Part III

Data Representation Formats

Speech Processing for IP Networks: Media Resource Control Protocol (MRCP) Dave Burke
 ISBN: 0-470-02834-3

8

Speech Synthesis Markup Language (SSML)

The MRCP speech synthesiser resource supports both plain text and marked-up text as input to be spoken. This chapter introduces the standard markup format that speech synthesiser resources are required to support, namely the W3C's Speech Synthesis Markup Language (SSML) [8]. SSML also comes up in the context of VoiceXML (see Chapter 16) as an author-friendly language for providing control over the speech synthesis process. Readers new to XML might find it useful to review Appendix B at this point.

8.1 Introduction

SSML is a standard, XML-based, markup annotation for instructing speech synthesisers how to convert written language input into spoken language output. SSML is primarily intended to help application authors improve the quality of synthesised text by providing a standard way of controlling aspects of the speech output such as pronunciation, volume, pitch and rate. SSML can also express playback of prerecorded audio. A number of key design criteria were chosen in the standardisation efforts that led to the development of SSML:

- *Consistency*: Provide predictable control of voice output across different speech synthesisers.
- *Interoperability*: Work with other W3C specifications such as VoiceXML.
- *Generality*: Support a wide range of applications with varied speech content.
- *Internationalisation*: Enable speech output in a large number of languages.
- *Generation and readability*: Support both automatic generation and hand authoring. The latter implies human readability.
- *Implementable*: The specification is designed to be implementable with existing, generally available technology.

Different speech synthesiser implementations use different underlying speech processing techniques (see Chapter 2). Hence the synthesised result for a given input text will inevitably vary across different implementations. SSML does not attempt to provide specific control of speech synthesisers but rather just provides a way for the application author to make prosodic and other information available

Speech Processing for IP Networks: Media Resource Control Protocol (MRCP) Dave Burke
 ISBN: 0-470-02834-3

to the speech synthesiser that it would otherwise not be able to acquire on its own. In the end, it is up to the speech synthesiser to determine whether and in what way to use this information.

8.2 Document structure

SSML documents are identified by the media type `application/ssml+xml`. Table 8.1 summarises the elements and attributes defined in SSML.

The basic structure of an SSML document is illustrated below:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       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-GB">
     Hello world!
</speak>
```

All SSML documents include the root element `<speak>`. The `version` attribute indicates the version of SSML and is fixed at 1.0 for the specification described in [8]. The default namespace for the SSML `<speak>` element and its children is indicated by the `xmlns` attribute and is defined as `http://www.w3.org/2001/10/synthesis`.

The `xmlns:xsi` attribute associates the namespace prefix of `xsi` to the namespace name `http://www.w3.org/2001/XMLSchema-instance`. The namespace prefix is defined since it is needed for the attribut, `xsi:schemaLocation.` The `xsi:schemaLocation` attribute indicates the location of the schema to validate the SSML document against. The first attribute value contains the SSML namespace followed by a hint to the location of the schema document. In the rest of the examples in this chapter, we will omit the `xmlns:xsi` and `xsi:schemaLocation` attributes for clarity. In doing so, this does not invalidate the markup although the SSML specification does recommend that both attributes be present. In reality, high-density systems typically may not validate against a schema for each and every document that arrives due to the significant processing overhead that is required for validation.

The `xml:lang` attribute indicates the language for the document and optionally also indicates a country or other variation. The format for the `xml:lang` value follows the language tag syntax described in RFC 3066 [55]. The syntax includes a two- or three-letter language identifier followed by a hyphen, and a two-letter country identifier (although other information can be registered here also). The language tag syntax is case insensitive although by convention the language subtag tends to be written in lower case and the country identifier in upper case characters. Table 8.2 illustrates examples of language identifiers.

It is possible to override the language within the document by including the `xml:lang` attribute on other elements as we will see. If the `xml:lang` attribute is omitted, the speech synthesis uses a default language. The contents of the `<speak>` element contains the text to synthesis and optionally further markup elements.

The `<p>` element and `<s>` element can be used explicitly to demarcate paragraphs and sentences. If they are omitted, the synthesiser will attempt to determine the structure using language-specific knowledge of the format of the plain text (e.g. the presence of a full stop, '.', to mark the end of the sentence). The `xml:lang` attribute may be specified on the `<p>` and `<s>` elements to indicate the language of the contents.

Table 8.1 Elements and attributes defined in SSML

Elements	Attributes	Description

`<speak>`	`version`
	`xmlns`
	`xml:lang`
	`xmlns:xsi`
	`xsi:schemaLocation`
	`xml:base`
		Root element for SSML documents.

`<lexicon>`	`uri` `type`	References an external pronunciation lexicon document.
`<p>`	`xml:lang`	Explicitly demarcates a paragraph.
`<s>`	`xml:lang`	Explicitly demarcates a sentence.
`<audio>`	`src`	Inserts a recorded audio file.
`<desc>`	`xml:lang`	Container element for a description of the audio source.
`<phoneme>`	`ph` `alphabet`	Provides a phonemic/phonetic pronunciation for the contained text.
`<sub>`	`alias`	Provides acronym / abbreviation expansions.

`<say-as>`	`interpret-as`
	`format`
	`detail`
		Used to indicate information on the type of text construct contained within the element.

`<break>`	`time` `strength`	Controls the pausing or other prosodic boundaries between words.
`<emphasis>`	`level`	Requests that the contained text be spoken with emphasis.

`<voice>`	`xml:lang`
	`gender`
	`age`
	`variant`
	`name`
		Requests a change to the speaking voice.

`<prosody>`	`pitch`
	`contour`
	`range`
	`rate`
	`duration`
	`volume`
		Provides control of the pitch, speaking rate and volume of the speech output.

`<mark>`	`name`	Places a marker into the text/tag sequence.

`<meta>`	`name`
	`http-equiv`
	`content`
		Contains metadata for the document.

`<metadata>`	—	Contains metadata for the document.

Table 8.2 Language identifier examples

Identifer	Meaning
en-GB	English, Great Britain
en-US	English, United States
en-IE	English, Ireland
fr-FR	French, France
fr-CA	French, Canada
de-DE	German, Germany
de-CH	German, Switzerland
it-IT	Italian, Italy
da-DK	Danish, Denmark
nl-NL	Dutch, Netherlands
sv-SE	Swedish, Sweden
es-ES	Spanish, Spain
pt-BR	Portuguese, Brazil

An example SSML document using these elements is illustrated below:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-GB">
   <p>
       <s>This is the first sentence in a paragraph</s>
       <s>This is a second sentence in a paragraph</s>
   </p>
</speak>
```

Most speech synthesisers internally provide extensive high quality lexicons with pronunciation information for many words or phrases. The `<lexicon>` element can be used to indicate an additional external pronunciation lexicon document identified by a URI for use in conjunction with the SSML document. A lexicon maps the orthography or written form of words and phrases to an associated pronunciation expressed in some sort of phonetic alphabet. The W3C has created a standard pronunciation lexicon format (see Chapter 11 for more information). One or more `<lexicon>` elements may be present as immediate children of the `<speak>` element. The `uri` attribute indicates the location of the external lexicon document and the optional `type` attribute specifies the type of the document (`application/pls+xml` is the standard media type for the W3C's Pronunciation Lexicon Specification [9]). For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-GB">
  <lexicon uri="http://www.example.com/lexicon.pls"
                type="application/pls+xml"/>
       The centre is equidistant from the corners.
</speak>
```

The `xml:base` attribute may be specified on the `<speak>` element to resolve URIs used within `<lexicon>` and `<audio>` elements against. For example, if the `uri` attribute on `<lexicon>` specifies `names.pls` and the `xml:base` attribute on `<speak>` specifies `http://www.example.com/`, then the URI `http://www.example.com/names.pls` is used to identify the external lexicon.

8.3 Recorded audio

A simple yet powerful capability in SSML is its ability to play recorded audio referenced by a URI using the `<audio>` element. This allows the insertion of recorded audio within synthesised speech output or even creation of SSML documents that just play prerecorded audio. Table 8.3 lists the audio formats that need to be supported by SSML-compliant implementations. Other formats may be supported. The audio format can be identified[1] by its media type e.g. as supplied in the `Content-Type` header in a HTTP response.

Although SSML makes no explicit requirements in this area, common URI formats supported are `http:` and `https`: for fetching of files via the Hyper Text Transfer Protocol (HTTP) (see Appendix C, located on the Web at http://www.daveburke.org/speechprocessing/), `file:` for accessing local files, and `rtsp`: for streaming files using the Real-Time Streaming Protocol (RTSP) [54]. Unfortunately, SSML does not include capabilities to control fetching such as timeouts and cache control settings. VoiceXML (see Chapter 16), for example, extends SSML by adding the `fetchtimeout`, `fetchhint`, `maxage` and `maxstale` attributes to `<audio>`. The `fetchtimeout` attribute specifies the interval to wait for the content to be returned before giving up. The `fetchhint` attribute defines when the content should be retrieved from the server: a value of `prefetch` indicates that the content may be fetched when the page is loaded and a value of `safe` indicates that the content need only be fetched when actually needed. The `maxage` attribute originates from HTTP [12] and indicates that the SSML document is willing to use content whose age is no greater than the specified time in seconds. The `maxstale` attribute also originates from HTTP and indicates that the document is willing to use content that has exceeded its expiration time by the specified number of seconds. Without these extensions, the author is left to use the defaults of the speech synthesiser. A future version of SSML may address these shortcoming by adding similar attributes.

Table 8.3 SSML audio formats

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

[1] There are cases when the speech synthesiser has to determine the audio type itself, e.g. if the `Content-Type` is omitted in a HTTP response or the audio URI is a local file. In these cases, some speech synthesisers are able to open the file and analyse it, e.g. look for an identifying header.

An example of an SSML document containing synthesised and pre-recorded audio is illustrated below:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-GB">
     Please say your name after the beep.
     <audio src="http://www.example.com/beep.wav"/>
</speak>
```

SSML also provides a fallback capability that allows the application author to specify alternate content to be spoken if an audio resource is not available. In the following example, the `company-intro.wav` file is unavailable (e.g. the Web server is malfunctioning) and as a result the text within the audio is spoken instead:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-GB">
    <audio src="http://www.example.com/company-intro.wav">
       Welcome to AJAX, the asynchronous company.
    </audio>
</speak>
```

SSML includes an element called `<desc>`, which can be placed as a child of `<audio>` and whose textual content is a description of the audio source (e.g. 'screeching brakes'). A speech synthesiser, operating in a text-only output mode (as part of an accessible solution for the hearing impaired, for example), can use this element to provide descriptive text in place of recorded audio. The `<desc>` element also has an `xml:lang` attribute that may be used to specify the language of the descriptive text.

8.4 Pronunciation

In this section, we look at SSML mechanisms for controlling pronunciation.

8.4.1 Phonemic/phonetic content

The `<phoneme>` element allows an application author to insert content to be spoken in terms of a phonemic or phonetic alphabet. A phonemic alphabet consists of phonemes – language-dependent speech units that represent all the sounds needed to distinguish one word from another. A phonetic alphabet, on the other hand, consists of phones – language-independent speech units that characterise the manner (puff of air, click, vocalized, etc.) and place (front, middle, back, etc.) of articulation within the human vocal tract. Speech synthesisers implementing SSML usually support the International Phonetic Alphabet (IPA), which is mostly expressible in Unicode (IPA is discussed in Section 2.1.1). Many speech synthesisers also support their own proprietary alphabet formats. The `<phoneme>` element is useful for rendering words that a speech synthesiser is not able to render appropriately, for example some proper nouns or words from a different language.

An example using the <phoneme> element is illustrated below:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-US">
   <phoneme alphabet="ipa"
            ph="t&#x259;mei&#x325;&#x27E;ou&#325;">
        tomato
   </phoneme>
</speak>
```

The alphabet attribute on <phoneme> identifies the phonemic/phonetic alphabet in use – in this case IPA. Vendor-specific alphabets must commence with 'x-'. The ph attribute contains the string of phonetic characters to synthesise. The text content of the <phoneme> element is optional and is provided only to make the content more human readable – the speech synthesiser ignores it. For the ph attribute value, this example uses the XML entity escaped versions of the IPA characters encoded in UTF-8 (this is common practice since many editors cannot correctly cut and past Unicode characters).

8.4.2 Substitution

The purpose of the <sub> element is to allow an SSML document to contain both a written form and a spoken form – i.e. to make the document more human readable. The written form is contained as text within the <sub> element; the spoken form used by the speech synthesis is inserted into the alias attribute on <sub>. In the following example, the speech synthesiser speaks 'Media Resource Control Protocol':

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-US">
   <sub alias="Media Resource Control Protocol">
        MRCP
   </sub>
</speak>
```

Another common use of the <sub> element is to make sure an acronym is properly spoken. The following example ensures that the constituent letters of the acronym IPA are spoken:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-US">
   <sub alias="I P A">IPA</sub>
</speak>
```

8.4.3 Interpreting text

The <say-as> element is used to indicate information about the type of text construct contained within the element and to help specify the level of detail for rendering the contained text. Interpreting the contained text in different ways will typically result in a different pronunciation of the content (although a speech synthesiser is still required to pronounce the contained text in a manner consistent with how such content is normally produced for the language).

The <say-as> element has three attributes: interpret-as, format and detail. The format, and detail attributes are optional. The interpret-as attribute indicates the content type of the contained text construct, e.g. date to indicate a date, or telephone to indicate a telephone number. The optional format attribute provides further hints on the precise formatting of the contained text, e.g. a value of dmy could be used to indicate that a date should be spoken in the format of date, then month, then year. The optional detail attribute indicates the level of detail to be spoken although it is not defined for many interpret-as types.

The values for the <say-as> attributes are not specified in the SSML specification itself but rather in a separate W3C Note [53]. Values for interpret-as, format and detail are simply ignored if they are not recognised by the speech synthesiser. Below are some common examples of <say-as>:

```
<say-as interpret-as="date" format="mdy">2/3/2006</say-as>
<!-- Interpreted as 3rd of February 2006 -->

<say-as interpret-as="time" format="hms24">01:59:59</say-as>
<!-- Interpreted as 1 second before 2 o'clock in the morning -->

<say-as interpret-as="ordinal">12</say-as>
<!-- Spoken in English as "twelfth" -->

<say-as interpret-as="cardinal">9</say-as>
<!-- Spoken in English as "nine" -->

<say-as interpret-as="characters" format="characters">WAY</say-as>
<!-- Spoken as letters W, A, Y and not the word "WAY" -->
```

8.5 Prosody

In this section, we look at the mechanisms SSML provides to control prosody – patterns of stress and intonation in the spoken audio.

8.5.1 Prosodic boundaries

The <break> element is used to insert explicit pauses or prosodic boundaries between words. When this element is not present, the speech synthesiser determines the break based on linguistic content. The <break> element has two attributes: time and strength. The time attribute indicates the duration to pause in seconds or milliseconds. The strength attribute indicates the strength of the prosodic break in the speech output. Allowed values are none, x-weak, weak, medium (the default value), strong, or x-strong. The value none indicates that no prosodic break boundary should

be outputted and is used to prevent a prosodic break in places where the speech synthesiser would have otherwise inserted one. The other values indicate increasing break strength between words. The stronger values are typically accompanied by pauses. The `time` and `strength` attributes are optional: if both are omitted, a break will be produced by the speech synthesiser with a prosodic strength greater than that which the processor would otherwise have used if no `<break>` element was supplied. A simple example is:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-GB">
    Take a deep breath and count to three <break time="3s"/>
    Now we are ready to proceed.
</speak>
```

8.5.2 Emphasis

The `<emphasis>` element is used to indicate that the contained text should be spoken with prominence or stress. Emphasis may be realised with a combination of pitch changes, timing changes, volume and other acoustic differences. The `<emphasis>` element has a single optional attribute called `level` that indicates the strength of the emphasis to apply. Allowable values for this attribute are: `strong`, `moderate` (the default value), `none`, and `reduced`. The `none` level is used to prevent the speech synthesiser from emphasising words that it might otherwise emphasise. The `reduced` level is effectively the opposite of emphasising a word. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-GB">
    That is the <emphasis level="strong">right</emphasis> answer!
</speak>
```

8.5.3 Speaking voice

The `<voice>` element is used to request a change in speaking voice. The `<voice>` element has five optional attributes that can be used to indicate various aspects of the requested voice: `xml:lang`, `gender`, `age`, `name`, and `variant`. The `xml-lang` attribute is used to indicate the language of the content. The `gender` attribute indicates the preferred gender of the voice and may take a value of `male`, `female` or `neutral`. The `age` specifies the preferred age in years since birth of the voice. The `name` attribute can be used to indicate an implementation-specific voice name. The value may be a space-separated list of names ordered from top preference down. Finally, the `variant` attribute can be used to provide a positive integer indicating a preferred variant of the other voice characteristics to speak the contained text (e.g. the third female voice for language en-GB).

When a speech synthesiser encounters a `<voice>` element and a single voice is available that exactly matches the attributes specified, then that voice is used. Otherwise, the following steps are used to find a voice:

(i) If the `xml:lang` attribute is specified and a voice is available for that language, it must be used. If there are multiple such voices available, the speech synthesis typically chooses the voice that best matches the specified values for `name`, `variant`, `gender` and `age`.
(ii) If there is no voice that matches the requested `xml:lang`, then the speech synthesiser may choose one that is closely related, for example a variation on the dialect for a language.
(iii) Otherwise an error may be returned. In the context of MRCP, this means a `Completion-Cause` of `005 language-unsupported` may be returned in the `SPEAK-COMPLETE` event (see Chapter 12).

Since changing voice is quite a drastic operation (different voices may have different defaults for pitch, volume, rate, etc.), it is usually recommended only on sentence or paragraph boundaries. The following example illustrates use of the `<voice>` element:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-GB">
    <voice gender="male">
         Welcome to online shopping for him
    </voice>
    <voice gender="female">
         and her!
    </voice>
</speak>
```

8.5.4 Prosodic control

The `<prosody>` element enables the application author to control the volume, speaking rate and pitch of the speech output of the contained text within the element. The `<prosody>` element has six optional attributes: `volume` for indicating the volume for the contained text, `pitch`, `range` and `contour` to specify pitch information, and `rate` and `duration` to control the speaking rate. The `<prosody>` attribute values should be interpreted as 'hints' to the speech synthesiser. If the `<prosody>` attribute value is outside the permissible range for the speech synthesiser, a best effort to continue processing is made by setting that attribute value as close to the desired value as possible. A speech synthesiser may also choose to ignore a `<prosody>` attribute value if it would otherwise result in degraded speech quality.

8.5.4.1 Volume

The value of the `volume` attribute can either be a number, a relative change or a label. Legal number values may range from 0.0 to 100.0 where higher values are louder and 0 is silent. Relative changes are specified with a '+' or '−' character followed by a number and a '%' character (e.g. '+10%') or just a number (e.g. '+15'). Legal labels are `silent`, `x-soft`, `soft`, `medium`, `loud`, `x-loud`, or `default`. Labels `silent` through `x-loud` represent a linearly increasing sequence of volume levels. For example

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

```
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-GB">
   Please consult our website for job vacancies.
   <prosody volume="soft">
        ACME is an equal-opportunities employer
   </prosody>
</speak>
```

8.5.4.2 Speaking rate

The speaking rate is controlled by the `rate` and `duration` attributes. The value of the `rate` attribute can either be a relative change or a label. When a number is used as a relative change, its value is interpreted as a multiplier, e.g. '+2' indicates a doubling of the rate. Legal labels are `x-slow`, `slow`, `medium`, `fast`, `x-fast`, or `default`. Labels `x-slow` through `x-fast` represent an increasing sequence of rates. The `default` rate is the normal speaking rate for the voice when reading text aloud. Alternatively, the `duration` attribute can be used to express the rate indirectly by specifying a value in milliseconds or seconds for the desired time to read the text contained in the `<prosody>` element, e.g. '750 ms' or '5 s'. If both `rate` and `duration` are specified, `duration` takes precedence. For example

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-IE">
   Please select an option from the following.
   <prosody rate="-30%">
        Press 1 for sales or press 2 for support.
   </prosody>
   Otherwise hold the only line and a customer
   service representative will be with you shortly.
</speak>
```

8.5.4.3 Pitch

Pitch is controlled by the `pitch`, `range`, and `contour` attributes on the `<prosody>` element. The `pitch` attribute sets the baseline pitch for the contained text of the `<prosody>` element. Increasing the value for `pitch` will increase the approximate pitch of the spoken output. The value of the `pitch` attribute can either be a number followed by 'Hz', a relative change, or a label. As before, a relative change is indicated by a '+' or '−' followed by a number or a percentage e.g. '+3' or '+80%'. Legal label values are: `x-low`, `low`, `medium`, `high`, `x-high`, or `default`. Labels `x-low` through `x-high` represent increasing pitch levels. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-IE">
   <prosody pitch="x-high">
        When it hurts, I speak in a high-pitched voice!
```

```
    </prosody>
</speak>
```

The `range` attribute enables the pitch variability for the contained text of `<prosody>` to be altered. The values for the `range` attribute are the same as for the `pitch` attribute. Increasing the value for `range` will increase the dynamic range of the output pitch. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-IE">
    <prosody range="+100%">
         Life is full of ups and downs.
    </prosody>
</speak>
```

Finally, the `contour` attribute provides more advanced control of pitch. The `contour` attribute enables the application author to state the target pitch at different time positions for the contained text of the `<prosody>` element. The format for the `contour` attribute value is a space-delimited list of `(time-position, pitch-target)` pairs. The format for the `time-position` is a percentage ranging from 0 % to 100 %. The format for the `pitch-target` is the same as for the `pitch` attribute on `<prosody>`. The `pitch-target` value is taken relative to the pitch just before the `<prosody>` element. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-IE">
    <prosody contour="(0%,+10Hz) (50%,+50Hz) (100%,-50Hz)">
         Life is full of ups and downs.
    </prosody>
</speak>
```

8.6 Markers

SSML provides the `<mark>` element to allow markers to be inserted into the markup. The `<mark>` element has one required attribute: `name`. The `name` attribute specifies the name of the mark. In the context of MRCP, when the rendered audio output of the speech synthesiser resource encounters a `<mark>` element, the `SPEECH-MARKER` event is sent to the MRCP client with the `name` of the mark inserted in the `Speech-Marker` header field (see Chapter 12 for more information). Markers can be used to determine how much of the output actually gets rendered to the user when barge-in is activated. Mark information is useful to decide if an advertisement or important notice was played, or to mark a place in the content to resume from later, etc. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-GB">
```

```
    <mark name="before-ad"/>
    <audio src="advertisement.wav"/>
    <mark name="after-ad"/>
    Please select from the following options.
</speak>
```

8.7 Metadata

SSML[2] provides two container elements for metadata: the <meta> element and the <metadata> element. The <meta> element is an older mechanism while the <metadata> element provides a newer, more general and powerful mechanism for specifying metadata. It is likely that a future version of SSML will only support the <metadata> approach.

The <meta> element has three attributes: name, http-equiv and content. One of name or http-equiv is required. The content attribute is always present. The only defined name value in SSML is seeAlso – the corresponding content attribute value specifies a resource (i.e. a URI) that provides additional information about the content (SSML does not specify what format that additional information is in, however). The http-equiv attribute, on the other hand, is used to specify HTTP response header information. This is intended to be useful in cases where the author is unable to configure the HTTP header fields on their Web server. Note that if an application author specifies HTTP header field information inside an SSML document, there is no guarantee (in fact it is very unlikely) that the Web server (or intermediate proxy servers) will pick up that information and set the HTTP header fields themselves. This approach of specifying protocol information in the data markup is a legacy from HTML and not recommended for use in practice. An example of an SSML document using the <meta> element is illustrated below:

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-GB">
    <meta http-equiv="Cache-Control" content="max-age=0"/>
    <meta name="seeAlso"
          content="http://www.example.com/metadata.xml"/>
    Welcome to online banking.
</speak>
```

The <metadata> element is a container element where information about the document may be placed. That information is represented in XML syntax by using a different namespace to that of the rest of the SSML document. The SSML specification recommends the XML syntax of the Resource Description Framework (RDF) [63] be used in conjunction with the general metadata properties defined in the Dublin Core Metadata Initiative [64]. RDF provides a standard way to represent metadata in the form of statements and relationships. The Dublin Core Metadata Initiative provides generally applicable metadata properties such as title, subject, creator, etc. An SSML document using RDF and Dublin Core Metadata is illustrated below:

[2] Several other W3C specifications use the same metadata approach, including SRGS (described in Chapter 9), PLS (described in Chapter 11), and VoiceXML (described in Chapter 16).

```
<?xml version="1.0" encoding="UTF-8"?>
<speak version="1.0"
       xmlns="http://www.w3.org/2001/10/synthesis"
       xml:lang="en-GB">
    <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/">
         <rdf:Description
              rdf:about="http://www.example.com/meta.xml"
              dc:Title="Online banking welcome"
              dc:Description="Provides a welcome message"
              dc:Publisher="D Burke"
              dc:Language="en-US"
              dc:Date="2006-02-02"
              dc:Rights="Copyright (c) D Burke"
              dc:Format="application/ssml+xml" >
              <dc:Creator>
                   <rdf:Seq ID="CreatorsAlphabeticalBySurname">
                      <rdf:li>Dave Burke</rdf:li>
                   </rdf:Seq>
                </dc:Creator>
              </rdf:Description>
    </rdf:RDF>
    </metadata>

           Welcome to online banking.
</speak>
```

8.8 Summary

This chapter covered the main features of SSML. SSML is a simple yet flexible markup language designed to enable application authors control aspects of spoken output from speech synthesisers in a standard way. MRCP-compliant speech synthesisers are required to support SSML. Speech application authors using VoiceXML also employ SSML to express spoken and recorded audio to playback to the user.

Chapter 9, next, discusses a markup language that is used for expressing the permissible set of words and phrases from which a speech recogniser can recognise.