xml:base" Attribute](#)
 - 3.1.3.1 [Resolving Relative URIs](#)
 - 3.1.4 [Identifier: "xml:id" Attribute](#)
 - 3.1.5 [Lexicon Documents](#)
 - 3.1.5.1 ["lexicon" Element](#)
 - 3.1.5.2 ["lookup" Element](#)
 - 3.1.6 ["meta" Element](#)
 - 3.1.7 ["metadata" Element](#)
 - 3.1.8 [Text Structure](#)
 - 3.1.8.1 ["p" and "s" Elements](#)
 - 3.1.8.2 ["token" and "w" Elements](#)
 - 3.1.9 ["say-as" Element](#)
 - 3.1.10 ["phoneme" Element](#)
 - 3.1.10.1 [Pronunciation Alphabet Registry](#)
 - 3.1.11 ["sub" Element](#)
 - 3.1.12 ["lang" Element](#)
 - 3.1.13 [Language Speaking Failure: "onlangfailure" Attribute](#)
 - 3.2 [Prosody and Style](#)
 - 3.2.1 ["voice" Element](#)
 - 3.2.2 ["emphasis" Element](#)
 - 3.2.3 ["break" Element](#)
 - 3.2.4 ["prosody" Element](#)
 - 3.3 [Other Elements](#)
 - 3.3.1 ["audio" Element](#)
 - 3.3.1.1 [Trimming Attributes](#)
 - 3.3.1.2 ["gain" Attribute](#)
 - 3.3.1.3 ["speed" Attribute](#)
 - 3.3.2 ["mark" Element](#)
 - 3.3.3 ["desc" Element](#)
- 4. [References](#)
- 5. [Acknowledgments](#)
- Appendix A. [Audio File Formats](#) (normative)
- Appendix B. [Internationalization](#) (normative)
- Appendix C. [MIME Types and File Suffix](#) (normative)
- Appendix D. [Schema for the Speech Synthesis Markup Language](#) (normative)
- Appendix E. [DTD for the Speech Synthesis Markup Language](#) (informative)
- Appendix F. [Example SSML](#) (informative)
- Appendix G. [Changes since SSML 1.0](#) (informative)

1. Introduction

This W3C specification is known as the Speech Synthesis Markup Language specification (SSML) and is based upon the JSGF and/or JSML specifications, which are owned by Sun Microsystems, Inc., California, U.S.A. The JSML specification can be found at [[JSML](#)].

SSML is part of a larger set of markup specifications for [voice browsers](#) developed through the open processes of the W3C. It is designed to provide a rich, XML-based markup language for assisting the generation of synthetic speech in Web and other applications. The essential role of the markup language is to give authors of synthesizable content a standard way to control

aspects of speech output such as pronunciation, volume, pitch, rate, etc. across different synthesis-capable platforms. A related initiative to establish a standard system for marking up text input is SABLE [SABLE], which tried to integrate many different XML-based markups for [speech synthesis](#) into a new one. The activity carried out in SABLE was also used as the main starting point for defining the Speech Synthesis Markup Requirements for Voice Markup Languages [REQS]. Since then, SABLE itself has not undergone any further development.

The intended use of SSML is to improve the quality of synthesized content. Different markup elements impact different stages of the synthesis process (see [Section 1.2](#)). The markup may be produced either automatically, for instance via XSLT or CSS3 from an XHTML document, or by human authoring. Markup may be present within a complete SSML document (see [Section 2.2.2](#)) or as part of a fragment (see [Section 2.2.1](#)) embedded in another language, although no interactions with other languages are specified as part of SSML itself. Most of the markup included in SSML is suitable for use by the majority of content developers; however, some advanced features like [phoneme](#) and [prosody](#) (e.g. for speech contour design) may require specialized knowledge.

1.1 Design Concepts

The design and standardization process has followed from the Speech Synthesis Markup Requirements for Voice Markup Languages [REQS].

The following items were the key design criteria.

- *Consistency*: provide predictable control of voice output across platforms and across [speech synthesis](#) implementations.
- *Interoperability*: support use along with other W3C specifications including (but not limited to) VoiceXML, aural Cascading Style Sheets and SMIL.
- *Generality*: support speech output for a wide range of applications with varied speech content.
- *Internationalization*: Enable speech output in a large number of languages within or across documents.
- *Generation and Readability*: Support automatic generation and hand authoring of documents. The documents should be human-readable.
- *Implementable*: The specification should be implementable with existing, generally available technology, and the number of optional features should be minimal.

1.2 Speech Synthesis Process Steps

A [Text-To-Speech](#) system (a [synthesis processor](#)) that supports SSML will be responsible for rendering a document as spoken output and for using the information contained in the markup to render the document as intended by the author.

Document creation: A text document provided as input to the [synthesis processor](#) may be produced automatically, by human authoring, or through a combination of these forms. SSML defines the form of the document.

Document processing: The following are the six major processing steps undertaken by a [synthesis processor](#) to convert marked-up text input into automatically generated voice output. The markup language is designed to be sufficiently rich so as to allow control over each of the steps described below so that the document author (human or machine) can control the final voice output. Although each step below is divided into "markup support" and "non-markup behavior", actual behavior is usually a mix of the two and varies depending on the tag. The processor has the ultimate authority to ensure that what it produces is pronounceable (and ideally intelligible). In general the markup provides a way for the author to make prosodic and other information available to the processor, typically information the processor would be unable to

acquire on its own. It is then up to the processor to determine whether and in what way to use the information.

1. **XML parse:** An XML parser is used to extract the document tree and content from the incoming text document. The structure, tags and attributes obtained in this step influence each of the following steps.
2. **Structure analysis:** The structure of a document influences the way in which a document should be read. For example, there are common speaking patterns associated with paragraphs and sentences.
 - *Markup support:* The [p](#) and [s](#) elements defined in SSML explicitly indicate document structures that affect the speech output.
 - *Non-markup behavior:* In documents and parts of documents where these elements are not used, the [synthesis processor](#) is responsible for inferring the structure by automated analysis of the text, often using punctuation and other language-specific data.
3. **Text normalization:** All written languages have special constructs that require a conversion of the written form (orthographic form) into the spoken form. Text normalization is an automated process of the [synthesis processor](#) that performs this conversion. For example, for English, when "\$200" appears in a document it may be spoken as "two hundred dollars". Similarly, "1/2" may be spoken as "half", "January second", "February first", "one of two" and so on. By the end of this step the text to be spoken has been converted completely into tokens. The exact details of what constitutes a token are language-specific. In English, tokens are usually separated by white space and are typically words. For languages with different tokenization behavior, the term "word" in this specification is intended to mean an appropriately comparable unit. Tokens in SSML cannot span markup tags except within the [token](#) and [w](#) elements. A simple English example is "cup<break/>board"; outside the [token](#) and [w](#) elements, the [synthesis processor](#) will treat this as the two tokens "cup" and "board" rather than as one token (word) with a pause in the middle. Breaking one token into multiple tokens this way will likely affect how the processor treats it.
 - *Markup support:* The [say-as](#) element can be used in the input document to explicitly indicate the presence and type of these constructs and to resolve ambiguities. The set of constructs that can be marked has not yet been defined but might include dates, times, numbers, acronyms, currency amounts and more. Note that many acronyms and abbreviations can be handled by the author via direct text replacement or by use of the [sub](#) element, e.g. "BBC" can be written as "B B C" and "AAA" can be written as "triple A". These replacement written forms will likely be pronounced as one would want the original acronyms to be pronounced. In the case of Japanese text, if you have a [synthesis processor](#) that supports both Kanji and kana, you may be able to use the [sub](#) element to identify whether 今日は should be spoken as きょうは ("kyou wa" = "today") or こんにちは ("konnichiwa" = "hello").
 - *Non-markup behavior:* For text content that is not marked with the [say-as](#) element the [synthesis processor](#) is expected to make a reasonable effort to automatically locate and convert these constructs to a speakable form. Because of inherent ambiguities (such as the "1/2" example above) and because of the wide range of possible constructs in any language, this process may introduce errors in the speech output and may cause different processors to render the same document differently.
4. **Text-to-phoneme conversion:** Once the [synthesis processor](#) has determined the set of tokens to be spoken, it must derive pronunciations for each token. Pronunciations may be conveniently described as sequences of phonemes, which are units of sound in a language that serve to distinguish one word from another. Each language (and sometimes each

national or dialect variant of a language) has a specific phoneme set: e.g., most US English dialects have around 45 phonemes, Hawai'ian has between 12 and 18 (depending on who you ask), and some languages have more than 100! This conversion is made complex by a number of issues. One issue is that there are differences between written and spoken forms of a language, and these differences can lead to indeterminacy or ambiguity in the pronunciation of written words. For example, compared with their spoken form, words in Hebrew and Arabic are usually written with no vowels, or only a few vowels specified. In many languages the same written word may have many spoken forms. For example, in English, "read" may be spoken as "reed" (I will read the book) or "red" (I have read the book). Both human speakers and synthesis processors can pronounce these words correctly in context but may have difficulty without context (see "Non-markup behavior" below). Another issue is the handling of words with non-standard spellings or pronunciations. For example, an English [synthesis processor](#) will often have trouble determining how to speak some non-English-origin names, e.g. "Caius College" (pronounced "keys college") and President Tito (pronounced "sutto"), the president of the Republic of Kiribati (pronounced "kiribass").

- *Markup support:* The [phoneme](#) element allows a phonemic sequence to be provided for any token or token sequence. This provides the content creator with explicit control over pronunciations. The [say-as](#) element might also be used to indicate that text is a proper name that may allow a [synthesis processor](#) to apply special rules to determine a pronunciation. The [lexicon](#) and [lookup](#) elements can be used to reference external definitions of pronunciations. These elements can be particularly useful for acronyms and abbreviations that the processor is unable to resolve via its own [text normalization](#) and that are not addressable via direct text substitution or the [sub](#) element (see paragraph 3, above).
- *Non-markup behavior:* In the absence of a [phoneme](#) element the [synthesis processor](#) *must* apply automated capabilities to determine pronunciations. This is typically achieved by looking up tokens in a pronunciation dictionary (which may be language-dependent) and applying rules to determine other pronunciations. [Synthesis processors](#) are designed to perform text-to-phoneme conversions so most words of most documents can be handled automatically. As an alternative to relying upon the processor, authors may choose to perform some conversions themselves prior to encoding in SSML. Written words with indeterminate or ambiguous pronunciations could be replaced by words with an unambiguous pronunciation; for example, in the case of "read", "I will reed the book". Authors should be aware, however, that the resulting SSML document may not be optimal for visual display.

5. Prosody analysis: Prosody is the set of features of speech output that includes the pitch (also called intonation or melody), the timing (or rhythm), the pausing, the speaking rate, the emphasis on words and many other features. Producing human-like prosody is important for making speech sound natural and for correctly conveying the meaning of spoken language.

- *Markup support:* The [emphasis](#) element, [break](#) element and [prosody](#) element may all be used by document creators to guide the [synthesis processor](#) in generating appropriate prosodic features in the speech output.
- *Non-markup behavior:* In the absence of these elements, [synthesis processors](#) are expert (but not perfect) in automatically generating suitable prosody. This is achieved through analysis of the document structure, sentence syntax, and other information that can be inferred from the text input.

While most of the elements of SSML can be considered high-level in that they provide either content to be spoken or logical descriptions of style, the [break](#) and [prosody](#) elements mentioned above operate at a later point in the process and thus must coexist both with uses of the [emphasis](#) element and with the processor's own determinations of prosodic behavior. Unless specified in the appropriate sections, details of the interactions between

the processor's own determinations and those provided by the author at this level are processor-specific. Authors are encouraged not to casually or arbitrarily mix these two levels of control.

6. **Waveform production:** The phonemes and prosodic information are used by the [synthesis processor](#) in the production of the audio waveform. There are many approaches to this processing step so there may be considerable processor-specific variation.
 - *Markup support:* The [voice](#) element allows the document creator to request a particular voice or specific voice qualities (e.g. a young male voice). The [audio](#) element allows for insertion of recorded audio data into the output stream. Rendering can be restricted to a subset of the document by using the trimming attributes on the [speak](#) element.

1.3 Document Generation, Applications and Contexts

There are many classes of document creator that will produce marked-up documents to be spoken by a [synthesis processor](#). Not all document creators (including human and machine) have access to information that can be used in all of the elements or in each of the processing steps described in the [previous section](#). The following are some of the common cases.

- The document creator has no access to information to mark up the text. All processing steps in the [synthesis processor](#) must be performed fully automatically on *raw text*. The document requires only the containing [speak](#) element to indicate the content is to be spoken.
- When marked text is generated programmatically the creator may have specific knowledge of the structure and/or special text constructs in some or all of the document. For example, an email reader can mark the location of the time and date of receipt of email. Such applications may use elements that affect structure, [text normalization](#), prosody and possibly text-to-phoneme conversion.
- Some document creators make considerable effort to mark as many details of the document as possible to ensure consistent speech quality across platforms and to more precisely specify output qualities. In these cases, the markup may use any or all of the available elements to tightly control the speech output. For example, prompts generated in telephony and [voice browser](#) applications may be fine-tuned to maximize the effectiveness of the overall system.
- The most advanced document creators may skip the higher-level markup (structure, [text normalization](#), text-to-phoneme conversion, and prosody analysis) and produce low-level [speech synthesis](#) markup for segments of documents or for entire documents. This typically requires tools to generate sequences of phonemes, plus pitch and timing information. For instance, tools that do "copy synthesis" or "prosody transplant" try to emulate human speech by copying properties from recordings.

The following are important instances of architectures or designs from which marked-up synthesis documents will be generated. The language design is intended to facilitate each of these approaches.

- *Dialog language:* It is a requirement that it *should* be possible to include documents marked with SSML into the dialog description document to be produced by the Voice Browser Working Group.
- *Interoperability with aural CSS (ACSS):* Any HTML processor that is aural CSS-enabled can produce SSML. ACSS is covered in [Section 19 of the Cascading Style Sheets, level 2 \(CSS2\) Specification \[CSS2 §19\]](#). This usage of [speech synthesis](#) facilitates improved accessibility to existing HTML and XHTML content.

- *Application-specific style sheet processing:* As mentioned above, there are classes of applications that have knowledge of text content to be spoken, and that can be incorporated into the [speech synthesis](#) markup to enhance rendering of the document. In many cases, it is expected that the application will use style sheets to perform transformations of existing XML documents to SSML. This is equivalent to the use of ACSS with HTML and once again SSML is the resulting representation to be passed to the [synthesis processor](#). In this context, SSML may be viewed as a superset of [ACSS](#) [CSS2§19] capabilities, excepting spatial audio.

1.4 Platform-Dependent Output Behavior of SSML Content

SSML provides a standard way to specify gross properties of synthetic speech production such as pronunciation, volume, pitch, rate, etc. Exact specification of synthetic speech output behavior across disparate processors, however, is beyond the scope of this document.

Unless otherwise specified, markup values are merely indications rather than absolutes. For example, it is possible for an author to explicitly indicate the duration of a text segment and also indicate an explicit duration for a subset of that text segment. If the two durations result in a text segment that the [synthesis processor](#) cannot reasonably render, the processor is permitted to modify the durations as needed to render the text segment.

1.5 Terminology

Requirements terms

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. However, for readability, these words do not appear in all uppercase letters in this specification.

At user option

A conforming [synthesis processor](#) *may* or *must* (depending on the modal verb in the sentence) behave as described; if it does, it *must* provide users a means to enable or disable the behavior described.

Error

Results are undefined. A conforming [synthesis processor](#) *may* detect and report an error and *may* recover from it.

Media Type

A *media type* (defined in [RFC2045] and [RFC2046]) specifies the nature of a linked resource. Media types are case insensitive. A list of registered media types is available for download [TYPES]. See [Appendix C](#) for information on media types for SSML.

Speech Synthesis

The process of automatic generation of speech output from data input which may include plain text, marked up text or binary objects.

Synthesis Processor

A [Text-To-Speech](#) system that accepts SSML documents as input and renders them as spoken output.

Text-To-Speech

The process of automatic generation of speech output from text or annotated text input.

A URI is a unifying syntax for the expression of names and addresses of objects on the network as used in the World Wide Web. A URI is defined as any legal [anyURI](#) primitive as defined in XML Schema Part 2: Datatypes [SCHEMA2 §3.2.17]. For informational purposes only, [RFC3986] and [RFC2732] may be useful in understanding the structure, format, and use of URIs. Note that IRIs (see [RFC3987]) are permitted within the above definition of URI. Any relative URI reference *must* be resolved according to the rules given in [Section 3.1.3.1](#). In this specification URIs are provided as attributes to elements, for example in the [audio](#) and [lexicon](#) elements.

Voice Browser

A device which interprets a (voice) markup language and is capable of generating voice output and/or interpreting voice input, and possibly other input/output modalities.

2. SSML Documents

2.1 Document Form

A legal stand-alone Speech Synthesis Markup Language document *must* have a legal XML Prolog [XML 1.0 or XML 1.1, as appropriate, §2.8]. If present, the *optional* DOCTYPE *must* read as follows:

```
<!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESIS 1.0//EN"
"http://www.w3.org/TR/speech-synthesis/synthesis.dtd">
```

The XML prolog is followed by the root [speak](#) element. See [Section 3.1.1](#) for details on this element.

The [speak](#) element *must* designate the SSML namespace. This can be achieved by declaring an `xmlns` attribute or an attribute with an "xmlns" prefix. See [XMLNS 1.0 or XMLNS 1.1, as appropriate, §2] for details. Note that when the `xmlns` attribute is used alone, it sets the default namespace for the element on which it appears and for any child elements. The namespace for SSML is defined to be <http://www.w3.org/2001/10/synthesis>.

It is *recommended* that the [speak](#) element also indicate the location of the SSML schema (see [Appendix D](#)) via the [xsi:schemaLocation](#) attribute from [SCHEMA1 §2.6.3]. Although such indication is not required, to encourage it this document provides such indication on all of the examples.

The following are two examples of legal SSML headers:

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
```

```
<?xml version="1.0"?>
<!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESIS 1.0//EN"
  "http://www.w3.org/TR/speech-synthesis/synthesis.dtd">
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xml:lang="en-US">
```

The [meta](#), [metadata](#) and [lexicon](#) elements *must* occur before all other elements and text contained within the root [speak](#) element. There are no other ordering constraints on the elements in this specification.

2.2. Conformance

2.2.1 Conforming Speech Synthesis Markup Language Fragments

A document fragment is a *Conforming Speech Synthesis Markup Language Fragment* if:

- it conforms to the criteria for [Conforming Stand-Alone Speech Synthesis Markup Language Documents](#) after:
 - with the exception of [xml:lang](#) and [xml:base](#), all non-synthesis namespace elements and attributes and all `xmlns` attributes which refer to non-synthesis namespace elements are removed from the document,
 - and, if the [speak](#) element does not already designate the synthesis namespace using the `xmlns` attribute, then `xmlns="http://www.w3.org/2001/10/synthesis"` is added to the element.

2.2.2 Conforming Stand-Alone Speech Synthesis Markup Language Documents

A document is a *Conforming Stand-Alone Speech Synthesis Markup Language Document* if it meets both the following conditions:

- It is a well-formed XML document [[XML 1.0](#) or [XML 1.1](#) §2.1] conforming to Namespaces in XML (1.0 [[XMLNS 1.0](#)] or 1.1 [[XMLNS 1.1](#)], respectively).
- It is a valid XML document [[XML 1.0](#) or [XML 1.1](#) §2.8] which adheres to the specification described in this document ([Speech Synthesis Markup Language Specification](#)) including the constraints expressed in the Schema (see [Appendix D](#)) and having an XML Prolog and [speak](#) root element as specified in [Section 2.1](#).

The SSML specification and these conformance criteria provide no designated size limits on any aspect of synthesis documents. There are no maximum values on the number of elements, the amount of character data, or the number of characters in attribute values.

2.2.3 Using SSML with other Namespaces

The synthesis namespace *may* be used with other XML namespaces as per the appropriate Namespaces in XML Recommendation (1.0 [[XMLNS 1.0](#)] or 1.1 [[XMLNS 1.1](#)], depending on the version of XML being used). Future work by W3C is expected to address ways to specify conformance for documents involving multiple namespaces. Language-specific (i.e. non-SSML) elements may be inserted into SSML using an appropriate namespace. However, such content would only be rendered by a [synthesis processor](#) that supported the custom markup. Here is an example of how one might insert Ruby [[RUBY](#)] elements into SSML:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="ja">
  <!-- It's 20 July today. -->
  <s>今日は七月
    <xhtml:ruby>
      <xhtml:rb>二十日</xhtml:rb>
      <xhtml:rt role="alphabet:x-JEITA">ハツカ</xhtml:rt>
    </xhtml:ruby>
    です。
  </s>

  <!-- It's 20 July today. -->
  <s>今日は七月
    <xhtml:ruby>
      <xhtml:rb>二十日</xhtml:rb>
      <xhtml:rt role="alphabet:x-JEITA">ニジューニチ</xhtml:rt>
```

```
</xhtml:ruby>
  です。
</s>
</speak>
```

2.2.4 Conforming Speech Synthesis Markup Language Processors

A Speech Synthesis Markup Language processor is a program that can parse and process [Conforming Stand-Alone Speech Synthesis Markup Language documents](#).

In a *Conforming Speech Synthesis Markup Language Processor*, the XML parser *must* be able to parse and process all XML constructs defined by XML 1.0 [[XML 1.0](#)] and XML 1.1 [[XML 1.1](#)] and the corresponding versions of Namespaces in XML (1.0 [[XMLNS 1.0](#)] and 1.1 [[XMLNS 1.1](#)]). This XML parser is not required to perform validation of an SSML document as per its schema or DTD; this implies that during processing of an SSML document it is *optional* to apply or expand external entity references defined in an external DTD.

A Conforming Speech Synthesis Markup Language Processor *must* correctly understand and apply the semantics of each markup element as described by this document.

A Conforming Speech Synthesis Markup Language Processor *must* meet the following requirements for handling of natural (human) languages:

- A Conforming Speech Synthesis Markup Language Processor is *required* to parse all legal natural language declarations successfully.
- A Conforming Speech Synthesis Markup Language Processor may be able to apply the semantics of markup languages which refer to more than one natural language. When a processor is able to support each natural language in the set but is unable to handle them concurrently it *should* inform the hosting environment. When the set includes one or more natural languages that are not supported by the processor it *should* inform the hosting environment.
- A Conforming Speech Synthesis Markup Language Processor *may* implement natural languages by approximate substitutions according to a documented, processor-specific behavior. For example, a US English synthesis processor could process British English input.

When a Conforming Speech Synthesis Markup Language Processor encounters elements or attributes, other than [xml:lang](#) and [xml:base](#), in a non-synthesis namespace it *may*:

- ignore the non-standard elements and/or attributes
- or, process the non-standard elements and/or attributes
- or, reject the document containing those elements and/or attributes

There is, however, no conformance requirement with respect to performance characteristics of the Speech Synthesis Markup Language Processor. For instance, no statement is required regarding the accuracy, speed or other characteristics of speech produced by the processor. No statement is made regarding the size of input that a Speech Synthesis Markup Language Processor must support.

2.2.5 Conforming User Agent

A *Conforming User Agent* is a [Conforming Speech Synthesis Markup Language Processor](#) that is capable of accepting an SSML document as input and producing a spoken output by using the information contained in the markup to render the document as intended by the author. A Conforming User Agent *must* support at least one natural language.

Since the output cannot be guaranteed to be a correct representation of all the markup contained in the input there is no conformance requirement regarding accuracy. A conformance test *may*,

however, require some examples of correct synthesis of a reference document to determine conformance.

2.3 Integration With Other Markup Languages

2.3.1 SMIL

The Synchronized Multimedia Integration Language (SMIL, pronounced "smile") [[SMIL](#)] enables simple authoring of interactive audiovisual presentations. SMIL is typically used for "rich media"/multimedia presentations which integrate streaming audio and video with images, text or any other media type. SMIL is an easy-to-learn HTML-like language, and many SMIL presentations are written using a simple text editor. See the SMIL/SSML integration examples in [Appendix F](#).

2.3.2 ACSS

Aural Cascading Style Sheets [[CSS2](#) §19] are employed to augment standard visual forms of documents (like HTML) with additional elements that assist in the synthesis of the text into audio. In comparison to SSML, ACSS-generated documents are capable of more complex specifications of the audio sequence, including the designation of 3D location of the audio source. Many of the other ACSS elements overlap SSML functionality, especially in the specification of voice type/quality. SSML may be viewed as a superset of ACSS capabilities, excepting spatial audio.

2.3.3 VoiceXML

The Voice Extensible Markup Language [[VXML](#)] enables Web-based development and content-delivery for interactive voice response applications (see [voice browser](#)). VoiceXML supports [speech synthesis](#), recording and playback of digitized audio, speech recognition, DTMF input, telephony call control, and form-driven mixed initiative dialogs. VoiceXML 2.0 extends SSML for the markup of text to be synthesized. For an example of the integration between VoiceXML and SSML see [Appendix F](#).

2.4 Fetching SSML Documents

The fetching and caching behavior of SSML documents is defined by the environment in which the [synthesis processor](#) operates. In a VoiceXML interpreter context for example, the caching policy is determined by the VoiceXML interpreter.

3. Elements and Attributes

The following elements and attributes are defined in this specification.

3.1 [Document Structure, Text Processing and Pronunciation](#)

- 3.1.1 ["speak" Root Element](#)
 - 3.1.1.1 [Trimming Attributes](#)
- 3.1.2 [Language: "xml:lang" Attribute](#)
- 3.1.3 [Base URI: "xml:base" Attribute](#)
 - 3.1.3.1 [Resolving Relative URIs](#)
- 3.1.4 [Identifier: "xml:id" Attribute](#)
- 3.1.5 [Lexicon Documents](#)
 - 3.1.5.1 ["lexicon" Element](#)
 - 3.1.5.2 ["lookup" Element](#)
- 3.1.6 ["meta" Element](#)
- 3.1.7 ["metadata" Element](#)
- 3.1.8 [Text Structure](#)

- 3.1.8.1 ["p" and "s" Elements](#)
- 3.1.8.2 ["token" and "w" Elements](#)
- 3.1.9 ["say-as" Element](#)
- 3.1.10 ["phoneme" Element](#)
 - 3.1.10.1 [Pronunciation Alphabet Registry](#)
- 3.1.11 ["sub" Element](#)
- 3.1.12 ["lang" Element](#)
- 3.1.13 [Language Speaking Failure: "onlangfailure" Attribute](#)
- 3.2 [Prosody and Style](#)
 - 3.2.1 ["voice" Element](#)
 - 3.2.2 ["emphasis" Element](#)
 - 3.2.3 ["break" Element](#)
 - 3.2.4 ["prosody" Element](#)
- 3.3 [Other Elements](#)
 - 3.3.1 ["audio" Element](#)
 - 3.3.1.1 [Trimming Attributes](#)
 - 3.3.1.2 ["gain" Attribute](#)
 - 3.3.1.3 ["speed" Attribute](#)
 - 3.3.2 ["mark" Element](#)
 - 3.3.3 ["desc" Element](#)

3.1 Document Structure, Text Processing and Pronunciation

3.1.1 speak Root Element

The Speech Synthesis Markup Language is an XML application. The root element is [speak](#).

[xml:lang](#) is a *required* attribute specifying the language of the root document.

[xml:base](#) is an *optional* attribute specifying the Base [URI](#) of the root document.

The `version` attribute is a *required* attribute that indicates the version of the specification to be used for the document and *must* have the value "1.1".

The trimming attributes are specified in a subsection, below.

Before the [speak](#) element is executed, the [synthesis processor](#) *must* select a default voice. Note that a language speaking failure (see [Section 3.1.13](#)) will occur as soon as the first text is encountered if the language of the text is one that the default voice cannot speak. This assumes that the voice has not been changed before encountering the text, of course.

```
<?xml version="1.0"?>
<speak version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
  ... the body ...
</speak>
```

The [speak](#) element can only contain text to be rendered and the following elements: [audio](#), [break](#), [emphasis](#), [lang](#), [lexicon](#), [lookup](#), [mark](#), [meta](#), [metadata](#), [p](#), [phoneme](#), [prosody](#), [say-as](#), [sub](#), [s](#), [token](#), [voice](#), [w](#).

3.1.1.1 Trimming Attributes

Trimming attributes define the span of the document to be rendered. Both the start and the end of the span within the [speak](#) content can be specified using marks.

The following trimming attributes are defined for [speak](#):

Name	Required	Type	Default Value	Description
startmark	false	TOKEN	none	The mark used to determined when rendering starts.
endmark	false	TOKEN	none	The mark used to determine when rendering ends.

The span of the document rendered is determined as follows:

- If the `startmark` is specified, then rendering starts at the `startmark`. If `startmark` is not specified, rendering begins at the beginning of the document.
- If the `endmark` is specified, then rendering ends at the `endmark`. If the `endmark` is not specified, rendering ends at the document end.
- If the `startmark` is after the `endmark`, then no audio is generated.

Issue: Error reporting has not yet been addressed. Neither have certain edge cases.

Examples

If no trimming attributes are specified, then the complete document is rendered:

```
<speak version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
  <audio src="first.wav"/>
  <markname="mark1"/>
  <audio src="middle.wav"/>
  <markname="mark2"/>
  <audio src="last.wav"/>
</speak>
```

here "first.wav", "middle.wav" and "last.wav" are rendered, where the mark "mark2" is the last mark rendered.

The `startmark` can be used to specify that rendering begins from a specific mark:

```
<speak startmark="mark1" version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
  <audio src="first.wav"/>
  <markname="mark1"/>
  <audio src="middle.wav"/>
  <markname="mark2"/>
  <audio src="last.wav"/>
</speak>
```

"middle.wav" and "last.wav" are rendered, but not "first.wav" since it occurs before the `startmark` "mark1".

The end of rendering can be specified using the `endmark`:

```
<speak endmark="mark2" version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
  <audio src="first.wav"/>
  <markname="mark1"/>
  <audio src="middle.wav"/>
  <markname="mark2"/>
  <audio src="last.wav"/>
</speak>
```

where "first.wav" and "middle.wav" are completely rendered but none of "last.wav" is rendered.

Finally, these trimming attributes can be used to control both the start and end of rendering:

```
<speak startmark="mark1" endmark="mark1"
  version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
  <audio src="first.wav"/>
  <markname="mark1"/>
  <audio src="middle.wav"/>
  <markname="mark2"/>
  <audio src="last.wav"/>
</speak>
```

where only "middle.wav" is played.

3.1.2 Language: `xml:lang` Attribute

The [`xml:lang` attribute](#), as defined by XML [[XML 1.0](#) or [XML 1.1](#), as appropriate, §2.12], *may* be used in SSML to indicate the natural language of the written content of the element on which it occurs. BCP47 [[BCP47](#)] can help in understanding how to use this attribute.

Language information is inherited down the document hierarchy, i.e. it needs to be given only once if the whole document is in one language, and language information nests, i.e. inner attributes overwrite outer attributes.

[`xml:lang`](#) is a defined attribute for the [speak](#), [lang](#), [desc](#), [p](#), [s](#), [token](#), and [w](#) elements.

[`xml:lang`](#) is permitted on [p](#), [s](#), [token](#), and [w](#) only because it is common to change the language at those levels.

The [synthesis processor](#) *should* use the value of the [`xml:lang`](#) attribute to assist it in determining the best way of rendering the content of the element on which it occurs. The [voice](#), [say-as](#), [phoneme](#), [sub](#), [emphasis](#), and [break](#) elements *should* also be rendered in a manner that is appropriate to the current language.

If the document author requires a new voice that is better adapted to the new language, then the [synthesis processor](#) can be explicitly requested to select a new voice by using the [voice](#) element. Further information about voice selection appears in [Section 3.2.1](#).

The [text normalization](#) processing step may be affected by the enclosing language. This is true for both markup support by the [say-as](#) element and non-markup behavior. In the following example the same text "2/1/2000" may be read as "February first two thousand" in the first sentence,

following American English pronunciation rules, but as "the second of January two thousand" in the second one, which follows Italian preprocessing rules.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xml:lang="en-US">
  <s>Today, 2/1/2000.</s>
  <!-- Today, February first two thousand -->
  <s xml:lang="it">Un mese fà, 2/1/2000.</s>
  <!-- Un mese fà, il due gennaio duemila -->
  <!-- One month ago, the second of January two thousand -->
</speak>
```

3.1.3 Base URI: [xml:base](#) Attribute

Relative [URIs](#) are resolved according to a *base URI*, which may come from a variety of sources. The base URI declaration allows authors to specify a document's base URI explicitly. See [Section 3.1.3.1](#) for details on the resolution of relative URIs.

The [base URI declaration](#) is permitted but *optional*. The two elements affected by it are

[audio](#)

The *optional* `src` attribute can specify a relative URI.

[lexicon](#)

The `uri` attribute can specify a relative URI.

The `xml:base` attribute

The base [URI](#) declaration follows [\[XML-BASE\]](#) and is indicated by an [xml:base](#) attribute on the root [speak](#) element.

```
<?xml version="1.0"?>
<speak version="1.1" xml:lang="en-US"
      xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xml:base="http://www.example.com/base-file-path">
```

```
<?xml version="1.0"?>
<speak version="1.1" xml:lang="en-US"
      xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xml:base="http://www.example.com/another-base-file-path">
```

3.1.3.1 Resolving Relative URIs

User agents *must* calculate the base [URI](#) for resolving relative URIs according to [\[RFC3986\]](#). The following describes how RFC3986 applies to synthesis documents.

User agents *must* calculate the base URI according to the following precedences (highest priority to lowest):

1. The base URI is set by the [xml:base](#) attribute on the [speak](#) element (see [Section 3.1.3](#)).
2. The base URI is given by metadata discovered during a protocol interaction, such as an HTTP header (see [\[RFC2616\]](#)).

3. By default, the base URI is that of the current document. Not all synthesis documents have a base URI (e.g., a valid synthesis document may appear in an email and may not be designated by a URI). It is an [error](#) if such documents contain relative URIs.

3.1.4 Identifier: [xml:id](#) Attribute

The [xml:id](#) attribute [XML-ID](#) may be used in SSML to give an element an identifier that is unique to the document, allowing the element to be referenced from other documents.

[xml:id](#) is a defined attribute for the [lexicon](#), [p](#), [s](#), [token](#), and [w](#) elements.

3.1.5 Lexicon Documents: [lexicon](#) and [lookup](#) Elements

An SSML document *may* reference one or more lexicon documents. A lexicon document is located by a [URI](#) with an *optional* [media type](#) and is assigned a name that is unique in the SSML document.

3.1.5.1 [lexicon](#) Element

Any number of [lexicon](#) elements *may* occur as immediate children of the [speak](#) element.

The [lexicon](#) element *must* have a `uri` attribute specifying a [URI](#) that identifies the location of the lexicon document.

The [lexicon](#) element *must* have an [xml:id](#) attribute that assigns a name to the lexicon document. The name *must* be unique to the current SSML document. The scope of this name is the current SSML document.

The [lexicon](#) element *may* have a `type` attribute that specifies the [media type](#) of the lexicon document. The default value of the `type` attribute is `application/pls+xml`, the media type associated with Pronunciation Lexicon Specification [PLS] documents as defined in [\[RFC4267\]](#).

The [lexicon](#) element is an empty element.

Details of the `type` attribute

Note: the description and table that follow use an imaginary vendor-specific lexicon type of `x-vnd.example.lexicon`. This is intended to represent whatever format is returned/available, as appropriate.

A lexicon resource indicated by a [URI](#) reference may be available in one or more [media types](#). The SSML author can specify the preferred media type via the `type` attribute. When the content represented by a URI is available in many data formats, a [synthesis processor](#) *may* use the preferred type to influence which of the multiple formats is used. For instance, on a server implementing HTTP content negotiation, the processor may use the type to order the preferences in the negotiation.

Upon delivery, the resource indicated by a URI reference may be considered in terms of two types. The *declared media type* is the alleged value for the resource and the actual media type is the true format of its content. The *actual type* should be the same as the declared type, but this is not always the case (e.g. a misconfigured HTTP server might return `text/plain` for a document following the vendor-specific `x-vnd.example.lexicon` format). A specific URI scheme may require that the resource owner always, sometimes, or never return a media type. Whenever a type is returned, it is treated as authoritative. The declared media type is determined by the value returned by the resource owner or, if none is returned, by the preferred media type given in the SSML document.

Three special cases may arise. The declared type may not be supported by the processor; this is an [error](#). The declared type may be supported but the actual type may not match; this is also an [error](#). Finally, no media type may be declared; the behavior depends on the specific URI scheme and the capabilities of the [synthesis processor](#). For instance, HTTP 1.1 allows document introspection (see [RFC2616](#) §7.2.1), the data scheme falls back to a default media type, and local file access defines no guidelines. The following table provides some informative examples:

Media type examples

	HTTP 1.1 request		Local file access	
Media type returned by the resource owner	text/plain	x-vnd.example.lexicon	<none>	<none>
Preferred media type from the SSML document	Not applicable; the returned type is authoritative.		x-vnd.example.lexicon	application/pls+xml
Declared media type	text/plain	x-vnd.example.lexicon	x-vnd.example.lexicon	<none>
Behavior for an actual media type of x-vnd.example.lexicon	This <i>must</i> be processed as text/plain. This will generate an error if text/plain is not supported or if the document does not follow the expected format.	The declared and actual types match; success if x-vnd.example.lexicon is supported by the synthesis processor; otherwise an error .		Scheme specific; the synthesis processor might introspect the document to determine the type.

3.1.5.2 lookup Element

The [lookup](#) element *must* have a `ref` attribute. The `ref` attribute specifies a name that references a lexicon document as assigned by the `xml:id` attribute of the [lexicon](#) element.

The referenced lexicon document may contain information (e.g., pronunciation) for tokens that can appear in a text to be rendered. For PLS lexicon documents, the information contained within the PLS document *must* be used by the [synthesis processor](#) when rendering tokens that appear within the context of a lookup element. For non-PLS lexicon documents, the information contained within the lexicon document *should* be used by the [synthesis processor](#) when rendering tokens that appear within the content of a lookup element, although the processor *may* choose not to use the information if it is deemed incompatible with the content of the SSML document. For example, a vendor-specific lexicon may be used only for particular values of the `interpret-as` attribute of the [say-as](#) element, or for a particular set of voices. Vendors *should* document the expected behavior of the [synthesis processor](#) when SSML content refers to a non-PLS lexicon.

A [lookup](#) element *may* contain other [lookup](#) elements. When a [lookup](#) element contains other [lookup](#) elements, the child [lookup](#) elements have higher precedence. Precedence means that a token is first looked up in the lexicon with highest precedence. Only if the token is not found in that lexicon

is it then looked up in the lexicon with the next lower precedence, and so on until the token is successfully found or until all lexicons have been used for lookup. It is assumed that the [synthesis processor](#) already has one or more built-in system lexicons which will be treated as having a lower precedence than those specified using the [lexicon](#) and [lookup](#) elements.

The [lookup](#) element can only contain text to be rendered and the following elements: [audio](#), [break](#), [emphasis](#), [lang](#), [lookup](#), [mark](#), [p](#), [phoneme](#), [prosody](#), [say-as](#), [sub](#), [s](#), [token](#), [voice](#), [w](#).

```
<?xml version="1.0"?>
<!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESES 1.0//EN"
    "http://www.w3.org/TR/speech-synthesis/synthesis.dtd">
<speak version="1.1"
    xmlns="http://www.w3.org/2001/10/synthesis"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
    xml:lang="en-US">

    <lexicon uri="http://www.example.com/lexicon.pls"
        xml:id="pls"/>
    <lexicon uri="http://www.example.com/strange-words.file"
        xml:id="sw"
        type="media-type"/>
    <lookup ref="pls">
        tokens here are looked up in lexicon.pls
    <lookup ref="sw">
        tokens here are looked up first in strange-words.file and then, if not found, in lexicon.pls
    </lookup>
        tokens here are looked up in lexicon.pls
    </lookup>
        tokens here are not looked up in lexicon documents
    ...
</speak>
```

3.1.6 meta Element

The [metadata](#) and [meta](#) elements are containers in which information about the document can be placed. The [metadata](#) element provides more general and powerful treatment of metadata information than [meta](#) by using a metadata schema.

A [meta](#) declaration associates a string to a declared meta property or declares "http-equiv" content. Either a name or http-equiv attribute is *required*. It is an [error](#) to provide both name and http-equiv attributes. A content attribute is *required*. The [seeAlso](#) property is the only defined [meta](#) property name. It is used to specify a resource that might provide additional metadata information about the content. This property is modeled on the [seeAlso](#) property of Resource Description Framework (RDF) Schema Specification 1.0 [RDF-SCHEMA §5.4.1]. The http-equiv attribute has a special significance when documents are retrieved via HTTP. Although the preferred method of providing HTTP header information is by using HTTP header fields, the "http-equiv" content *may* be used in situations where the SSML document author is unable to configure HTTP header fields associated with their document on the origin server, for example, cache control information. Note that HTTP servers and caches are not required to introspect the contents of [meta](#) in SSML documents and thereby override the header values they would send otherwise.

Informative: This is an example of how [meta](#) elements can be included in an SSML document to specify a resource that provides additional metadata information and also indicate that the document must not be cached.

```
<?xml version="1.0"?>
<!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESES 1.0//EN"
    "http://www.w3.org/TR/speech-synthesis/synthesis.dtd">
<speak version="1.1"
    xmlns="http://www.w3.org/2001/10/synthesis"
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
                    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
xml:lang="en-US">

<meta name="seeAlso" content="http://example.com/my-ssml-metadata.xml"/>
<meta http-equiv="Cache-Control" content="no-cache"/>

</speak>

```

The [meta](#) element is an empty element.

3.1.7 metadata Element

The [metadata](#) element is a container in which information about the document can be placed using a metadata schema. Although any metadata schema can be used with [metadata](#), it is *recommended* that the XML syntax of the Resource Description Framework (RDF) [[RDF-XMLSYNTAX](#)] be used in conjunction with the general metadata properties defined in the Dublin Core Metadata Initiative [[DC](#)].

The Resource Description Format [[RDF](#)] is a declarative language and provides a standard way for using XML to represent metadata in the form of statements about properties and relationships of items on the Web. Content creators should refer to W3C metadata Recommendations [[RDF-XMLSYNTAX](#)] and [[RDF-SCHEMA](#)] when deciding which metadata RDF schema to use in their documents. Content creators should also refer to the Dublin Core Metadata Initiative [[DC](#)], which is a set of generally applicable core metadata properties (e.g., Title, Creator, Subject, Description, Rights, etc.).

Document properties declared with the [metadata](#) element can use any metadata schema.

Informative: This is an example of how [metadata](#) can be included in an SSML document using the Dublin Core version 1.0 RDF schema [[DC](#)] describing general document information such as title, description, date, and so on:

```

<?xml version="1.0"?>
<!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESIS 1.0//EN"
                    "http://www.w3.org/TR/speech-synthesis/synthesis.dtd">
<speak version="1.1"
"
    xmlns="http://www.w3.org/2001/10/synthesis"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
                        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
    xml:lang="en-US">

    <metadata>
    <rdf:RDF
        xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
        xmlns:dc = "http://purl.org/dc/elements/1.1/">

        <!-- Metadata about the synthesis document -->
        <rdf:Description rdf:about="http://www.example.com/meta.ssml"
            dc:Title="Hamlet-like Soliloquy"
            dc:Description="Aldine's Soliloquy in the style of Hamlet"
            dc:Publisher="W3C"
            dc:Language="en-US"
            dc>Date="2002-11-29"
            dc:Rights="Copyright 2002 Aldine Turnbet"
            dc:Format="application/ssml+xml" >
            <dc:Creator>
                <rdf:Seq ID="CreatorsAlphabeticalBySurname">
                    <rdf:li>William Shakespeare</rdf:li>
                    <rdf:li>Aldine Turnbet</rdf:li>
                </rdf:Seq>
            </dc:Creator>
        </rdf:Description>
    </rdf:RDF>
    </metadata>

```

```
</speak>
```

The [metadata](#) element can have arbitrary content, although none of the content will be rendered by the [synthesis processor](#).

3.1.8 Text Structure: p, s, and w Elements

3.1.8.1 p and s Elements

A [p](#) element represents a paragraph. An [s](#) element represents a sentence.

[xml:lang](#) and [xml:id](#) are defined attributes on the [p](#) and [s](#) elements.

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xml:lang="en-US">
  <p>
    <s>This is the first sentence of the paragraph.</s>
    <s>Here's another sentence.</s>
  </p>
</speak>
```

The use of [p](#) and [s](#) elements is *optional*. Where text occurs without an enclosing [p](#) or [s](#) element the [synthesis processor](#) *should* attempt to determine the structure using language-specific knowledge of the format of plain text.

The [p](#) element can only contain text to be rendered and the following elements: [audio](#), [break](#), [emphasis](#), [lang](#), [mark](#), [phoneme](#), [prosody](#), [say-as](#), [sub](#), [s](#), [token](#), [voice](#), [w](#).

The [s](#) element can only contain text to be rendered and the following elements: [audio](#), [break](#), [emphasis](#), [lang](#), [mark](#), [phoneme](#), [prosody](#), [say-as](#), [sub](#), [token](#), [voice](#), [w](#).

3.1.8.2 token and w Elements

The [token](#) element allows the author to indicate its content is a token and to eliminate token (word) segmentation ambiguities of the synthesis processor.

The [token](#) element is necessary in order to render languages

- that do not use white-space as a token boundary identifier, such as Chinese, Thai, and Japanese
- that use white space for syllable segmentation, such as Vietnamese
- that use white space for other purposes, such as Urdu

Use of this element can result in improved cues for prosodic control (e.g., pause) and may assist the synthesis processor in selection of the correct pronunciation for homographs. Other elements such as [break](#), [mark](#), and [prosody](#) are permitted within [token](#) to allow annotation at a sub-token level (e.g., syllable, mora, or whatever units are reasonable for the current language). [Synthesis processors](#) are *required* to parse these annotations and *may* render them as they are able.

The use of [token](#) elements is *optional*. Where text occurs without an enclosing [token](#) element the [synthesis processor](#) *should* attempt to determine the token segmentation using language-specific knowledge of the format of plain text. Within the [token](#) element, all text (non-markup) content is significant; white space is preserved. Note that this is different from how text and markup outside a [token](#) element are treated (see "XML Parse" in [Section 1.2](#)). The entire text content of the [token](#)

element is considered to be one token for lexical lookup purposes. Thus, "<token><emphasis>hap</emphasis>py</token>" and "<token><emphasis> hap </emphasis> py</token>" would refer to the tokens "happy" and "hap py", respectively.

[xml:lang](#) is a defined attribute on the [token](#) element to identify the written language of the content.

[xml:id](#) is a defined attribute on the [token](#) element.

[role](#) is an *optional* defined attribute on the [token](#) element. The [role](#) attribute takes as its value one or more white-space separated [QNames](#) (as defined in Section 4 of Namespaces in XML (1.0 [XMLNS 1.0] or 1.1 [XMLNS 1.1], depending on the version of XML being used)). A [QName](#) in the attribute content is expanded into an [expanded-name](#) using the namespace declarations in scope for the containing [token](#) element. Thus, each QName provides a reference to a specific item in the designated namespace. In the second example below, the QName within the [role](#) attribute expands to the "VV0" item in the "http://www.example.com/claws7tags" namespace. This mechanism allows for referencing defined taxonomies of word classes, with the expectation that they are documented at the specified namespace URI.

The [role](#) attribute is intended to be of use in synchronizing with other specifications, for example to describe additional information to help the selection of the most appropriate pronunciation for the contained text inside an external lexicon (see the [lexicon](#) element).

The [token](#) element can only contain text to be rendered and the following elements: [audio](#), [break](#), [emphasis](#), [mark](#), [phoneme](#), [prosody](#), [say-as](#), [sub](#), [voice](#).

The [token](#) element can only be contained in the following elements: [audio](#), [emphasis](#), [lang](#), [lookup](#), [prosody](#), [speak](#), [p](#), [s](#), [voice](#).

The [w](#) element is an alias for the [token](#) element.

Here is an example showing the use of the [token](#) element.

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xml:lang="zh-CN">

  <!-- The Nanjing Changjiang River Bridge -->
  <w>南京市</w><w>长江大桥</w>
  <!-- The mayor of Nanjin city, Jiang Daqiao -->
  南京市市长<w>江大桥</w>
  <!-- Shanghai is a metropoli -->
  上海是个<w>大都会</w>
  <!-- Most Shanghainese will say something like that -->
  上海人<w>大都</w>会那么说
</speak>
```

The next example shows the use of the [role](#) attribute. The first document below is a sample lexicon (PLS) for the Chinese word "处". The second references this lexicon and shows how the role attribute may be used to select the appropriate pronunciation of the Chinese word "处" in the dialog.

```
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
  xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
  xmlns:claws="http://www.example.com/claws7tags"
  alphabet="x-myorganization-pinyin"
  xml:lang="zh-CN">
  <lexeme role="claws:VV0">
    <!-- base form of lexical verb -->
```

```

    <grapheme>处</grapheme>
    <phoneme>chu3</phoneme>
    <!-- pinyin string is: "chǔ" in 处罚 处置 -->
</lexeme>
<lexeme role="claws:NN">
    <!-- common noun, neutral for number -->
    <grapheme>处</grapheme>
    <phoneme>chu4</phoneme>
    <!-- pinyin string is: "chù" in 处所 妙处 -->
</lexeme>
</lexicon>

```

```

<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
                        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xmlns:claws="http://www.example.com/claws7tags"
      xml:lang="zh-CN">
  <lexicon uri="http://www.example.com/lexicon.pls"
    type="application/pls+xml"
    xml:id="mylex"/>
  <lookup ref="mylex">
    他这个人很不好相<w role="claws:VW0">处</w>。
    此<w role="claws:NN">处</w>不准照相。
  </lookup>
</speak>

```

3.1.9 say-as Element

The [say-as](#) element allows the author to indicate information on the type of text construct contained within the element and to help specify the level of detail for rendering the contained text.

Defining a comprehensive set of text format types is difficult because of the variety of languages that have to be considered and because of the innate flexibility of written languages. SSML only specifies the [say-as](#) element, its attributes, and their purpose. It does not enumerate the possible values for the attributes. The Working Group expects to produce a separate document that will define standard values and associated normative behavior for these values. Examples given here are only for illustrating the purpose of the element and the attributes.

The [say-as](#) element has three attributes: `interpret-as`, `format`, and `detail`. The `interpret-as` attribute is always *required*; the other two attributes are *optional*. The legal values for the `format` attribute depend on the value of the `interpret-as` attribute.

The [say-as](#) element can only contain text to be rendered.

The `interpret-as` and `format` attributes

The `interpret-as` attribute indicates the content type of the contained text construct. Specifying the content type helps the [synthesis processor](#) to distinguish and interpret text constructs that may be rendered in different ways depending on what type of information is intended. In addition, the *optional* `format` attribute can give further hints on the precise formatting of the contained text for content types that may have ambiguous formats.

When specified, the `interpret-as` and `format` values are to be interpreted by the [synthesis processor](#) as hints provided by the markup document author to aid [text normalization](#) and pronunciation.

In all cases, the text enclosed by any [say-as](#) element is intended to be a standard, orthographic form of the language currently in context. A [synthesis processor](#) *should* be able to support the common, orthographic forms of the specified language for every content type that it supports.

When the value for the `interpret-as` attribute is unknown or unsupported by a processor, it *must* render the contained text as if no `interpret-as` value were specified.

When the value for the `format` attribute is unknown or unsupported by a processor, it *must* render the contained text as if no `format` value were specified, and *should* render it using the `interpret-as` value that is specified.

When the content of the [say-as](#) element contains additional text next to the content that is in the indicated `format` and `interpret-as` type, then this additional text *must* be rendered. The processor *may* make the rendering of the additional text dependent on the `interpret-as` type of the element in which it appears.

When the content of the [say-as](#) element contains no content in the indicated `interpret-as` type or `format`, the processor *must* render the content either as if the `format` attribute were not present, or as if the `interpret-as` attribute were not present, or as if neither the `format` nor `interpret-as` attributes were present. The processor *should* also notify the environment of the mismatch.

Indicating the content type or format does not necessarily affect the way the information is pronounced. A [synthesis processor](#) *should* pronounce the contained text in a manner in which such content is normally produced for the language.

The `detail` attribute

The `detail` attribute is an *optional* attribute that indicates the level of detail to be read aloud or rendered. Every value of the `detail` attribute *must* render all of the informational content in the contained text; however, specific values for the `detail` attribute can be used to render content that is not usually informational in running text but may be important to render for specific purposes. For example, a [synthesis processor](#) will usually render punctuation through appropriate changes in prosody. Setting a higher level of detail may be used to speak punctuation explicitly, e.g. for reading out coded part numbers or pieces of software code.

The `detail` attribute can be used for all `interpret-as` types.

If the `detail` attribute is not specified, the level of detail that is produced by the [synthesis processor](#) depends on the text content and the language.

When the value for the `detail` attribute is unknown or unsupported by a processor, it *must* render the contained text as if no value were specified for the `detail` attribute.

3.1.10 `phoneme` Element

The [phoneme](#) element provides a phonemic/phonetic pronunciation for the contained text. The [phoneme](#) element *may* be empty. However, it is *recommended* that the element contain human-readable text that can be used for non-spoken rendering of the document. For example, the content may be displayed visually for users with hearing impairments.

The `ph` attribute is a *required* attribute that specifies the phoneme/phone string.

This element is designed strictly for phonemic and phonetic notations and is intended to be used to provide pronunciations for words or very short phrases. The phonemic/phonetic string does not undergo [text normalization](#) and is not treated as a token for lookup in the lexicon (see [Section 3.1.5](#)), while values in [say-as](#) and [sub](#) may undergo both. Briefly, phonemic strings consist of phonemes, language-dependent speech units that characterize linguistically significant differences in the language; loosely, phonemes represent all the sounds needed to distinguish

one word from another in a given language. On the other hand, phonetic strings consist of phones, speech units that characterize the manner (puff of air, click, vocalized, etc.) and place (front, middle, back, etc.) of articulation within the human vocal tract and are thus independent of language; phones represent realized distinctions in human speech production.

The `a1phabet` attribute is an *optional* attribute that specifies the phonemic/phonetic alphabet. An alphabet in this context refers to a collection of symbols to represent the sounds of one or more human languages. The only valid values for this attribute are "**ipa**" (see the next paragraph), values defined in Pronunciation Alphabet Registry and vendor-defined strings of the form "**x-organization**" or "**x-organization-alphabet**". For example, the Japan Electronics and Information Technology Industries Association [[JEITA](#)] might wish to encourage the use of an alphabet such as "x-JEITA" or "x-JEITA-2000" for their phoneme alphabet [[JEIDAALPHABET](#)].

[Synthesis processors](#) *should* support a value for `a1phabet` of "**ipa**", corresponding to Unicode representations of the phonetic characters developed by the International Phonetic Association [[IPA](#)]. In addition to an exhaustive set of vowel and consonant symbols, this character set supports a syllable delimiter, numerous diacritics, stress symbols, lexical tone symbols, intonational markers and more. For this alphabet, legal `ph` values are strings of the values specified in Appendix 2 of [[IPAHNDBK](#)]. Informative tables of the IPA-to-Unicode mappings can be found at [[IPAUNICODE1](#)] and [[IPAUNICODE2](#)]. Note that not all of the IPA characters are available in Unicode. For processors supporting this alphabet,

- The processor *must* syntactically accept all legal `ph` values.
- The processor *should* produce output when given Unicode IPA codes that can reasonably be considered to belong to the current language.
- The production of output when given other codes is entirely at processor discretion.

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
  <phoneme alphabet="ipa" ph="t&#x259;mei&#x325;&#x27E;ou&#x325;"> tomato </phoneme>
  <!-- This is an example of IPA using character entities -->
  <!-- Because many platform/browser/text editor combinations do not
    correctly cut and paste Unicode text, this example uses the entity
    escape versions of the IPA characters. Normally, one would directly
    use the UTF-8 representation of these symbols: "təmeɪrou". -->
</speak>
```

It is an [error](#) if a value for `a1phabet` is specified that is not known or cannot be applied by a [synthesis processor](#). The default behavior when the `a1phabet` attribute is left unspecified is processor-specific.

The [phoneme](#) element itself can only contain text (no elements).

3.1.10.1 Pronunciation Alphabet Registry

The Pronunciation Alphabet Registry will be maintained by W3C.

Issue: We are still working out the location and details of the Registry. A link will be provided in this document when it is available.

Issue: The LTRU IETF WG (which is working on language tags) is currently discussing the introduction of a subtag for IPA, and maybe other alphabets. We are coordinating with them to determine what overlap, if any, there is between our two efforts.

3.1.11 sub Element

The [sub](#) element is employed to indicate that the text in the `alias` attribute value replaces the contained text for pronunciation. This allows a document to contain both a spoken and written form. The *required* `alias` attribute specifies the string to be spoken instead of the enclosed string. The processor *should* apply [text normalization](#) to the `alias` value.

The [sub](#) element can only contain text (no elements).

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xml:lang="en-US">
  <sub alias="World Wide Web Consortium">W3C</sub>
  <!-- World Wide Web Consortium -->
</speak>
```

3.1.12 lang Element

The [lang](#) element is used to specify the natural language of the content.

The [lang](#) element has one attribute, `xml:lang`, which is always *required*.

This element *may* be used when there is a change in the natural language. There is no text structure associated with the language change indicated by the [lang](#) element. It *may* be used to specify the language of the content at a level other than a paragraph, sentence or word level. When language change is to be associated with text structure, it is *recommended* to use the `xml:lang` attribute on the respective [p](#), [s](#), [token](#), or [w](#) element.

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xml:lang="en-US">
  The French word for cat is <w xml:lang="fr">chat</w>.
  He prefers to eat pasta that is <lang xml:lang="it">al dente</lang>.
</speak>
```

The [lang](#) element can only contain text to be rendered and the following elements: [audio](#), [break](#), [emphasis](#), [lang](#), [mark](#), [p](#), [phoneme](#), [prosody](#), [say-as](#), [sub](#), [s](#), [token](#), [voice](#), [w](#).

3.1.13 Language Speaking Failure: onlangfailure Attribute

The `onlangfailure` attribute is an *optional* attribute that contains one value from the following enumerated list describing the desired behavior of the [synthesis processor](#) upon language speaking failure. A conforming [synthesis processor](#) *must* report a language speaking failure in addition to taking the action(s) below.

- **changevoice** - if a voice exists that can speak the language, the [synthesis processor](#) will switch to that voice and speak the content. Otherwise, the processor chooses another behavior (either `ignoretext` or `ignorelang`).
- **ignoretext** - the [synthesis processor](#) will not attempt to render the text that is in the failed language.
- **ignorelang** - the [synthesis processor](#) will ignore the change in language and speak as if the content were in the previous language.
- **processorchoice** - the [synthesis processor](#) chooses the behavior (either `changevoice`, `ignoretext`, or `ignorelang`).

A language speaking failure occurs whenever the [synthesis processor](#) decides that the currently-selected voice (see [Section 3.2.1](#)) cannot speak the declared language of the text.

The value of this attribute is inherited down the document hierarchy, i.e. it needs to be given only once if the desired behavior for the whole document is the same, and settings of this value nest, i.e. inner attributes overwrite outer attributes. The top-level default value for this attribute is "processorchoice". Other languages which embed fragments of SSML (without a [speak](#) element) *must* declare the top-level default value for this attribute.

[onlangfailure](#) is permitted on all elements which can contain [xml:lang](#), so it is a defined attribute for the [speak](#), [lang](#), [desc](#), [p](#), [s](#), [token](#), and [w](#) elements.

3.2 Prosody and Style

3.2.1 voice Element

The [voice](#) element is a production element that requests a change in speaking voice. There are two kinds of attributes for the voice element: those that indicate desired features of a voice and those that control behavior. The voice feature attributes are:

- **gender**: *optional* attribute indicating the preferred gender of the voice to speak the contained text. Enumerated values are: "male", "female", "neutral", "".
- **age**: *optional* attribute indicating the preferred age in years (since birth) of the voice to speak the contained text. Acceptable values are of type [xsd:nonNegativeInteger](#) [SCHEMA2 §3.3.20] or the empty string "".
- **variant**: *optional* attribute indicating a preferred variant of the other voice characteristics to speak the contained text. (e.g. the second male child voice). Valid values of **variant** are of type [xsd:positiveInteger](#) [SCHEMA2 §3.3.25] or the empty string "".
- **name**: *optional* attribute indicating a processor-specific voice name to speak the contained text. The value *may* be a space-separated list of names ordered from top preference down or the empty string "". As a result a name *must not* contain any white space.
- **languages**: *optional* attribute indicating the list of languages the voice can speak. The value *must* be either the empty string "" or a space-separated list of languages, with *optional* accent indication per language. Each language/accent pair is of the form "*language*" or "*language:accent*", where both *language* and *accent* *must* be an Extended Language Range [RFC4647 §2.2], except that the values "und" and "zxx" are disallowed. A voice satisfies the **languages** feature if, for each language/accent pair in the list,
 1. the voice can read/speak a language whose description as a language tag matches the Extended Language Range given by *language* according to the Extended Filtering matching algorithm [RFC4647 §3.3.2], and
 2. if an *accent* is given, the voice can read/speak the language above with an accent whose description as a language tag matches the Extended Language Range given by *accent* according to the Extended Filtering matching algorithm [RFC4647 §3.3.2], except that the script and extension subtags of the *accent* *may* be ignored by the [synthesis processor](#).

For example, a **languages** value of "en:zh fr:de" can legally be matched by any voice that can both read English (speaking it with a Chinese accent) and read French (speaking it with a German accent). Thus, a voice that only supports "en-US" with a "zh-CN-HK" accent and "fr-CA" with a "de-SU" accent would match. As another example, if we have <voice languages="fr:zh"> and there is no voice that supports French with a Chinese accent, then a voice selection failure will occur. Note that if no accent indication is given for a language, then any voice that speaks the language is acceptable, regardless of accent. Also, note that author control over language support during voice selection is now independent of any value of [xml:lang](#) in the text.

For the feature attributes above, an empty-string value indicates that any voice will satisfy the feature. The top-level default value for all feature attributes is "", the empty string.

The behavior control attributes of [voice](#) are:

- **required:** *optional* attribute that specifies which feature attributes must be matched for voice selection to be a success. Valid values of **required** are a space-separated list composed of values from the list of feature names: **"name"**, **"languages"**, **"gender"**, **"age"**, **"variant"** or the empty string "". The default value for this attribute is "languages".
- **ordering:** *optional* attribute that specifies the matching priority ordering for voice selection. Valid values of **ordering** are a space-separated list composed of values from the list of feature names: **"name"**, **"languages"**, **"gender"**, **"age"**, **"variant"** or the empty string "", where features named earlier in the list have higher priority. The default value for this attribute is "languages". When multiple voices are available that satisfy the **required** criteria, voices matching higher-priority attributes *must* be selected over voices matching lower-priority attributes. Features not listed in the **ordering** list will be considered to have equal priority to each other but lower than that of the last feature in the list. Note that if the **ordering** attribute has been set to an empty value then all features have the same priority.
- **onvoicefailure:** *optional* attribute containing one value from the following enumerated list describing the desired behavior of the [synthesis processor](#) upon voice selection failure. A conforming [synthesis processor](#) *must* report a voice selection failure in addition to taking the action(s) below. The default value for this attribute is "priorityselect".
 - **priorityselect** - the [synthesis processor](#) selects a voice by matching the highest priority feature first (in the **ordering** list), then the second, etc.
 - **keepexisting** - the voice does not change.
 - **processorchoice** - the [synthesis processor](#) chooses the behavior (either **priorityselect** or **keepexisting**).

Issue: the voice selection algorithm is still not described as precisely as it should be. The general idea is that if multiple voices match, then the one that matches the highest-priority attributes wins. Although we know what behavior we want, it still needs to be written out carefully.

Although each attribute individually is optional, it is an [error](#) if no attributes are specified when the [voice](#) element is used.

```
<?xml version="1.0"?>
<speech version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
  <voice gender="female" languages="en-US" required="languages gender variant">Mary had a little lamb,</voice>
  <!-- now request a different female child's voice -->
  <voice gender="female" variant="2">
    Its fleece was white as snow.
  </voice>
  <!-- processor-specific voice selection -->
  <voice name="Mike" required="name">I want to be like Mike.</voice>
</speech>
```

Although indication of language (using [xml:lang](#)) and selection of voice (using [voice](#)) are independent, there is no requirement that a synthesis processor support every possible combination of values of the two. However, a synthesis processor *must* document expected rendering behavior for every possible combination.

Voice selection is considered successful if and only if the values of all voice feature attributes listed in the **required** attribute value are matched. Otherwise, it's considered a failure. A **required** attribute with a value of the empty string "" is always considered to be a full match. If more than one voice can match all of the required voice feature attributes, the [synthesis processor](#) *must* choose the voice that best matches the remaining features.

[voice](#) attributes are inherited down the tree including to within elements that change the language. The defaults described for each attribute only apply at the top (document) level and are overridden by explicit author use of the [voice](#) element. In addition, changes in voice are scoped and apply only to the content of the element in which the change occurred. When processing reaches the end of a [voice](#) element content, i.e. the closing `</voice>` tag, the voice in effect before the beginning tag is restored. Similarly, if a voice is changed by the processor as a result of a language speaking failure, when the element in which the new [xml:lang](#) value occurs completes, the voice in effect before the element is restored.

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
  <voice gender="female" required="languages gender age" languages="en-US ja">
    Any female voice here.
    <voice age="6">
      A female child voice here.
      <lang xml:lang="ja">
        <!-- Same female child voice rendering Japanese text. -->
      </lang>
    </voice>
  </voice>
</speak>
```

Relative changes in prosodic parameters *should* be carried across voice changes. However, different voices have different natural defaults for pitch, speaking rate, etc. because they represent different personalities, so absolute values of the prosodic parameters may vary across changes in the voice.

The quality of the output audio or voice may suffer if a change in voice is requested within a sentence.

The [voice](#) element can only contain text to be rendered and the following elements: [audio](#), [break](#), [emphasis](#), [lang](#), [mark](#), [p](#), [phoneme](#), [prosody](#), [say-as](#), [sub](#), [s](#), [token](#), [voice](#), [w](#).

3.2.2 [emphasis](#) Element

The [emphasis](#) element requests that the contained text be spoken with emphasis (also referred to as prominence or stress). The [synthesis processor](#) determines how to render emphasis since the nature of emphasis differs between languages, dialects or even voices. The attributes are:

- **level**: the *optional* **level** attribute indicates the strength of emphasis to be applied. Defined values are **"strong"**, **"moderate"**, **"none"** and **"reduced"**. The default **level** is **"moderate"**. The meaning of **"strong"** and **"moderate"** emphasis is interpreted according to the language being spoken (languages indicate emphasis using a possible combination of pitch change, timing changes, loudness and other acoustic differences). The **"reduced"** **level** is effectively the opposite of emphasizing a word. For example, when the phrase "going to" is reduced it may be spoken as "gonna". The **"none"** **level** is used to prevent the [synthesis processor](#) from emphasizing words that it might typically emphasize. The values **"none"**, **"moderate"**, and **"strong"** are monotonically non-decreasing in strength.

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
  That is a <emphasis> big </emphasis> car!
  That is a <emphasis level="strong"> huge </emphasis>
  bank account!
</speak>
```

The [emphasis](#) element can only contain text to be rendered and the following elements: [audio](#), [break](#), [emphasis](#), [lang](#), [mark](#), [phoneme](#), [prosody](#), [say-as](#), [sub](#), [token](#), [voice](#), [w](#).

3.2.3 break Element

The [break](#) element is an empty element that controls the pausing or other prosodic boundaries between tokens. The use of the [break](#) element between any pair of tokens is *optional*. If the element is not present between tokens, the [synthesis processor](#) is expected to automatically determine a break based on the linguistic context. In practice, the [break](#) element is most often used to override the typical automatic behavior of a synthesis processor. The attributes on this element are:

- **strength**: the **strength** attribute is an *optional* attribute having one of the following values: **"none"**, **"x-weak"**, **"weak"**, **"medium"** (default value), **"strong"**, or **"x-strong"**. This attribute is used to indicate the strength of the prosodic break in the speech output. The value **"none"** indicates that no prosodic break boundary should be outputted, which can be used to prevent a prosodic break which the processor would otherwise produce. The other values indicate monotonically non-decreasing (conceptually increasing) break strength between tokens. The stronger boundaries are typically accompanied by pauses. **"x-weak"** and **"x-strong"** are mnemonics for "extra weak" and "extra strong", respectively.
- **time**: the **time** attribute is an *optional* attribute indicating the duration of a pause to be inserted in the output in seconds or milliseconds. It follows the time value format from the Cascading Style Sheets Level 2 Recommendation [[CSS2](#)], e.g. "250ms", "3s".

The **strength** attribute is used to indicate the prosodic strength of the break. For example, the breaks between paragraphs are typically much stronger than the breaks between words within a sentence. The [synthesis processor](#) may insert a pause as part of its implementation of the prosodic break. A pause of a specific length can also be inserted by using the **time** attribute.

If a [break](#) element is used with neither **strength** nor **time** attributes, a break will be produced by the processor with a prosodic strength greater than that which the processor would otherwise have used if no [break](#) element was supplied.

If both **strength** and **time** attributes are supplied, the processor will insert a break with a duration as specified by the **time** attribute, with other prosodic changes in the output based on the value of the **strength** attribute.

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
      http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xml:lang="en-US">
  Take a deep breath <break/>
  then continue.
  Press 1 or wait for the tone. <break time="3s"/>
  I didn't hear you! <break strength="weak"/> Please repeat.
</speak>
```

3.2.4 prosody Element

The [prosody](#) element permits control of the pitch, speaking rate and volume of the speech output. The attributes, all *optional*, are:

- **pitch**: the baseline pitch for the contained text. Although the exact meaning of "baseline pitch" will vary across synthesis processors, increasing/decreasing this value will typically increase/decrease the approximate pitch of the output. Legal values are: a [number](#) followed by "Hz", a [relative change](#) or **"x-low"**, **"low"**, **"medium"**, **"high"**, **"x-high"**, or **"default"**.

Labels **"x-low"** through **"x-high"** represent a sequence of monotonically non-decreasing pitch levels.

- **contour**: sets the actual pitch contour for the contained text. The format is specified in [Pitch contour](#) below.
- **range**: the pitch range (variability) for the contained text. Although the exact meaning of "pitch range" will vary across synthesis processors, increasing/decreasing this value will typically increase/decrease the dynamic range of the output pitch. Legal values are: a [number](#) followed by "Hz", a [relative change](#) or **"x-low"**, **"low"**, **"medium"**, **"high"**, **"x-high"**, or **"default"**. Labels **"x-low"** through **"x-high"** represent a sequence of monotonically non-decreasing pitch ranges.
- **rate**: a change in the speaking rate for the contained text. Legal values are: a [relative change](#) or **"x-slow"**, **"slow"**, **"medium"**, **"fast"**, **"x-fast"**, or **"default"**. Labels **"x-slow"** through **"x-fast"** represent a sequence of monotonically non-decreasing speaking rates. When a [number](#) is used to specify a [relative change](#) it acts as a multiplier of the default rate. For example, a value of 1 means no change in speaking rate, a value of 2 means a speaking rate twice the default rate, and a value of 0.5 means a speaking rate of half the default rate. The default rate for a voice depends on the language and dialect and on the personality of the voice. The default rate for a voice *should* be such that it is experienced as a normal speaking rate for the voice when reading aloud text. Since voices are processor-specific, the default rate will be as well.
- **duration**: a value in seconds or milliseconds for the desired time to take to read the element contents. Follows the time value format from the Cascading Style Sheet Level 2 Recommendation [[CSS2](#)], e.g. "250ms", "3s".
- **volume**: the volume for the contained text in the range 0.0 to 100.0 (higher values are louder and specifying a value of zero is equivalent to specifying **"silent"**). Legal values are: [number](#), a [relative change](#) or **"silent"**, **"x-soft"**, **"soft"**, **"medium"**, **"loud"**, **"x-loud"**, or **"default"**. The volume scale is linear amplitude. The default is 100.0. Labels **"silent"** through **"x-loud"** represent a sequence of monotonically non-decreasing volume levels.

Although each attribute individually is optional, it is an [error](#) if no attributes are specified when the [prosody](#) element is used. The **"x-foo"** attribute value names are intended to be mnemonics for "extra foo". All units ("Hz", "st") are case-sensitive. Note also that customary pitch levels and standard pitch ranges may vary significantly by language, as may the meanings of the labelled values for pitch targets and ranges.

Number

A number is a simple positive floating point value without exponentials. Legal formats are "n", "n.", ".n" and "n.n" where "n" is a sequence of one or more digits.

Relative values

Relative changes for the attributes above can be specified

- as a percentage (a [number](#) optionally preceded by "+" or "-" and followed by "%"), e.g. "3%", "+15.2%", "-8.0%", or
- as a relative number:
 - For the **rate** attribute, relative changes are a [number](#).
 - For the **volume** attribute, relative changes are a [number](#) preceded by "+" or "-", e.g. "+10", "-5.5".
 - For the **pitch** and **range** attributes, relative changes can be given in semitones (a [number](#) preceded by "+" or "-" and followed by "st") or in Hertz (a [number](#) preceded by

"+" or "-" and followed by "Hz"): "+0.5st", "+5st", "-2st", "+10Hz", "-5.5Hz". A semitone is half of a tone (a half step) on the standard diatonic scale.

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xml:lang="en-US">
  The price of XYZ is <prosody rate="-10%">$45</prosody>
</speak>
```

Pitch contour

The pitch contour is defined as a set of white space-separated targets at specified time positions in the speech output. The algorithm for interpolating between the targets is processor-specific. In each pair of the form (time position,target), the first value is a percentage of the period of the contained text (a [number](#) followed by "%") and the second value is the value of the pitch attribute (a [number](#) followed by "Hz", a [relative change](#), or a label value). Time position values outside 0% to 100% are ignored. If a pitch value is not defined for 0% or 100% then the nearest pitch target is copied. All relative values for the pitch are relative to the pitch value just before the contained text.

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xml:lang="en-US">
  <prosody contour="(0%,+20Hz) (10%,+30%) (40%,+10Hz)">
    good morning
  </prosody>
</speak>
```

The **duration** attribute takes precedence over the **rate** attribute. The **contour** attribute takes precedence over the **pitch** and **range** attributes.

The default value of all prosodic attributes is no change. For example, omitting the **rate** attribute means that the rate is the same within the element as outside.

The [prosody](#) element can only contain text to be rendered and the following elements: [audio](#), [break](#), [emphasis](#), [lang](#), [mark](#), [p](#), [phoneme](#), [prosody](#), [say-as](#), [sub](#), [s](#), [token](#), [voice](#), [w](#).

Limitations

All prosodic attribute values are indicative. If a [synthesis processor](#) is unable to accurately render a document as specified (e.g., trying to set the pitch to 1 MHz or the speaking rate to 1,000,000 words per minute), it *must* make a best effort to continue processing by imposing a limit or a substitute for the specified, unsupported value and *may* inform the host environment when such limits are exceeded.

In some cases, [synthesis processors](#) *may* elect to ignore a given prosodic markup if the processor determines, for example, that the indicated value is redundant, improper or in error. In particular, concatenative-type synthetic speech systems that employ large acoustic units *may* reject prosody-modifying markup elements if they are redundant with the prosody of a given acoustic unit(s) or would otherwise result in degraded speech quality.

3.3 Other Elements

3.3.1 audio Element

The [audio](#) element supports the insertion of recorded audio files (see [Appendix A](#) for *required* formats) and the insertion of other audio formats in conjunction with synthesized speech output. The [audio](#) element *may* be empty. If the [audio](#) element is not empty then the contents should be the marked-up text to be spoken if the audio document is not available. The alternate content *may* include text, speech markup, [desc](#) elements, or other [audio](#) elements. The alternate content *may* also be used when rendering the document to non-audible output and for accessibility (see the [desc](#) element). [audio](#) has one *required* attribute, `src`, which is the [URI](#) of a document with an appropriate MIME type, and several *optional* attributes, described in subsections below.

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">

  <!-- Empty element -->
  Please say your name after the tone. <audio src="beep.wav"/>

  <!-- Container element with alternative text -->
  <audio src="prompt.au">What city do you want to fly from?</audio>
  <audio src="welcome.wav">
    <emphasis>Welcome</emphasis> to the Voice Portal.
  </audio>

</speak>
```

An [audio](#) element is successfully rendered by:

1. Playing the referenced audio source successfully
2. If the referenced audio source fails to play, rendering the alternative content
3. Additionally if the processor can detect that text-only output is required then it *may* render the alternative content

When attempting to play the audio source a number of different issues may arise such as mismatched MIME types or bad header information about the media. In general the [synthesis processor](#) makes a best effort to play the referenced media and, when unsuccessful, the processor *must* play the alternative content. Note the processor *must not* render both all or part of the referenced media and all or part of the referenced alternative content. If any of the referenced media is processed and rendered then the playback is considered a successful playback within the context of this section. If an error occurs that causes the alternative content to be rendered instead of the referenced media the processor *must* notify the hosting environment that such an error has occurred. The processor *may* notify the hosting environment immediately with an asynchronous event, or the processor *may* notify the hosting environment only at the end of playback when it signals to the hosting environment that it has completed rendering the request, or the processor *may* make the error notification through its logging system. The processor *should* include information about the error where possible; for example, if the media resource couldn't be fetched due to an http 404 error, that error code could be included with the notification).

The [audio](#) element can only contain text to be rendered and the following elements: [audio](#), [break](#), [desc](#), [emphasis](#), [lang](#), [mark](#), [p](#), [phoneme](#), [prosody](#), [say-as](#), [sub](#), [s](#), [token](#), [voice](#), [w](#).

3.3.1.1 Trimming attributes

Trimming attributes define the span of the audio to be rendered. Both the start and the end of the span within the [audio](#) content can be specified using time offsets.

The following trimming attributes are defined for [audio](#):

Name	Required	Type	Default Value	Description
clipBegin	false	subset of SMIL 2.1 Timing Module "clipBegin" properties. Must be non-negative.	0s	offset from start of media to begin rendering. This offset is measured in normal media playback time from the beginning of the media.
clipEnd	false	subset of SMIL 2.1 Timing Module "clipEnd" properties. Must be non-negative.	None	offset from start of media to end rendering. This offset is measured in normal media playback time from the beginning of the media.
repeatCount	false	SMIL 2.1 "repeatCount": numeric value greater than zero, or 'indefinite'. Must be non-negative.	1	number of iterations of media to render
repeatDur	false	SMIL 2.1 "repeatDur": (Clock)Time Designation or "indefinite". Must be non-negative.	None	total duration for repeatedly rendering media

Calculations of rendered durations and interaction with other timing properties follow SMIL 2.1 [Computing the active duration](#). In addition,

- If **clipEnd** is after the end of the audio, then rendering ends at the audio end.
- If **clipBegin** is after **clipEnd**, no audio will be produced.

Note that not all SMIL 2.1 Timing features are supported. Also, since the length of an audio clip may not be known in advance, for example if it is streamed, large values of **clipEnd** may result in significant latency before playback begins.

Examples

In the following example, rendering of the media begins 10 seconds into the audio:

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">

  <audio src="radio.wav" clipBegin="10s" />

</speak>
```

Here the rendering of the media ends after 20 seconds of audio:

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">

  <audio src="radio.wav" clipBegin="10s" clipEnd="20s" />

</speak>
```

Note that if the duration of "radio.wav" is less than 20 seconds, the **clipEnd** value is ignored, and the rendering end is set equal to the effective end of the media.

In the following example, the duration of the audio is constrained by `repeatCount`:

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">

  <audio src="3second_sound.au" repeatCount="0.5" />

</speak>
```

Only the first half of the clip will play; the active duration will be 1.5 seconds.

In the following example, the audio will repeat for a total of 7 seconds. It will play fully two times, followed by a fractional part of 2 seconds. This is equivalent to a `repeatCount` of 2.8.

```
<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">

  <audio src="2.5second_music.mp3" repeatDur="7s" />

</speak>
```

These attributes can interact with the rendering specified by [speak](#) trimming attributes:

```
<speak version="1.1" startmark="mark1" endmark="mark2"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
  <audio src="first.wav"/>
  <markname="mark1"/>
  <audio src="15second_music.mp3" clipBegin="2s" clipEnd="7s" />
  <markname="mark2"/>
  <audio src="last.wav"/>
</speak>
```

The [speak](#) `startmark` and `endmark` allow only the "15second_music.mp3" clip to be played. The actual duration of the audio is 5 seconds: the clip begins at 2 seconds into the audio and ends after 7 seconds, hence a duration of 5 seconds.

3.3.1.2 gain Attribute

The `gain` attribute controls the amplitude of the referenced audio.

Name	Required	Type	Default Value	Description
gain	false	x x% +/-x% (where x is a non-negative real value)	The default value is 1, which corresponds to the amplitude of an unmodified audio waveform.	The amplitude at which to play the referenced audio, relative to the original amplitude. If the value is of the form "+/-x%", the amplitude is set to the amplitude of the original waveform plus/minus the given percentage. If the value is of the form "x%", the amplitude is set to that percentage of the amplitude of the original

				waveform. If the value is of the form "x", the amplitude is set to x times the amplitude of the original waveform.
--	--	--	--	---

Here is an example of how to use the `gain` attribute:

```
<speak version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">

  <s>This is the original, unmodified waveform:
    <audio src="message.wav"/>
  </s>
  <s>This is the same audio at twice the amplitude:
    <audio gain="2" src="message.wav"/>
  </s>
  <s>This is the same audio at half the original amplitude:
    <audio gain="-50%" src="message.wav"/>
  </s>
  <s>This is the same audio, also at half the original amplitude:
    <audio gain="50%" src="message.wav"/>
  </s>
</speak>
```

3.3.1.3 speed Attribute

The `speed` attribute specifies the rate at which the referenced audio is played, relative to the original speed of the waveform.

Name	Required	Type	Default Value	Description
speed	false	x x% +/-x% (where x is a non-negative real value)	The default value is 1, which corresponds to the speed of an unmodified audio waveform.	The speed at which to play the referenced audio, relative to the original speed. If the value is of the form "+/-x%", the speed is set to the speed of the original waveform plus/minus the given percentage. If the value is of the form "x%", the speed is set to that percentage of the speed of the original waveform. If the value is of the form "x", the speed is set to x times the speed of the original waveform.

Issue note: Is there value to authors in allowing/requiring the synthesis processor to adjust the speed without changing the effective pitch of the audio?

Here is an example of how to use the `speed` attribute:

```
<speak version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">
```

```

<s>This is the original, unmodified waveform:
  <audio src="message.wav"/>
</s>
<s>This is the same audio at twice the speed:
  <audio speed="2" src="message.wav"/>
</s>
<s>This is the same audio at half the original speed:
  <audio speed="-50%" src="message.wav"/>
</s>
<s>This is the same audio, also at half the original speed:
  <audio speed="50%" src="message.wav"/>
</s>
</speak>

```

3.3.2 mark Element

A [mark](#) element is an empty element that places a marker into the text/tag sequence. It has one *required* attribute, `name`, which is of type [xsd:token](#) [SCHEMA2 §3.3.2]. The [mark](#) element can be used to reference a specific location in the text/tag sequence, and can additionally be used to insert a marker into an output stream for asynchronous notification. When processing a [mark](#) element, a [synthesis processor](#) *must* do one or both of the following:

- inform the hosting environment with the value of the `name` attribute and with information allowing the platform to retrieve the corresponding position in the rendered output.
- when audio output of the SSML document reaches the [mark](#), issue an event that includes the *required* `name` attribute of the element. The hosting environment defines the destination of the event.

The [mark](#) element does not affect the speech output process.

```

<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">

Go from <mark name="here"/> here, to <mark name="there"/> there!

</speak>

```

3.3.3 desc Element

The [desc](#) element can only occur within the content of the [audio](#) element. When the audio source referenced in [audio](#) is not speech, e.g. audio wallpaper or sonicon punctuation, it should contain a [desc](#) element whose textual content is a description of the audio source (e.g. "door slamming"). If text-only output is being produced by the [synthesis processor](#), the content of the [desc](#) element(s) *should* be rendered instead of other alternative content in [audio](#). The *optional* `xml:lang` attribute can be used to indicate that the content of the element is in a different language from that of the content surrounding the element.

```

<?xml version="1.0"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">

  <!-- Normal use of <desc> -->
  Heads of State often make mistakes when speaking in a foreign language.
  One of the most well-known examples is that of John F. Kennedy:
  <audio src="ichbineinberliner.wav">If you could hear it, this would be
  a recording of John F. Kennedy speaking in Berlin.

```

```

    <desc>Kennedy's famous German language gaffe</desc>
  </audio>

  <!-- Suggesting the language of the recording -->
  <!-- Although there is no requirement that a recording be in the current language
        (since it might even be non-speech such as music), an author might wish to
        suggest the language of the recording by marking the entire <audio> element
        using <lang>. In this case, the xml:lang attribute on <desc> can be used
        to put the description back into the original language. -->
  Here's the same thing again but with a different fallback:
  <lang xml:lang="de-DE">
    <audio src="ichbineinberliner.wav">Ich bin ein Berliner.
    <desc xml:lang="en-US">Kennedy's famous German language gaffe</desc>
    </audio>
  </lang>
</speak>

```

The [desc](#) element can only contain descriptive text.

4. References

4.1 Normative References

[CSS2]

[Cascading Style Sheets, level 2: CSS2 Specification](#), B. Bos, et al., Editors. World Wide Web Consortium, 12 May 1998. This version of the CSS2 Recommendation is <http://www.w3.org/TR/1998/REC-CSS2-19980512/>. The [latest version of CSS2](#) is available at <http://www.w3.org/TR/REC-CSS2/>.

[IPA H NDBK]

[Handbook of the International Phonetic Association](#), International Phonetic Association, Editors. Cambridge University Press, July 1999. Information on the Handbook is available at <http://www.arts.gla.ac.uk/ipa/handbook.html>.

[PLS]

[Pronunciation Lexicon Specification \(PLS\) Version 1.0](#), P. Baggia, Editor. World Wide Web Consortium, 26 October 2006. This version of the PLS specification is <http://www.w3.org/TR/2006/WD-pronunciation-lexicon-20061026/> and is a work in progress. The [latest version of PLS](#) is available at <http://www.w3.org/TR/pronunciation-lexicon/>.

[RFC1521]

[MIME \(Multipurpose Internet Mail Extensions\) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies](#), N. Borenstein and N. Freed, Editors. IETF, September 1993. This RFC is available at <http://www.ietf.org/rfc/rfc1521.txt>.

[RFC2045]

[Multipurpose Internet Mail Extensions \(MIME\) Part One: Format of Internet Message Bodies](#), N. Freed and N. Borenstein, Editors. IETF, November 1996. This RFC is available at <http://www.ietf.org/rfc/rfc2045.txt>.

[RFC2046]

[Multipurpose Internet Mail Extensions \(MIME\) Part Two: Media Types](#), N. Freed and N. Borenstein, Editors. IETF, November 1996. This RFC is available at <http://www.ietf.org/rfc/rfc2046.txt>.

[RFC2119]

[Key words for use in RFCs to Indicate Requirement Levels](#), S. Bradner, Editor. IETF, March 1997. This RFC is available at <http://www.ietf.org/rfc/rfc2119.txt>.

[RFC3986]

[Uniform Resource Identifier \(URI\): Generic Syntax](#), T. Berners-Lee et al., Editors. IETF, January 2005. This RFC is available at <http://www.ietf.org/rfc/rfc3986.txt>.

[RFC3987]

[Internationalized Resource Identifiers \(IRIs\)](#), M. Duerst and M. Suignard, Editors. IETF, January 2005. This RFC is available at <http://www.ietf.org/rfc/rfc3987.txt>.

[RFC3066]

[Tags for the Identification of Languages](#), H. Alvestrand, Editor. IETF, January 2001. This RFC is available at <http://www.ietf.org/rfc/rfc3066.txt>.

[RFC4267]

[The W3C Speech Interface Framework Media Types: application/voicexml+xml, application/ssml+xml, application/srgs, application/srgs+xml, application/ccxml+xml, and application/pls+xml](#), M. Froumentin, Editor. IETF, November 2005. This RFC is available at <http://www.ietf.org/rfc/rfc4267.txt>.

[RFC4647]

[Matching of Language Tags](#), A. Phillips and M. Davis, Editors. IETF, September 2006. This RFC is available at <http://www.ietf.org/rfc/rfc4647.txt>.

[SCHEMA1]

[XML Schema Part 1: Structures](#), H. S. Thompson, et al., Editors. World Wide Web Consortium, 2 May 2001. This version of the XML Schema Part 1 Recommendation is <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>. The [latest version of XML Schema 1](#) is available at <http://www.w3.org/TR/xmlschema-1/>.

[SCHEMA2]

[XML Schema Part 2: Datatypes](#), P.V. Biron and A. Malhotra, Editors. World Wide Web Consortium, 2 May 2001. This version of the XML Schema Part 2 Recommendation is <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>. The [latest version of XML Schema 2](#) is available at <http://www.w3.org/TR/xmlschema-2/>.

[SMIL]

[Synchronized Multimedia Integration Language \(SMIL 2.1\)](#), D. Bulterman, et al., Editors. World Wide Web Consortium, 13 December 2005. This version of the SMIL 2 Recommendation is <http://www.w3.org/TR/2005/REC-SMIL2-20051213/>. The [latest version of SMIL2](#) is available at <http://www.w3.org/TR/SMIL2/>.

[TYPES]

[MIME Media types](#), IANA. This continually-updated list of media types registered with IANA is available at <http://www.iana.org/assignments/media-types/index.html>.

[XML 1.0]

[Extensible Markup Language \(XML\) 1.0 \(Fourth Edition\)](#), T. Bray et al., Editors. World Wide Web Consortium, 16 August 2006. This version of the XML 1.0 Recommendation is <http://www.w3.org/TR/2006/REC-xml-20060816/>. The [latest version of XML 1.0](#) is available at <http://www.w3.org/TR/REC-xml/>.

[XML 1.1]

[Extensible Markup Language \(XML\) 1.1 \(Second Edition\)](#), T. Bray et al., Editors. World Wide Web Consortium, 16 August 2006. This version of the XML 1.1 Recommendation is <http://www.w3.org/TR/2006/REC-xml11-20060816/>. The [latest version of XML 1.1](#) is available at <http://www.w3.org/TR/xml11/>.

[XML-BASE]

[XML Base](#), J. Marsh, Editor. World Wide Web Consortium, 27 June 2001. This version of the XML Base Recommendation is <http://www.w3.org/TR/2001/REC-xmlbase-20010627/>. The [latest version of XML Base](#) is available at <http://www.w3.org/TR/xmlbase/>.

[XML-ID]

[xml:id Version 1.0](#), J. Marsh et al., Editors. World Wide Web Consortium, 9 September 2005. This version of the xml:id Recommendation is <http://www.w3.org/TR/2005/REC-xml-id-20050909/>. The [latest version of xml:id](#) is available at <http://www.w3.org/TR/xml-id/>.

[XMLNS 1.0]

[Namespaces in XML 1.0 \(Second Edition\)](#), T. Bray et al., Editors. World Wide Web Consortium, 16 August 2006. This version of the XML Namespaces 1.0 Recommendation is <http://www.w3.org/TR/2006/REC-xml-names-20060816/>. The [latest version of XML Namespaces 1.0](#) is available at <http://www.w3.org/TR/REC-xml-names/>.

[XMLNS 1.1]

[Namespaces in XML 1.1 \(Second Edition\)](#), T. Bray et al., Editors. World Wide Web Consortium, 16 August 2006. This version of the XML Namespaces 1.1 Recommendation is <http://www.w3.org/TR/2006/REC-xml-names11-20060816/>. The [latest version of XML Namespaces 1.1](#) is available at <http://www.w3.org/TR/xml-names11/>.

4.2 Informative References

[BCP47]

[*Tags for Identifying Languages*](#), A. Phillips and M. Davis, Editors. IETF, September 2006. This RFC is available at <http://www.ietf.org/rfc/bcp/bcp47.txt>.

[DC]

Dublin Core Metadata Initiative. See <http://dublincore.org/>

[HTML]

[*HTML 4.01 Specification*](#), D. Raggett et al., Editors. World Wide Web Consortium, 24 December 1999. This version of the HTML 4 Recommendation is <http://www.w3.org/TR/1999/REC-html401-19991224/>. The [latest version of HTML 4](#) is available at <http://www.w3.org/TR/html4/>.

[IPA]

[*International Phonetic Association*](#). See <http://www.arts.gla.ac.uk/ipa/ipa.html> for the organization's website.

[IPAUNICODE1]

[*The International Phonetic Alphabet*](#), J. Esling. This table of IPA characters in Unicode is available at http://web.uvic.ca/ling/resources/ipa/charts/unicode_ipa-chart.htm.

[IPAUNICODE2]

[*The International Phonetic Alphabet in Unicode*](#), J. Wells. This table of Unicode values for IPA characters is available at <http://www.phon.ucl.ac.uk/home/wells/ipa-unicode.htm>.

[JEIDAALPHABET]

[*JEIDA-62-2000 Phoneme Alphabet*](#). JEITA. An abstract of this document (in Japanese) is available at <http://it.jeita.or.jp/document/publica/standard/summary/JEIDA-62-2000.pdf>.

[JEITA]

[*Japan Electronics and Information Technology Industries Association*](#). See <http://www.jeita.or.jp/>.

[JSML]

[*JSpeech Markup Language*](#), A. Hunt, Editor. World Wide Web Consortium, 5 June 2000. Copyright ©2000 Sun Microsystems, Inc. This version of the JSML submission is <http://www.w3.org/TR/2000/NOTE-jsml-20000605/>. The [latest W3C Note of JSML](#) is available at <http://www.w3.org/TR/jsml/>.

[LEX]

[*Pronunciation Lexicon Markup Requirements*](#), F. Scahill, Editor. World Wide Web Consortium, 12 March 2001. This document is a work in progress. This version of the Lexicon Requirements is <http://www.w3.org/TR/2001/WD-lexicon-reqs-20010312/>. The [latest version of the Lexicon Requirements](#) is available at <http://www.w3.org/TR/lexicon-reqs/>.

[RDF]

[*RDF Primer*](#), F. Manola and E. Miller, Editors. World Wide Web Consortium, 10 February 2004. This version of the RDF Primer Recommendation is <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>. The [latest version of the RDF Primer](#) is available at <http://www.w3.org/TR/rdf-primer/>.

[RDF-XMLSYNTAX]

[*RDF/XML Syntax Specification*](#), D. Beckett, Editor. World Wide Web Consortium, 10 February 2004. This version of the RDF/XML Syntax Recommendation is <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>. The [latest version of the RDF XML Syntax](#) is available at <http://www.w3.org/TR/rdf-syntax-grammar/>.

[RDF-SCHEMA]

[*RDF Vocabulary Description Language 1.0: RDF Schema*](#), D. Brickley and R. Guha, Editors. World Wide Web Consortium, 10 February 2004. This version of the RDF Schema Recommendation is <http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>. The [latest version of RDF Schema](#) is available at <http://www.w3.org/TR/rdf-schema/>.

[REQS]

[*Speech Synthesis Markup Requirements for Voice Markup Languages*](#), A. Hunt, Editor. World Wide Web Consortium, 23 December 1999. This document is a work in progress. This version of the Synthesis Requirements is <http://www.w3.org/TR/1999/WD-voice-tts->

reqs-19991223/. The [latest version of the Synthesis Requirements](http://www.w3.org/TR/voice-tts-reqs/) is available at <http://www.w3.org/TR/voice-tts-reqs/>.

[REQS11]

[Speech Synthesis Markup Language Version 1.1 Requirements](http://www.w3.org/TR/2007/WD-ssml11reqs-20070611/), D. Burnett and Z. Shuang, Editors. World Wide Web Consortium, 11 June 2007. This document is a work in progress. This version of the SSML 1.1 Requirements is <http://www.w3.org/TR/2007/WD-ssml11reqs-20070611/>. The [latest version of the SSML 1.1 Requirements](http://www.w3.org/TR/ssml11reqs/) is available at <http://www.w3.org/TR/ssml11reqs/>.

[RFC2616]

[Hypertext Transfer Protocol -- HTTP/1.1](http://www.ietf.org/rfc/rfc2616.txt), R. Fielding, et al., Editors. IETF, June 1999. This RFC is available at <http://www.ietf.org/rfc/rfc2616.txt>.

[RFC2732]

[Format for Literal IPv6 Addresses in URL's](http://www.ietf.org/rfc/rfc2732.txt), R. Hinden, et al., Editors. IETF, December 1999. This RFC is available at <http://www.ietf.org/rfc/rfc2732.txt>.

[RUBY]

[Ruby Annotation](http://www.w3.org/TR/2001/REC-ruby-20010531/), Marcin Sawicki, et al., Editors. World Wide Web Consortium, 31 May 2001. This version of the Ruby Recommendation is <http://www.w3.org/TR/2001/REC-ruby-20010531/>. The latest version is available at <http://www.w3.org/TR/ruby/>.

[SABLE]

"SABLE: A Standard for TTS Markup", Richard Sproat, et al. *Proceedings of the International Conference on Spoken Language Processing*, R. Mannell and J. Robert-Ribes, Editors. [Causal Productions Pty Ltd](http://www.causalproductions.com/) (Adelaide), 1998. Vol. 5, pp. 1719-1722. Conference proceedings are available from the publisher at <http://www.causalproductions.com/>.

[SSML]

[Speech Synthesis Markup Language \(SSML\) Version 1.0](http://www.w3.org/TR/2004/REC-speech-synthesis-20040907/), Daniel C. Burnett, et al., Editors. World Wide Web Consortium, 7 September 2004. This version of the SSML 1.0 Recommendation is <http://www.w3.org/TR/2004/REC-speech-synthesis-20040907/>. The latest version is available at <http://www.w3.org/TR/speech-synthesis/>.

[UNICODE]

[The Unicode Standard](http://www.unicode.org/standard/standard.html). The Unicode Consortium. Information about the Unicode Standard and its versions can be found at <http://www.unicode.org/standard/standard.html>.

[VXML]

[Voice Extensible Markup Language \(VoiceXML\) Version 2.0](http://www.w3.org/TR/2004/REC-voicexml20-20040316/), S. McGlashan, et al., Editors. World Wide Web Consortium, 16 March 2004. This version of the VoiceXML 2.0 Recommendation is <http://www.w3.org/TR/2004/REC-voicexml20-20040316/>. The [latest version of VoiceXML 2](http://www.w3.org/TR/voicexml20/) is available at <http://www.w3.org/TR/voicexml20/>.

[WS]

[Minutes](http://www.w3.org/2005/08/SSML/ssml-workshop-agenda.html), W3C Workshop on Internationalizing the Speech Synthesis Markup Language, 2-3 November 2005. The agenda and minutes are available at <http://www.w3.org/2005/08/SSML/ssml-workshop-agenda.html>.

[WS2]

[Minutes](http://www.w3.org/2006/02/SSML/agenda.html), W3C Workshop on Internationalizing the Speech Synthesis Markup Language, 30-31 May 2006. The agenda is available at <http://www.w3.org/2006/02/SSML/agenda.html>. The minutes are available at <http://www.w3.org/2006/02/SSML/minutes.html>.

[WS3]

[Minutes](http://www.w3.org/2006/10/SSML/agenda.html), W3C Workshop on Internationalizing the Speech Synthesis Markup Language, 13-14 January 2007. The agenda is available at <http://www.w3.org/2006/10/SSML/agenda.html>. The minutes are available at <http://www.w3.org/2006/10/SSML/minutes.html>.

5. Acknowledgments

This document was written with the participation of the following participants in the W3C Voice Browser Working Group (*listed in family name alphabetical order*):

Max Froumentin, W3C
Jim Larson, Intel

Wai-Kit Lo, Chinese University of Hong Kong
 Mark Walker, Intel
 夏海荣 (XIA Hairong), Panasonic

Appendix A: Audio File Formats

This appendix is normative.

SSML requires that a platform support the playing of the audio formats specified below.

Required audio formats

Audio Format	Media Type
Raw (headerless) 8kHz 8-bit mono mu-law (PCM) single channel. (G.711)	audio/basic (from [RFC1521])
Raw (headerless) 8kHz 8 bit mono A-law (PCM) single channel. (G.711)	audio/x-alaw-basic
WAV (RIFF header) 8kHz 8-bit mono mu-law (PCM) single channel.	audio/x-wav
WAV (RIFF header) 8kHz 8-bit mono A-law (PCM) single channel.	audio/x-wav

The 'audio/basic' MIME type is commonly used with the 'au' header format as well as the headerless 8-bit 8kHz mu-law format. If this MIME type is specified for playing, the mu-law format *must* be used. For playback with the 'audio/basic' MIME type, processors *must* support the mu-law format and *may* support the 'au' format.

Appendix B: Internationalization

This appendix is normative.

SSML is an application of XML [XML 1.0 or XML 1.1] and thus supports [UNICODE] which defines a standard universal character set.

SSML provides a mechanism for control of the spoken language via the use of the `xml:lang` attribute. Language changes can occur as frequently as per token (word), although excessive language changes can diminish the output audio quality. SSML also permits finer control over output pronunciations via the `lexicon` and `phoneme` elements, features that can help to mitigate poor quality default lexicons for languages with only minimal commercial support today.

Appendix C: MIME Types and File Suffix

This appendix is normative.

The media type associated with the Speech Synthesis Markup Language specification is "application/ssml+xml" and the filename suffix is ".ssml" as defined in [RFC4267].

Appendix D: Schema for the Speech Synthesis Markup Language

This appendix is normative.

The synthesis schema is located at <http://www.w3.org/TR/speech-synthesis/synthesis.xsd>.

Note: the synthesis schema includes a no-namespace core schema, located at <http://www.w3.org/TR/speech-synthesis/synthesis-core.xsd>, which *may* be used as a basis for specifying Speech Synthesis Markup Language Fragments ([Sec. 2.2.1](#)) embedded in non-synthesis namespace schemas.

Appendix E: DTD for the Speech Synthesis Markup Language

This appendix is informative.

The SSML DTD is located at <http://www.w3.org/TR/speech-synthesis/synthesis.dtd>.

Due to DTD limitations, the SSML DTD does not correctly express that the [metadata](#) element can contain elements from other XML namespaces.

Appendix F: Example SSML

This appendix is informative.

The following is an example of reading headers of email messages. The [p](#) and [s](#) elements are used to mark the text structure. The [break](#) element is placed before the time and has the effect of marking the time as important information for the listener to pay attention to. The [prosody](#) element is used to slow the speaking rate of the email subject so that the user has extra time to listen and write down the details.

```
<?xml version="1.0"?>
<!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESIS 1.0//EN"
    "http://www.w3.org/TR/speech-synthesis/synthesis.dtd">
<speak version="1.1"
    xmlns="http://www.w3.org/2001/10/synthesis"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
    xml:lang="en-US">
  <p>
    <s>You have 4 new messages.</s>
    <s>The first is from Stephanie Williams and arrived at <break/> 3:45pm.
    </s>
    <s>
      The subject is <prosody rate="-20%">ski trip</prosody>
    </s>
  </p>
</speak>
```

The following example combines audio files and different spoken voices to provide information on a collection of music.

```
<?xml version="1.0"?>
<!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESIS 1.0//EN"
    "http://www.w3.org/TR/speech-synthesis/synthesis.dtd">
<speak version="1.1"
    xmlns="http://www.w3.org/2001/10/synthesis"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
        http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
    xml:lang="en-US">
  <p>
    <voice gender="male">
      <s>Today we preview the latest romantic music from Example.</s>

      <s>Hear what the Software Reviews said about Example's newest hit.</s>
    </voice>
  </p>

  <p>
    <voice gender="female">
```

```

    He sings about issues that touch us all.
  </voice>
</p>

<p>
  <voice gender="male">
    Here's a sample. <audio src="http://www.example.com/music.wav"/>
    Would you like to buy it?
  </voice>
</p>

</speak>

```

It is often the case that an author wishes to include a bit of foreign text (say, a movie title) in an application without having to switch languages (for example via the [lang](#) element). A simple way to do this is shown here. In this example the synthesis processor would render the movie name using the pronunciation rules of the container language ("en-US" in this case), similar to how a reader who doesn't know the foreign language might try to read (and pronounce) it.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">

  The title of the movie is:
  "La vita è bella"
  (Life is beautiful),
  which is directed by Roberto Benigni.
</speak>

```

With some additional work the output quality can be improved tremendously either by creating a custom pronunciation in an external lexicon (see [Section 3.1.5](#)) or via the [phoneme](#) element as shown in the next example.

It is worth noting that IPA alphabet support is an *optional* feature and that phonemes for an external language may be rendered with some approximation (see [Section 3.1.5](#) for details). The following example only uses phonemes common to US English.

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">

  The title of the movie is:
  <phoneme alphabet="ipa"
    ph="&#x2C8;l&#x251; &#x2C8;vi&#x2D0;&#x27E;&#x259; &#x2C8;&#x294;e&#x26A; &#x2C8;b&#x25B;l&#x259;">
  La vita è bella </phoneme>
  <!-- The IPA pronunciation is 'la 'vi:ɾə 'ʔei 'bɛlə -->
  (Life is beautiful),
  which is directed by
  <phoneme alphabet="ipa"
    ph="&#x279;&#x259;&#x2C8;b&#x25B;&#x2D0;&#x279;&#x27E;o&#x28A; b&#x25B;&#x2C8;ni&#x2D0;nji">
  Roberto Benigni </phoneme>
  <!-- The IPA pronunciation is ɹə'beɪnrou bɛ'ni:nji -->

  <!-- Note that in actual practice an author might change the
    encoding to UTF-8 and directly use the Unicode characters in
    the document rather than using the escapes as shown.
    The escaped values are shown for ease of copying. -->
</speak>

```

SMIL Integration Example

The SMIL language [[SMIL](#)] is an XML-based multimedia control language. It is especially well suited for describing dynamic media applications that include synthetic speech output.

File **'greetings.ssmil'** contains the following:

```
<?xml version="1.0"?>
<!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESIS 1.0//EN"
    "http://www.w3.org/TR/speech-synthesis/synthesis.dtd">

<speak version="1.1"
  xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
    http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
  xml:lang="en-US">

  <s>
    <mark name="greetings"/>
    <emphasis>Greetings</emphasis> from the <sub alias="World Wide Web Consortium">W3C</sub>!
  </s>
</speak>
```

SMIL Example 1: W3C logo image appears, and then one second later, the speech sequence is rendered. File **'greetings.smil'** contains the following:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <top-layout width="640" height="320">
      <region id="whole" width="640" height="320"/>
    </top-layout>
  </head>
  <body>
    <par>
      
      <ref src="greetings.ssmil" begin="1s"/>
    </par>
  </body>
</smil>
```

SMIL Example 2: W3C logo image appears, then clicking on the image causes it to disappear and the speech sequence to be rendered. File **'greetings.smil'** contains the following:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <top-layout width="640" height="320">
      <region id="whole" width="640" height="320"/>
    </top-layout>
  </head>
  <body>
    <seq>
      
      <ref src="greetings.ssmil"/>
    </seq>
  </body>
</smil>
```

VoiceXML Integration Example

The following is an example of SSML in VoiceXML (see [Section 2.3.3](#)) for [voice browser](#) applications. It is worth noting that the VoiceXML namespace includes the SSML namespace elements and attributes. See Appendix O of [[VXML](#)] for details.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
    http://www.w3.org/TR/voicexml20/vxml.xsd">
  <form>
```

```

<block>
  <prompt>
    <emphasis>Welcome</emphasis> to the Bird Seed Emporium.
    <audio src="rtsp://www.birdsounds.example.com/thrush.wav"/>
    We have 250 kilogram drums of thistle seed for
    $299.95
    plus shipping and handling this month.
    <audio src="http://www.birdsounds.example.com/mourningdove.wav"/>
  </prompt>
</block>
</form>
</vxml>

```

Appendix G: Changes since SSML 1.0

Changes in draft 10 January 2007:

- Added <w> element to mark the boundaries of a word.
- Added role attribute onto <w> element to provide more explicit linkage with PLS lexicons.
- Introduced lang-voice attribute on <speak> (behavioral explanation is in 3.1.2). This attribute controls whether and how automatic voice selection can occur upon change in language.
- In 3.1.2, xml:lang is defined to describe the language of the text content and is added onto the new <w> element.
- Added xml:id attribute onto <lexicon>, <p>, <s>, and <w> elements to enhance external references into SSML content.
- Added <lookup> element to control which lexicons are currently in use. <lexicon> now only defines which lexicons are used in the document.
- Removed general text describing how text may be mapped to entries in the lexicon.
- The default <lexicon> type is now "application/pls+xml", as defined by the PLS 1.0 specification.
- Introduced the notion of a Pronunciation Alphabet Registry that would maintain a list of registered values for the alphabet attribute of the <phoneme> element.
- Removed the xml:lang attribute from the <voice> element to reduce confusion.
- Added <lang> element to allow setting xml:lang for arbitrary text content.
- Clarified in <voice> description that indication of language and voice are independent, no synthesis processor is required to support all combinations thereof, and processors must document behavior for every combination thereof.
- Changed several instances of "lexicon document" to "pronunciation lexicon document".

Changes in draft 11 June 2007:

- 3.1.5: SSCR1: Now mandates that processors use the pronunciations given in PLS lexicons.
- 3.1.5.2: SSCR9: Clarified that the processor already has built-in system lexicons whose values are overridden by use of the <lexicon> and <lookup> elements.
- SSCR10: Changed all generic instances of "pronunciation lexicon document" to "lexicon document".
- Editorial changes: SSCR11, SSCR17
- SSCR21: 3.1.8.2: Clarified that prosodic markup is permitted within <w>. Also, whitespace is now significant within the <w> element, and the entire text content of <w> will be treated as a single word for lexical lookup purposes.
- SSCR5: Updated entire document to allow for XML and XMLNS 1.1 in addition to 1.0. Clarified in definition of URI that IRIs are allowed and added an informative reference to RFC3987.
- SSCR20: Completely revamped how voice selection and language speaking control are done. Removed the lang-voice attribute and related text. There are several new attributes on the <voice> element and one new attribute on all elements that take the xml:lang attribute.
- Added duration, volume, and rate attributes to <speak> to accommodate expected VoiceXML 3 needs.
- Added duration attributes to <audio> to accommodate expected VoiceXML 3 needs.

- Revised error text in <audio> description to better explain appropriate notification behavior.

Changes in this draft:

- SSCR1: In 3.1.5, clarified that information in PLS documents must be used by the processor.
- SSCR3: In 2.2.3, added example of using Ruby within SSML. Added Informative reference to Ruby.
- SSCR8: Changed many instances of "word" to "token" to clarify when the specification is referring to tokens; this is particularly relevant in discussions of parsing and linkage with lexicons.
- SSCR16: Added text in status section explaining the motivation and background for SSML 1.1. Also added the requirements document and the workshops to the informative references section.
- SSCR32: Applied styling to relevant uses of RFC2119 keywords.
- SSCR33: Renamed "duration" attributes to be "trimming" attributes.
- SSCR35: In 3.2.1, added the "ordering" attribute and adjusted existing text to refer to it.
- SSCR37: Clarified in status section that we are referring to natural (human) languages. Clarified in 3.1.2 and 3.1.8.2 that xml:lang indicates the language of the *written* content. Clarified in 3.1.8.2 that white space is preserved within <w>. Clarified how the languages attribute value in 3.2.1 is used in voice selection, and disallowed "und" and "zxx". Removed issue note in 3.1.12 about changing the name to .
- SSCR38: Moved a paragraph from 3.1.5 to 3.1.5.2 that explains when the information in a lexicon document must be used.
- SSCR39: Clarified in 1.2 that the tokens may not span markup elements except within the <token> and <w> elements.
- In 3.2.1, clarified that voice selection is scoped.
- In 3.1.8.2, the <w> element has been renamed to <token>. Then, <w> was added as an alias for <token>.
- Added section 3.3.1.1, <audio> trimming attributes.
- Added section 3.3.1.2, the gain attribute for <audio>.
- Added section 3.3.1.3, the speed attribute for <audio>.
- Removed sections 3.1.1.2 and 3.1.1.3, the volume and rate attributes for <speak>. Also removed the startoffset and endoffset attributes from 3.1.1.1.