INTERNATIONAL STANDARD

ISO/IEC 15938-4

First edition
2002-06-15

Information technology — Multimedia content description interface —

Part 4: Audio

Technologies de l'information — Interface de description du contenu multimédia —

Partie 4: Audio

Reference number
ISO/IEC 15938-4:2002(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 15938-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 15938 consists of the following parts, under the general title *Information technology — Multimedia content description interface*:

— *Part 1: Systems*

— *Part 2: Description definition language*

— *Part 3: Visual*

— *Part 4: Audio*

— *Part 5: Multimedia description schemes*

— *Part 6: Reference software*

— *Part 7: Conformance testing*

— *Part 8: Extraction and use of MPEG-7 descriptions*

Annexes A and B of this part of ISO/IEC 15938 are for information only.

Introduction

This standard, also known as "Multimedia Content Description Interface," provides a standardized set of technologies for describing multimedia content. The standard addresses a broad spectrum of multimedia applications and requirements by providing a metadata system for describing the features of multimedia content.

The following are specified in this standard:

— **Description Schemes (DS)** describe entities or relationships pertaining to multimedia content. Description Schemes specify the structure and semantics of their components, which may be Description Schemes, Descriptors, or datatypes.

— **Descriptors (D)** describe features, attributes, or groups of attributes of multimedia content.

— **Datatypes** are the basic reusable datatypes employed by Description Schemes and Descriptors

— **Description Definition Language (DDL)** defines Description Schemes, Descriptors, and Datatypes by specifying their syntax, and allows their extension.

— **Systems tools** support delivery of descriptions, multiplexing of descriptions with multimedia content, synchronization, file format, and so forth.

This standard is subdivided into eight parts:

Part 1 – Systems: specifies the tools for preparing descriptions for efficient transport and storage, compressing descriptions, and allowing synchronization between content and descriptions.

Part 2 – Description definition language: specifies the language for defining the standard set of description tools (DSs, Ds, and datatypes) and for defining new description tools.

Part 3 – Visual: specifies the description tools pertaining to visual content.

Part 4 – Audio: specifies the description tools pertaining to audio content.

Part 5 – Multimedia description schemes: specifies the generic description tools pertaining to multimedia including audio and visual content.

Part 6 – Reference software: provides a software implementation of the standard.

Part 7 – Conformance testing: specifies the guidelines and procedures for testing conformance of implementations of the standard.

Part 8 – Extraction and use of MPEG-7 descriptions: provides guidelines and examples of the extraction and use of descriptions.

Information technology — Multimedia content description interface —

Part 4: Audio

1 Scope

1.1 Definition of Scope

This International Standard defines a Multimedia Content Description Interface, specifying a series of interfaces from system to application level to allow disparate systems to interchange information about multimedia content. It describes the architecture for systems, a language for extensions and specific applications, description tools in the audio and visual domains, as well as tools that are not specific to audio-visual domains. As a whole, this International Standard encompassing all of the aforementioned components is known as “MPEG-7.” MPEG-7 is divided into eight parts (as defined in the Foreword).

This part of the MPEG-7 Standard (Part 4: Audio) specifies description tools that pertain to multimedia in the audio domain. See below for further details of application.

This part of the MPEG-7 Standard is intended to be implemented in conjunction with other parts of the standard. In particular, MPEG-7 Part 4: Audio assumes knowledge of Part 2: Description Definition Language (DDL) in its normative syntactic definitions of Descriptors and Description Schemes. This part of the standard also has dependencies upon clauses in Part 5: Multimedia Description Schemes, namely many of the fundamental Description Schemes that extend the basic type capabilities of the DDL.

MPEG-7 is an extensible standard. The method to extend the standard beyond the Description Schemes provided in the standard is to define new ones in the DDL, and to make those DSs available with the instantiated descriptions. Further details are available in Part 2. To avoid duplicate functionality with other parts of the standard, the DDL is the only extension facility provided.

1.2 Fields of application

MPEG-7 Part 4: Audio is applicable to all forms of audio content. The encoding format or medium of the said audio is not limited in any way, and may include audio held in an analogue medium such as magnetic tape or optical film. The content of the audio is not limited within or without music, speech, sound effects, soundtracks, or any mixtures thereof.

The tools listed in this part of the International Standard are applicable to both audio in isolation and to audio associated with video.

The specific tools provided within the Audio portion of the standard are designed to work in conjunction with the Multimedia Description Schemes that apply to both audio and video. Because of the “toolbox” nature of the standard, the most appropriate tools from the different parts of the standard may be mixed, within the constraints of the DDL.

The MPEG-7 Audio tools are applicable to two general areas: low-level audio description, in the case of the Audio Framework (clause 5), and application-driven description, in the case of the High Level Tools (clause 6).

The Audio Framework tools are applicable to general audio, without regard to the specific content carried by the encoded signal. The Scalable Series provides general capabilities for multi-level sampled data. The Audio Description Framework defines specific descriptors for use with the Scalable Series or with Audio Segments, which has properties inherited from the general Segment described in the Multimedia Description Schemes part of the standard. The Silence Descriptor works with the Segment descriptor, and is applicable across all possible audio signals.

The high level description tools are applicable to specific types of content within audio. The specific domains are well documented within the introduction to each sub-clause. The audio domains encompassed by the various MPEG-7 Audio tools are speech, sound effects, musical instruments, melodies within music and general audio recognition. These specialised tools may be employed in conjunction with the other tools within the standard.

2 Terms and definitions

For the purposes of this part of ISO/IEC 15938, the following terms and definitions apply.

2.1
Frame
A Frame is defined as a short period of time of the signal on which the instantaneous analysis if perform. For a signal, noted $s(t)$ (in continuous time noted t), and for an analysis window of type hamming, noted $h(t)$ and of temporal length L, the f^{th} signal frame is defined as

$$x(f,t) = s(t) \times h(t - f \times S)$$

where S is the hop size

2.2
Hop size
The hop size defines the temporal distance between two successive analyses

2.3
Running window analysis
A running window analysis is an analysis obtained by multiplying the signal by a window function which is shifted along time by integer multiple of a parameter called the hop size. For a window function $h(t)$, and a hop size S, the f^{th} shifting of the window is equal to $h(t - fS)$.

2.4
Instantaneous values
The instantaneous value of a (Timbre) descriptor based peak estimation is defined to be the result of analysis on a frame level. The global value of a (Timbre) descriptor based on peak estimation is defined to be the average over all frames of the segment of the instantaneous value.

3 Symbols and abbreviated terms

— ASR Automatic Speech Recognition

— CPU Central Processing Unit

— D Descriptor

— DC Direct Current (0 Hz)

— DDL Description Definition Language

— DFT Discrete Fourier Transform

— DS Description Scheme

— FFT Fast Fourier Transform

— HMM Hidden Markov Model

— Hz Hertz, frequency in cycles per second

— LLD Low Level Descriptor

— log Logarithm (unspecified base)

— LPC Linear Predictive Coding

— MSD Maximum Squared Distance (from the mean)

— OOV Out of Vocabulary, describing a word that is not in the vocabulary of an automatic speech recogniser

— RMS Root Mean Square

— SR Sample Rate

— STFT Short Time Fourier Transform

— XML Extensible Markup Language

4 Conventions

4.1 Description Definition Language

All DDL in this document is defined in a single namespace. The schema wrapper is assumed to begin

```
<schema targetNamespace="urn:mpeg:mpeg7:schema:2001"
        xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:xml="http://www.w3.org/XML/1998/namespace"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
```

and end

```
</schema>
```

Under this definition, the default namespace in a schema definition document is specified as XML Schema and thus a prefix `xsd:` is not needed. Instead, references to the element and types defined in the MPEG-7 schema must be qualified with `mpeg7:` prefix. For example,

```
<complexType name="MyElementType">
  <sequence>
    <element name="MyVector" type="mpeg7:MyVectorType"/>
  </sequence>
  <attribute name="myAttribute" type="mpeg7:unsigned8"/>
</complexType>
```

4.2 Audio representation

Within the scope of this standard, the samples of the described audio signals are interpreted as two's complement fractional numbers (i.e. numbers between –1, inclusive, and +1, exclusive), where the Most Significant Bit (MSB) represents the value -1.

5 Audio Framework

5.1 Introduction

The Audio Framework contains low level tools designed to provide a basis for construction of higher level audio applications.

There are essentially two ways of describing low-level audio features. One may sample values at regular intervals or one may use `AudioSegment`s to demark regions of similarity and dissimilarity within the sound. Both of these possibilities are embodied in the low-level descriptor types, `AudioLLDScalarType` and `AudioLLDVectorType`. A descriptor of either of these types may be instantiated as sampled values in a `ScalableSeries`, or as a summary descriptor within an `AudioSegment`. `AudioSegment`, which is a concept that permeates the MPEG-7 Audio standard, is specified in ISO/IEC 15938 Part 5, Multimedia Description Schemes, but we also give a brief overview here.

An `AudioSegment` is a temporal interval of audio material, which may range from arbitrarily short intervals to the entire audio portion of a media document. A required element of an `AudioSegment` is a `MediaTime` descriptor that denotes the beginning and end of the segment. The `TemporalMask` DS is a construct that allows one to specify a temporally non-contiguous `AudioSegment`. An `AudioSegment` (as with any `SegmentType`) may be decomposed hierarchically to describe a tree of `Segment`s.

Another key concept is in the abstract datatypes: `AudioDType` and `AudioDSType`. In order for an audio descriptor or description scheme to be attached to a segment, it must inherit from one of these two types. They are defined in ISO/IEC 15938 part 5. The relationship between these types is shown in Figure 1.

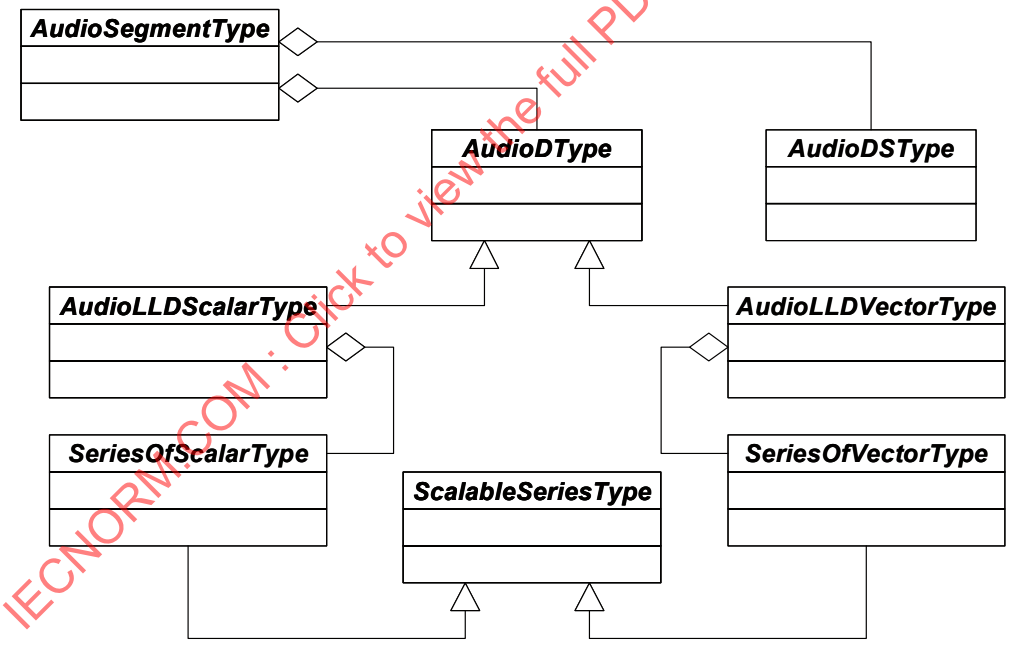

Figure 1 — Illustration of the various structural types in the Audio Framework

5.2 Scalable Series

5.2.1 Introduction

Scalable series are datatypes for series of values (scalars or vectors). They allow the series to be scaled (downsampled) in a well-defined fashion. Two types are available: `SeriesOfScalarType` and `SeriesOfVectorType`. They are useful in particular to build descriptors that contain *time series* of values.

5.2.2 ScalableSeriesType

This is an abstract type inherited by `SeriesOfScalarType` and `SeriesOfVectorType`. Its attributes define the dimensions and scaling ratio of the series.

5.2.2.1 Syntax

```
<!-- ############################################################################ -->
<!-- Definition of ScalableSeries datatype                                      -->
<!-- ############################################################################ -->
<complexType name="ScalableSeriesType" abstract="true">
  <sequence>
    <element name="Scaling" minOccurs="0" maxOccurs="unbounded">
      <complexType>
        <attribute name="ratio" type="positiveInteger" use="required"/>
        <attribute name="numOfElements" type="positiveInteger"
                   use="required"/>
      </complexType>
    </element>
  </sequence>
  <attribute name="totalNumOfSamples" type="positiveInteger" use="required"/>
</complexType>
```

5.2.2.2 Semantics

Name	Definition
`ScalableSeriesType`	An abstract type representing series of values, at full resolution or after scaling (downsampling) by a scaling operation. In the latter case the series contains sequences that have been concatenated together. Within each sequence, the elements share the same scale ratio.
`Scaling`	To specify how the original samples are scaled. If absent, the original samples are described without scaling.
`ratio`	Scale ratio (number of original samples represented by each scaled sample) common to all elements in a sequence. The value to be used when `Scaling` is absent is 1.
`numOfElements`	Number of scaled elements in a sequence. The value to be used when `Scaling` is absent is equal to the value of `totalNumOfSamples`.
`totalNumOfSamples`	Total number of samples of the original series (before scaling).

Note that the last sample of the series may summarize fewer than `ratio` samples. This happens if `totalNumOfSamples` is smaller than the sum over runs of the product of `numOfElements` by `ratio`. An illustration of the Scalable Series is shown in Figure 2, where 'k' is an index in the scaled series. In this figure, the 31 samples of the original series (filled circles) are summarized by 13 samples of the scaled series (open circles). The first three scaled samples each summarizes two original samples, the next two six, the next two one, etc. The last scaled sample has nominally a ratio of two, but actually summarizes only one original sample. This situation is legal, and detected by comparing the sum of `ratio` times `numOfElements` products to `totalNumOfSamples`.

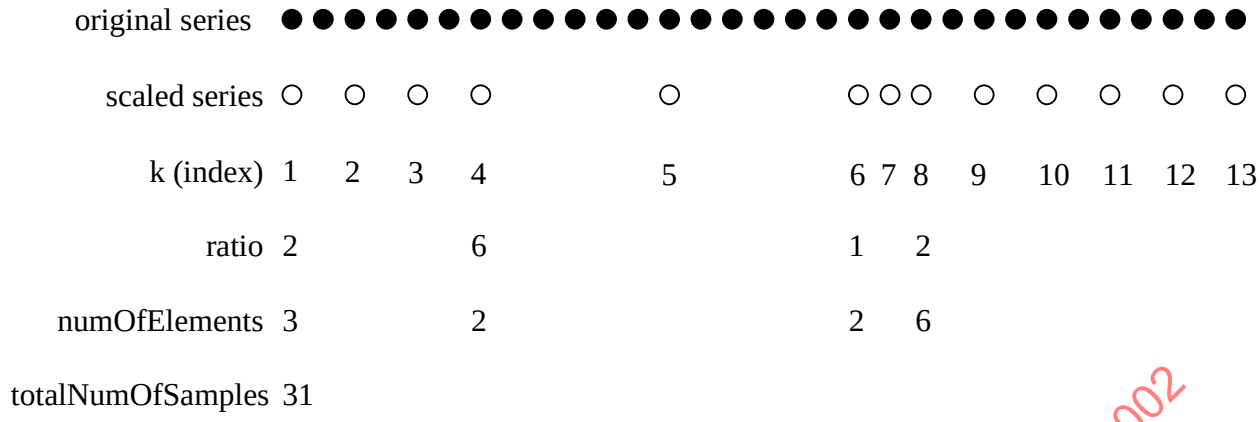

Figure 2 — An illustration of the scalable series

5.2.3 SeriesOfScalarType

This descriptor represents a series of scalars, at full resolution or scaled. Use this type within descriptor definitions to represent a series of feature values.

5.2.3.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SeriesOfScalar datatype                            -->
<!-- ################################################################ -->
<complexType name="SeriesOfScalarType">
  <complexContent>
    <extension base="mpeg7:ScalableSeriesType">
      <sequence>
        <element name="Raw" type="mpeg7:floatVector" minOccurs="0"/>
        <element name="Min" type="mpeg7:floatVector" minOccurs="0"/>
        <element name="Max" type="mpeg7:floatVector" minOccurs="0"/>
        <element name="Mean" type="mpeg7:floatVector" minOccurs="0"/>
        <element name="Random" type="mpeg7:floatVector" minOccurs="0"/>
        <element name="First" type="mpeg7:floatVector" minOccurs="0"/>
        <element name="Last" type="mpeg7:floatVector" minOccurs="0"/>
        <element name="Variance" type="mpeg7:floatVector" minOccurs="0"/>
        <element name="Weight" type="mpeg7:floatVector" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.2.3.2 Semantics

Name	Definition
`SeriesOfScalarType`	A representation of a series of scalar values of a feature.
`Raw`	Series of unscaled samples (full resolution). Use only if scaling is absent to indicate the entire series.
`Min`	Series of minima of groups of samples. The value of `numOfElements` shall equal the length of the vector. This element shall be absent or empty if the `Raw` element is present.

Name	Definition
Max	Series of maxima of groups of samples. The value of numOfElements shall equal the length of the vector. This element shall be absent or empty if the Raw element is present.
Mean	Series of means of groups of samples. The value of numOfElements shall equal the length of the vector. This element shall be absent or empty if the Raw element is present.
Random	Downsampled series (one sample selected at random from each group of samples). The value of numOfElements shall equal the length of the vector. This element shall be absent or empty if the Raw element is present.
First	Downsampled series (first sample selected from each group of samples). The value of numOfElements shall equal the length of the vector. This element shall be absent or empty if the Raw element is present.
Last	Downsampled series (last sample selected from each group of samples). The value of numOfElements shall equal the length of the vector. This element shall be absent or empty if the Raw element is present.
Variance	Series of variances of groups of samples. The value of numOfElements shall equal the length of the vector. This element shall be absent or empty if the Raw element is present. Mean must be present in order for Variance to be present.
Weight	Optional series of weights. Contrary to other fields, these do not represent values of the descriptor itself, but rather auxiliary weights to control scaling (see below). The value of numOfElements shall equal the length of the vector.

Note: Data of a full resolution series (ratio = 1) are stored in the Raw field. Accompanying zero-sized fields (such as Mean) indicate how the series may be scaled, if the need for scaling arises. The data are then stored in the scaled field(s) and the Raw field disappears.

Scalable Series allow data to be stored at reduced resolution, according to a number of possible scaling operations. The allowable operations are those that are *scalable* in the following sense. Suppose the original series is scaled by a scale ratio of P, and this scaled series is then rescaled by a factor of Q. The result is the same as if the original series had been scaled by a scale ratio of $N = PQ$.

Figure 3 illustrates the scalability property. This scaled series can be derived indifferently from the original series by applying the scaling operation with the ratios shown, or from the scaled Series of Figure 2 by applying the appropriate rescaling operation. The result is identical. Scaling operations are chosen among those for which this property can be enforced.

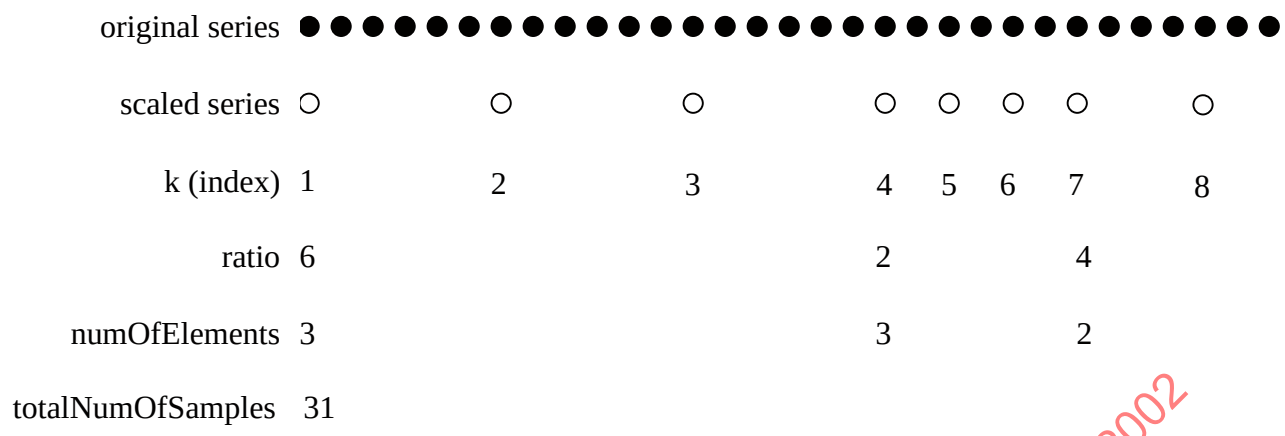

Figure 3 — An illustration of the scalability property

If the scaling operations are used, they shall be computed as follows.

Name	Definition	Definition if Weight present
Min	$m_k = \min_{i=1+(k-1)N}^{kN} x_i$	Ignore samples with zero weight. If all have zero weight, set to zero by convention.
Max	$M_k = \max_{i=1+(k-1)N}^{kN} x_i$	Ignore samples with zero weight. If all have zero weight, set to zero by convention.
Mean	$\bar{x}_k = (1/N) \sum_{i=1+(k-1)N}^{kN} x_i$	$\bar{x}_k = \sum_{i=1+(k-1)N}^{kN} w_i x_i \Big/ \sum_{i=1+(k-1)N}^{kN} w_i$ If all samples have zero weight, set to zero by convention.
Random	choose at random among N samples	Choose at random with probabilities proportional to weights. If all samples have zero weight, set to zero by convention.
First	choose the first of N samples	Choose first non-zero-weight sample. If all samples have zero weight, set to zero by convention.
Last	choose the last of N samples	Choose last non-zero-weight sample. If all samples have zero weight, set to zero by convention.
Variance	$z_k = (1/N) \sum_{i=1+(k-1)N}^{kN} (x_i - \bar{x}_k)^2$ $= (1/N) \sum_{i=1+(k-1)N}^{kN} x_i^2 - \bar{x}_k^2$	$z_k = \sum_{i=1+(k-1)N}^{kN} w_i (x_i - \bar{x}_k)^2 \Big/ \sum_{i=1+(k-1)N}^{kN} w_i$ If all samples have zero weight, set to zero by convention.
Weight	$\bar{w}_k = (1/N) \sum_{i=1+(k-1)N}^{kN} w_i$	

In these formulae, k is an index in the scaled series, and i an index in the original series. N is the number of samples summarized by each scaled sample. The formula for `Variance` differs from the standard formula for unbiased variance by the presence of N rather than $N-1$. Unbiased variance is easy to derive from it. If the `Weight` field is present, the terms of all sums are weighted.

5.2.4 SeriesOfScalarBinaryType

Use this type to instantiate a series of scalars with a uniform power-of-two ratio. The restriction to a power-of-two ratio eases the comparison of series with different `ratio`s as the decimation required for the comparison will also be a power of 2. Such decimation allows perfect comparison. It also allows an additional scaling operation to be defined (scalewise variance). Considering these computational properties of power-of-two scale ratios, the `SeriesOfScalarBinaryType` is the most useful of the Scalable Series family.

Note that the types `SeriesOfScalarBinaryType` and `SeriesOfVectorBinaryType` inherit from the appropriate non-binary type. This means that although they are not used directly in this document, they can be used in place of the non-binary type at any time.

Figure 4 shows an illustration of `SeriesOfScalarBinaryType`. In this subtype of `SeriesOfScalarType`, all `ratio`s must all be powers of two, and all `numOfElements` values must be such that the series may be rescaled to the largest `ratio` that it contains. In this example, the largest `ratio` is 8. The first four samples (`ratio` 2) can be rescaled to a single sample with `ratio` 8, and the last four samples can be rescaled to two samples with `ratio` 8. Note, as in previous illustrations, that the last scaled sample summarizes fewer original samples than its nominal `ratio`.

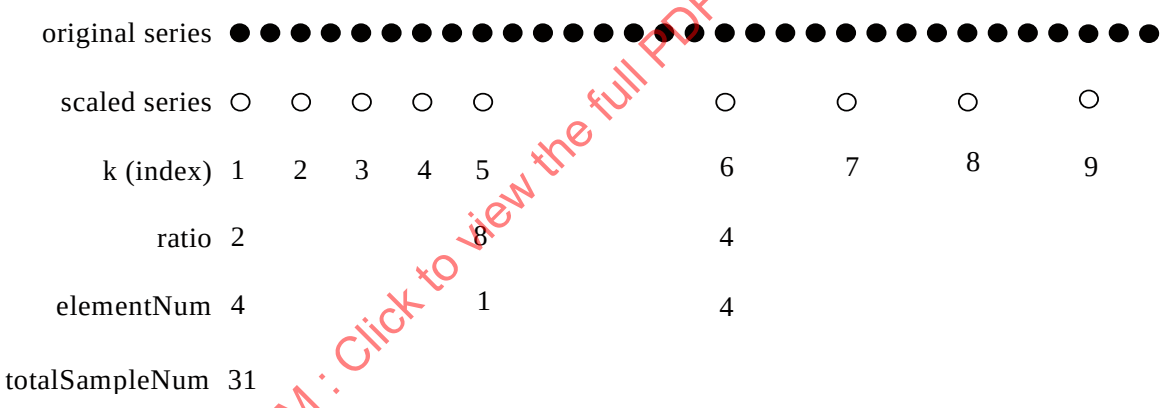

Figure 4 — An illustration of SeriesOfScalarBinaryType

5.2.4.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SeriesOfScalarBinary datatype                      -->
<!-- ################################################################ -->
<complexType name="SeriesOfScalarBinaryType">
  <complexContent>
    <extension base="mpeg7:SeriesOfScalarType">
      <sequence>
        <element name="VarianceScalewise" type="mpeg7:FloatMatrixType"
                 minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.2.4.2 Semantics

Name	Definition
`SeriesOfScalarBinaryType`	A representation of a series of scalar values scaled by a power of two factor.
`VarianceScalewise`	Optional array of arrays of scalewise variance coefficients. Scalewise variance is a decomposition of the variance into a series of coefficients, each of which describes the variability at a particular scale. There are log2(`ratio`) such coefficients. See definition below. Number of rows must equal `numOfElements`, number of columns must equal the number of coefficients of the scalewise variance.

5.2.4.3 Usage, extraction and examples

5.2.4.3.1 Scalewise Variance

Scalewise variance is a decomposition of the variance into a vector of coefficients that describe variability at different scales. The sum of these coefficients equals the variance. To calculate the scalewise variance of a set of $N = 2^m$ samples, first recursively form a binary tree of means:

$$\bar{x}_k^1 = (x_{2k-1} + x_{2k})/2,\ k = 1,\ldots N/2$$

$$\bar{x}_k^2 = (\bar{x}_{2k-1}^1 + \bar{x}_{2k}^1)/2,\ k = 1,\ldots N/4$$

…

$$\bar{x}_k^m = (\bar{x}_{2k-1}^{m-1} + \bar{x}_{2k}^{m-1})/2,\ k = 1$$

where x is a sample. Then calculate the coefficients z:

$$z^1 = (2/N)\sum_{k=1}^{N/2}(x_{2k-1} - x_{2k})^2/2$$

$$z^2 = (4/N)\sum_{k=1}^{N/4}(\bar{x}_{2k-1}^1 - \bar{x}_{2k}^1)^2/2$$

…

$$z^m = (\bar{x}_{2k-1}^{m-1} - \bar{x}_{2k}^{m-1})^2/2$$

The vector formed by these coefficients is the scalewise variance for this group of samples. The `VarianceScalewise` field stores a series of such vectors.

5.2.5 SeriesOfVectorType

This descriptor represents a series of vectors.

5.2.5.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SeriesOfVector datatype                            -->
<!-- ################################################################ -->
<complexType name="SeriesOfVectorType">
  <complexContent>
    <extension base="mpeg7:ScalableSeriesType">
      <sequence>
        <element name="Raw" type="mpeg7:FloatMatrixType" minOccurs="0"/>
        <element name="Min" type="mpeg7:FloatMatrixType" minOccurs="0"/>
        <element name="Max" type="mpeg7:FloatMatrixType" minOccurs="0"/>
        <element name="Mean" type="mpeg7:FloatMatrixType" minOccurs="0"/>
        <element name="Random" type="mpeg7:FloatMatrixType" minOccurs="0"/>
        <element name="First" type="mpeg7:FloatMatrixType" minOccurs="0"/>
        <element name="Last" type="mpeg7:FloatMatrixType" minOccurs="0"/>
        <element name="Variance" type="mpeg7:FloatMatrixType" minOccurs="0"/>
        <element name="Covariance" type="mpeg7:FloatMatrixType"
                 minOccurs="0"/>
        <element name="VarianceSummed" type="mpeg7:floatVector"
                 minOccurs="0"/>
        <element name="MaxSqDist" type="mpeg7:floatVector" minOccurs="0"/>
        <element name="Weight" type="mpeg7:floatVector" minOccurs="0"/>
      </sequence>
      <attribute name="vectorSize" type="positiveInteger" default="1"/>
    </extension>
  </complexContent>
</complexType>
```

5.2.5.2 Semantics

Name	Definition
SeriesOfVectorType	A type for scaled series of vectors.
Raw	Series of unscaled samples (full resolution). Use only if ratio=1 for the entire series.
Min	Series of minima of groups of samples. Number of rows must equal numOfElements, number of columns must equal vectorSize. This element must be absent or empty if the element Raw is present.
Max	Series of maxima of groups of samples. Number of rows must equal numOfElements, number of columns must equal vectorSize. This element must be absent or empty if the element Raw is present.
Mean	Series of means of groups of samples. Number of rows must equal numOfElements, number of columns must equal vectorSize. This element must be absent or empty if the element Raw is present.
Random	Downsampled series (one sample selected at random from each group of samples). Number of rows must equal numOfElements, number of columns must equal vectorSize. This element must be absent or empty if the element Raw is present.
First	Downsampled series (first sample selected from each group of samples). Number of rows must equal numOfElements, number of columns must equal vectorSize. This element must be absent or empty if the element Raw is present.

Name	Definition
`Last`	Downsampled series (last sample selected from each group of samples). Number of rows must equal `numOfElements`, number of columns must equal `vectorSize`. This element must be absent or empty if the element `Raw` is present.
`Variance`	Series of variance vectors of groups of vector samples. Number of rows must equal `numOfElements`, number of columns must equal `vectorSize`. This element must be absent or empty if the element `Raw` is present. `Mean` must be present in order for `Variance` to be present.
`Covariance`	Series of covariance matrices of groups of vector samples. This is a three-dimensional matrix. Number of rows must equal `numOfElements`, number of columns and number of pages must both equal `vectorSize`. This element must be absent or empty if the element `Raw` is present. `Mean` must be present in order for `Covariance` to be present.
`VarianceSummed`	Series of summed variance coefficients of groups of samples. Size of the vector must equal `numOfElements`. This element must be absent or empty if the element `Raw` is present. `Mean` must be present in order for `VarianceSummed` to be present.
`MaxSqDist`	Maximum Squared Distance (MSD). Series of coefficients representing an upper bound of the distance between groups of samples and their mean. Size of array must equal `numOfElements`. This element must be absent or empty if the element `Raw` is present. If `MaxSqDist` is present, `Mean` must also be present.
`Weight`	Optional series of weights. Weights control downsampling of other fields (see explanation for `SeriesOfScalars`). Size of array must equal `numOfElements`.
`vectorSize`	The number of elements of each vector within the series.

Most of the above operations are straightforward extensions of operations previously defined in section 5.2.3.2 for series of scalars, applied uniformly to each dimension of the vectors. Operations that are specific to vectors are defined here:

Name	Definition	Definition if `Weight` present
`Covariance`	$\sigma_k^{jj'} = \frac{1}{N}\sum_{i=1+(k-1)N}^{kN}(x_i^j - \bar{x}^j)(x_i^{j'} - \bar{x}^{j'})$	$\sigma_k^{jj'} = \sum_{i=1+(k-1)N}^{kN} w_i\,(x_i^j - \bar{x}^j)(x_i^{j'} - \bar{x}^{j'}) \Big/ \sum_{i=1+(k-1)N}^{kN} w_i$
`VarianceSummed`	$z_k = (1/N)\sum_{j=1}^{D}\sum_{i=1+(k-1)N}^{kN}(x_i^j - \bar{x}_i^j)^2$	$z_k = \sum_{j=1}^{D}\sum_{i=1+(k-1)N}^{kN} w_i\,(x_i^j - \bar{x}_i^j)^2 \Big/ \sum_{i=1+(k-1)N}^{kN} w_i$ If all samples have zero weight, set to zero by convention.
`MaxSqDist`	$MSD_k = \max_{i=1+(k-1)N}^{kN}\left\|x_i - \bar{x}_k\right\|^2$	Ignore samples with zero weight. If all samples have zero weight, set to zero by convention

In these formulae, k is an index in the scaled series, and i an index in the original series. N is the number of vectors summarized by each scaled vector. D is the size of each vector and j is an index into each vector. $\bar{x}_i^j$ is the mean of N samples.

The various variance/covariance options offer a choice of several cost/performance tradeoffs for the representation of variability.

5.2.6 SeriesOfVectorBinaryType

Use this type to instantiate a series of vectors with a uniform power-of-two `ratio`. The restriction to a power-of-two ratio eases the comparison of series with different `ratio`s as the decimation necessary for the comparison is simply another power of 2. The use of power-of-two scale ratios is recommended.

5.2.6.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SeriesOfVectorBinary datatype                      -->
<!-- ################################################################ -->
<complexType name="SeriesOfVectorBinaryType">
  <complexContent>
    <extension base="mpeg7:SeriesOfVectorType">
      <sequence>
        <element name="VarianceScalewise" type="mpeg7:FloatMatrixType"
                 minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.2.6.2 Semantics

Name	Definition
`SeriesOfVectorBinaryType`	A representation of a reduced-resolution series of vector samples with a power-of-two `ratio`.
`VarianceScalewise`	Array of arrays of scalewise summed-variance coefficients. Scalewise variance is a decomposition of the variance into a series of coefficients, each of which describes the variability at a particular scale. Number of rows must equal to `numOfElements`, number of columns must equal the number of coefficients of the scalewise variance.

5.3 Low level Audio Descriptors

5.3.1 Introduction

Low-level Audio Descriptors (LLDs) consist of a collection of simple, low complexity descriptors that are designed to be used within the `AudioSegment` framework, see ISO/IEC 15938-5 (E). Whilst being useful in themselves, they also provide examples of a design framework for future extension of the audio descriptors and description schemes.

All low-level audio descriptors are defined as subtypes of either `AudioLLDScalarType` or `AudioLLDVectorType`, except the `AudioSpectrumBasis D`. There are two description strategies using these data types: single-valued summary and sampled series segment description. These two description strategies are made available for the two data types, `Scalar`/`SeriesOfScalarType` and `Vector`/`SeriesOfVectorType`, and are implemented as a `choice` in DDL.

When using summary descriptions (containing a single scalar or vector) there are no normative methods for calculating the single-valued summarization. However, when using series-based descriptions, the summarization

values shall be calculated using the scaling methods provided by the `SeriesOfScalarType` and `SeriesOfVectorType` descriptors, such as the `min`, `max` and `mean` operators.

5.3.2 AudioLLDScalarType

5.3.2.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioLLDScalar D                                   -->
<!-- ################################################################ -->
<complexType name="AudioLLDScalarType" abstract="true">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <choice>
        <element name="Scalar" type="float"/>
        <element name="SeriesOfScalar" minOccurs="1" maxOccurs="unbounded">
          <complexType>
            <complexContent>
              <extension base="mpeg7:SeriesOfScalarType">
                <attribute name="hopSize" type="mpeg7:mediaDurationType"
                           default="PT10N1000F"/>
              </extension>
            </complexContent>
          </complexType>
        </element>
      </choice>
      <attribute name="confidence" type="mpeg7:zeroToOneType" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

5.3.2.2 Semantics

Name	Definition
`AudioLLDScalarType`	Abstract definition inherited by all scalar datatype audio descriptors.
`Scalar`	Value of the descriptor
`SeriesOfScalar`	Scalar values for sampled-series description of an audio segment. Use of this scalable series datatype promotes compatibility between sampled descriptions.
`hopSize`	Time interval between data samples for series description. The default value is `PT10N1000F` which is 10 milliseconds. Values other than the default shall be integer multiples/divisors of 10 milliseconds. This will ensure compatibility of descriptors sampled at different rates.

5.3.3 AudioLLDVectorType

5.3.3.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioLLDVector D                                   -->
<!-- ################################################################ -->
<complexType name="AudioLLDVectorType" abstract="true">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <choice>
        <element name="Vector" type="mpeg7:floatVector"/>
        <element name="SeriesOfVector" minOccurs="1" maxOccurs="unbounded">
          <complexType>
            <complexContent>
              <extension base="mpeg7:SeriesOfVectorType">
                <attribute name="hopSize" type="mpeg7:mediaDurationType"
                           default="PT10N1000F"/>
              </extension>
            </complexContent>
          </complexType>
        </element>
      </choice>
    </extension>
  </complexContent>
</complexType>
```

5.3.3.2 Semantics

Name	Definition
`AudioLLDVectorType`	Abstract definition inherited by all vector datatype audio descriptors.
`Vector`	Vector value of descriptor
`SeriesOfVector`	Vector values for sampled-series description of an audio segment. Use of this scalable series datatype promotes compatibility between sampled descriptions.
`hopSize`	Time interval between data samples for series description. The default value is `PT10N1000F` which is 10 milliseconds. Values other than the default shall be integer multiples/divisors of 10 milliseconds. This will ensure compatibility of descriptors sampled at different rates.

5.3.3.3 Usage and Extraction

Audio descriptors that calculated at regular intervals (sample period or frame period) shall use the `hopSize` field to specify the extraction period. In all cases, the `hopSize` shall be a positive integer multiple/divisor of the default 10 millisecond sampling period. Note that downsampling by means of the scalable series does not change the specified `hopSize` but instead specifies the downsampling scale ratio to be used together with the `hopSize`.

`AudioLLDScalarType` and `AudioLLDVectorType` are both abstract and therefore never instantiated.

5.3.4 AudioWaveformType

`AudioWaveForm D` describes the audio waveform envelope, typically for display purposes.

5.3.4.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioWaveform D                                    -->
<!-- ################################################################ -->
<complexType name="AudioWaveformType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType">
      <attribute name="minRange" type="float" use="optional"/>
      <attribute name="maxRange" type="float" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

5.3.4.2 Semantics

Name	Definition
`AudioWaveformType`	Description of the waveform of the audio signal.
`minRange`	Lower limit of audio signal amplitude
`maxRange`	Upper limit of audio signal amplitude

5.3.4.3 Usage and extraction

5.3.4.3.1 Purpose

`AudioWaveform D` allows economical display of an audio waveform. For example, a sound editing application program can display a summary of an entire audio file immediately without processing the audio data and data may be displayed and edited over a network, etc. Whatever the number of samples, the waveform may be displayed using a small set of values that represent extrema (min and max) of frames of samples. Min and max are stored as scalable time series within the `AudioWaveform D`. They may also be used for fast comparison between waveforms.

5.3.4.3.2 To create a description

a) Instantiate an `AudioWaveformType` with `hopSize` set to the required temporal resolution (default 10ms), or if the raw signal is desired set `hopSize` to 1/(sampling rate).

b) Determine the equivalent number of samples, ns, in each frame (ns = sampling rate x hop size).

c) If ns = 1 then instantiate the `Raw` and store the raw data. Instantiate empty `Min` and `Max` fields.

d) Otherwise take the minimum and maximum value in each `hopSize` frame. Instantiate `Min` and `Max` fields and fill them with scaled samples.

5.3.4.3.3 To use a description for display

a) Read the arrays of `min` and `max` values from the `AudioWaveformType`.

b) For each `min`/`max` pair, draw a vertical line from `min` to `max`.

c) Label axes using the `hopSize` and scaling information.

5.3.5 AudioPowerType

AudioPower D describes the temporally-smoothed instantaneous power (square of waveform values).

5.3.5.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioPower D                                       -->
<!-- ################################################################ -->
<complexType name="AudioPowerType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType"/>
  </complexContent>
</complexType>
```

5.3.5.2 Semantics

Name	Definition
`AudioPowerType`	Description of the power of the audio signal.

5.3.5.3 Usage and extraction

5.3.5.3.1 Extraction

Instantaneous power is calculated by taking the square of waveform samples. These are averaged over time intervals of length corresponding to `hopSize` and stored in the `Mean` field of a `SeriesOfScalarType`.

5.3.5.3.2 Purpose

Instantaneous power is a useful measure of the amplitude of a signal as a function of time, $P(t)=|s(t)|^2$. In association with `AudioSpectrumCentroid D` and `AudioSpectrumSpread D`, the `AudioPower D` provides an economical description of the power spectrum (spreading the power over the spectral range specified by the centroid and spread) that can be compared with a log-frequency spectrum. Another possibility is to store instantaneous power at high temporal resolution, in association with a high spectral resolution power spectrum at low temporal resolution, to obtain a cheap representation of the power spectrum that combines both spectral and temporal resolution.

5.3.5.3.3 Motivation for the design

Instantaneous power is coherent with the power spectrum. A signal labeled with the former can meaningfully be compared to a signal labeled with the latter. Note however that temporal smoothing operations are not quite the same, so values may differ slightly for identical signals.

5.3.6 Audio Spectrum Descriptors

5.3.6.1 Introduction

The following descriptors (5.3.7 to 5.3.12) are all descriptions of the audio spectrum. They have a central link that they all derive from a time-frequency analysis of the audio signal.

5.3.6.2 AudioSpectrumAttributeGrp

The `AudioSpectrumAttributeGrp` defines a common set of attributes applicable to many of the spectrum descriptions.

5.3.6.2.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of audioSpectrumAttributeGrp                          -->
<!-- ################################################################ -->
<attributeGroup name="audioSpectrumAttributeGrp">
  <!-- Edge values are in Hertz                                       -->
  <attribute name="loEdge" type="float" default="62.5"/>
  <attribute name="hiEdge" type="float" default="16000"/>
  <attribute name="octaveResolution" use="optional">
    <simpleType>
      <restriction base="string">
        <enumeration value="1/16"/>
        <enumeration value="1/8"/>
        <enumeration value="1/4"/>
        <enumeration value="1/2"/>
        <enumeration value="1"/>
        <enumeration value="2"/>
        <enumeration value="4"/>
        <enumeration value="8"/>
      </restriction>
    </simpleType>
  </attribute>
</attributeGroup>
```

5.3.6.2.2 Semantics

Name	Definition
`loEdge`	Lower edge of logarithmically-spaced frequency bands.
`hiEdge`	Higher edge of logarithmically-spaced frequency bands.
`Resolution`	Frequency resolution of logarithmic spectrum (width of each spectrum band between `loEdge` and `hiEdge`).

5.3.7 AudioSpectrumEnvelopeType

`AudioSpectrumEnvelope D` describes the spectrum of the audio according to a logarithmic frequency scale.

5.3.7.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioSpectrumEnvelope D                            -->
<!-- ################################################################ -->
<complexType name="AudioSpectrumEnvelopeType">
  <complexContent>
    <extension base="mpeg7:AudioLLDVectorType">
      <attributeGroup ref="mpeg7:audioSpectrumAttributeGrp"/>
    </extension>
  </complexContent>
</complexType>
```

5.3.7.2 Semantics

Name	Definition
AudioSpectrumEnvelopeType	Description of the power spectrum of the audio signal. The spectrum consists of one coefficient representing power between 0Hz and loEdge, a series of coefficients representing power in resolution sized bands, between loEdge and hiEdge, and a coefficient representing power beyond hiEdge, in this order.

5.3.7.3 Usage and extraction

5.3.7.3.1 Purpose

The AudioSpectrumEnvelope D describes the short-term power spectrum of the audio waveform as a time series of spectra with a logarithmic frequency axis. It may be used to display a spectrogram, to synthesize a crude "auralization" of the data, or as a general-purpose descriptor for search and comparison.

5.3.7.3.2 Motivation for the design

A logarithmic frequency axis is used to reconcile requirements of concision and descriptive power. Peripheral frequency analysis in the ear roughly follows a logarithmic axis.

The power spectrum is used because of its scaling properties (the power spectrum over an interval is equal to the sum of power spectra over subintervals).

5.3.7.3.3 Specification of the descriptor (normative)

These specifications are normative and ensure that descriptions produced by different programs are either directly comparable, or else can easily be converted to comparable descriptions. Importantly, a high-resolution description is readily convertible to a low-resolution description, in both the spectral or temporal dimension.

The AudioSpectrumEnvelope D describes the spectrum over the frequency range between loEdge and hiEdge. The range between loEdge and hiEdge is divided into multiple bands. The resolution, in octaves, of the bands is specified by resolution. Except for when the octaveResolution is 1/8, both loEdge and hiEdge must be related to 1KHz as described in the following equation:

$$edge = 2^{rm} \times 1KHz \qquad (5.3.7.1)$$

where r is the resolution in octaves, $m \in Z$ (i.e., m an integer).

For the case when resolution is "8 octave" the spectrum delivers a single coefficient representing within-band power, and two extra coefficients for below-band and above-band power. In this case the default values for loEdge and hiEdge should be used.

If m_l and m_h are the integers corresponding to Equation 5.3.7.1 when *edge* equals loEdge and hiEdge, respectively, then the full set of band edges are given by $edge = 2^{rm} \times 1KHz$, $m_l \le m \le m_h$.

In every case the spectrum contains two extra values, one representing the energy between 0Hz and loEdge, and the other energy between hiEdge and half the sampling rate (See Figure 5). If hiEdge equals half the sampling rate then the second extra value is set to zero. These values measure the "out-of-band" energy. Default hiEdge is 16000Hz (corresponding to the upper limit of hearing). Default loEdge is 62.5 Hz (8 octaves below hiEdge). The default analysis frame period is 10 ms, which is within the range of estimates for temporal acuity of the ear (8 to 13 ms) and is also the default analysis frame period for sampled audio descriptors.

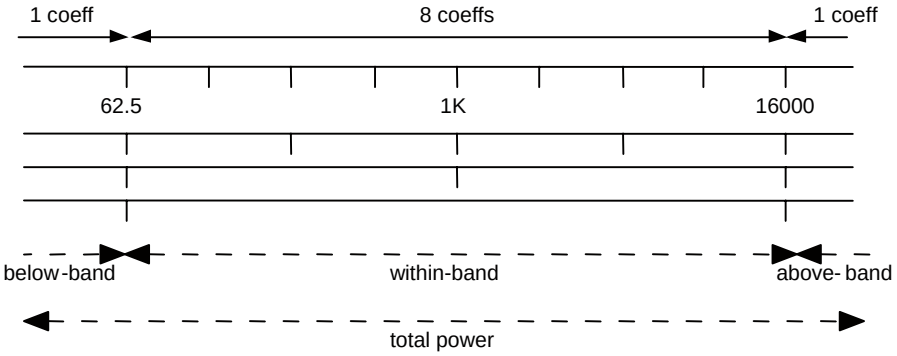

Figure 5 — Illustration of Audio Spectrum Envelope Bands

5.3.7.3.4 **Extraction** (informative)

To extract the AudioSpectrumEnvelope the following method is recommended. The method involves a sliding window FFT analysis, with a resampling to logarithmic spaced bands.

a) Determine the required hop length h, corresponding to the hopsize. If the sampling rate is sr, then h = sr*hopsize (e.g. h = 16000*0.01 = 160 samples). If sr*hopsize is not a whole number of samples then generate a vector h such that mean(h) = sr*hopsize (e.g. sr*hopsize = 22050 * 0.01 = 220.5, h = [220 221]). By cycling through the vector of hop lengths the analysis will not stray over time, but will give minor jitter from the defined hopsize. This enables reasonable comparison of data sampled at differing rates.

b) Determine the analysis window length *lw*. The analysis window has been chosen to have a default value of 3 hopsizes, 30ms. This is to provide enough spectral resolution to roughly resolve the 62.5 Hz-wide first channel of a '1 octave' resolution spectrum.

c) Determine the FFT size, NFFT. NFFT is the next-larger power-of-two number of samples from lw, e.g. If lw = 1323 samples then NFFT would be 2048.

d) Perform a STFT using a Hamming window of length lw, a shift of h samples (where h is a vector, rotate through the vector to prevent stray, and deliver minimal jitter), using a NFFT point FFT, with out-of-window samples set to 0. The descriptor only retains the square magnitude of the FFT coefficients, $|X_w(k)|^2$. The sum of the power spectrum coefficients is equal to the average power in the analysis window, P_w. By Parseval's theorem there is a further factor of 1/NFFT to equate the sum of the squared magnitude of the FFT coefficients to the sum of the squared, zero-padded, windowed signal.

$$P_w = \frac{1}{lw}\sum_{n=0}^{lw-1}|x_w(n)|^2 = \frac{1}{lw * NFFT}\sum_{k=0}^{NFFT-1}|X_w(k)|^2$$

where $x_w(n) = s(n) * w(n), 0 \le n < lw$ and $w(n)$ is the Hamming window of length lw.

Hence $P_x(k) = \frac{1}{lw * NFFT}|X_w(k)|^2$

Since the audio signal is a real signal its Fourier transform has even symmetry. Hence only the spectral coefficients up to the Nyquist frequency need be retained.

e) Resample to a logarithmic scale. Let DF be the frequency spacing of the FFT (DF = sr/NFFT). An FFT coefficient more than DF/2 from a band edge is assigned to the band. A coefficient less than DF/2 from a band edge is proportionally shared between bands, as illustrated in Figure 6.

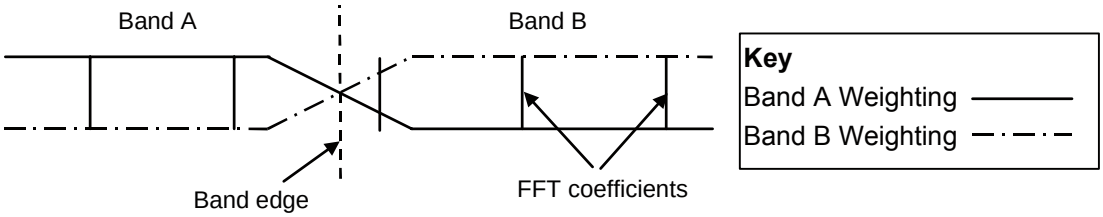

Figure 6 — Weighting Method for linear - log conversion

Important Note: Due to the weighting method illustrated in Figure 6 it is important to select an appropriate `loEdge` at fine frequency resolutions. To be able to resolve the logarithmic bands there needs to be at least one FFT coefficient in each band. In some cases this means that the default `loEdge` is unsuitable. Table 1 indicates the minimum value that `loEdge` should be set to for some popular sampling frequencies, assuming default `hopSize`.

Table 1 — Minimum `loEdge` for particular resolutions (exceeding default 62.5Hz)

Resolution	Minimum `loEdge` (DF = 31.25, FFT size 1024, SR = 32kHz, FFT size 512, SR = 16kHz)	Minimum `loEdge` (DF = 21.53, FFT size 2048, SR = 44.1kHz, FFT size 1024, SR = 22.05kHz)	Minimum `loEdge` (DF = 23.44, FFT size 2048, SR = 48kHz)
1/4 octave	88.388 ($62.5*2^{0.5}$)	62.5	105.1 ($62.5*2^{3/4}$)
1/8 octave	324.21 ($62.5*2^{19/8}$)	176.8 ($62.5*2^{12/8}$)	192.78 ($62.5*2^{13/8}$)
1/16 octave	545.25 ($62.5*2^{25/8}$)	439.1 ($62.5*2^{45/16}$)	478.8 ($62.5*2^{47/16}$)

5.3.8 AudioSpectrumCentroidType

`AudioSpectrumCentroid` D describes the center of gravity of the log-frequency power spectrum. The `SpectrumCentroid` is defined as the power weighted log-frequency centroid.

5.3.8.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioSpectrumCentroid D                            -->
<!-- ################################################################ -->
<complexType name="AudioSpectrumCentroidType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType"/>
  </complexContent>
</complexType>
```

5.3.8.2 Semantics

Name	Definition
AudioSpectrumCentroidType	Description of the center of gravity of the log-frequency power spectrum. Range: -5 to $\log_2$(sr/2000)

5.3.8.3 Usage and extraction

5.3.8.3.1 **Extraction** (informative)

To be coherent with other descriptors, in particular AudioSpectrumEnvelope D, the spectrum centroid is defined as the center-of-gravity of a log-frequency power spectrum. This definition is adjusted in the extraction to take into account the fact that a non-zero DC component creates a singularity, and eventual very-low frequency components (possibly spurious) have a disproportionate weight.

To extract the spectrum centroid:

a) Calculate the power spectrum coefficients, as described in AudioSpectrumEnvelope D extraction parts a-d.

$$P_x(k), k = 0,..,\frac{NFFT}{2}$$

b) Power spectrum coefficients below 62.5 Hz are replaced by a single coefficient, with power equal to their sum and a nominal frequency of 31.25 Hz.

$$bound = floor\left(\frac{62.5 \times NFFT}{sr}\right)$$

$$P'_x(0) = \sum_{k=0}^{bound} P_x(k), f(0) = 31.25$$

$$P'_x(n) = P_x(n + bound), f(n) = (n + bound)\frac{sr}{NFFT} \text{ where } n = 1,..,\frac{NFFT}{2} - bound$$

c) Frequencies of all coefficients are scaled to an octave scale anchored at 1 kHz. The spectrum centroid is calculated as: $C = \sum_n \log_2(f(n)/1000)P'_x(n) \Big/ \sum_n P'_x(n)$ where is the $P'_x(n)$ power associated with frequency $f(n)$.

5.3.8.3.2 Purpose

Spectrum centroid is an economical description of the shape of the power spectrum. It indicates whether the power spectrum is dominated by low or high frequencies and, additionally, it is correlated with a major perceptual dimension of timbre; i.e.sharpness.

5.3.8.3.3 Motivation for the design

There are many different ways to design a spectrum centroid, according to the scale used for the values (amplitude, power, log power, cubic root power, etc.) and frequencies (linear or logarithmic scale) of spectrum coefficients. Perceptual weighting and masking can also be taken into account in more sophisticated measures. This particular design of AudioSpectrumCentroid D was chosen to be coherent with other descriptors, in

particular AudioSpectrumEnvelope D, so that a signal labeled with the former can reasonably be compared to a signal labeled with the latter.

5.3.9 AudioSpectrumSpreadType

AudioSpectrumSpread D describes the second moment of the log-frequency power spectrum.

5.3.9.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioSpectrumSpread D                              -->
<!-- ################################################################ -->
<complexType name="AudioSpectrumSpreadType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType"/>
  </complexContent>
</complexType>
```

5.3.9.2 Semantics

Name	Definition
AudioSpectrumSpreadType	Description of the spread of the log-frequency power spectrum.

5.3.9.3 Usage and extraction

5.3.9.3.1 Extraction

To be coherent with other descriptors, in particular AudioSpectrumEnvelope D, the spectrum spread is defined as the RMS deviation of the log-frequency power spectrum with respect to its center of gravity. Details are similar to AudioSpectrumCentroid D.

To extract the spectrum spread:

a) Calculate the power spectrum, $P'_x(n)$, and corresponding frequencies, $f(n)$, of the waveform as for AudioSpectrumCentroid D extraction, parts a-b.

b) Calculate the spectrum centroid, C, as described in AudioSpectrumCentroid D extraction part d.

c) Calculate the spectrum spread, S, as the RMS deviation with respect to the centroid, on an octave scale:
$$S = \sqrt{\sum_n ((\log_2(f(n)/1000) - C)^2 P'_x(n)) \Big/ \sum_n P'_x(n)}$$

5.3.9.3.2 Purpose

Spectrum spread is an economical descriptor of the shape of the power spectrum that indicates whether it is concentrated in the vicinity of its centroid, or else spread out over the spectrum. It allows differentiating between tone-like and noise-like sounds.

5.3.9.3.3 Motivation for the design

As for the spectrum centroid, there are many different ways to design a spectrum spread measure. This definition follows the same criteria as `AudioSpectrumCentroid D`, with which it is coherent.

5.3.10 AudioSpectrumFlatnessType

`AudioSpectrumFlatness D` describes the flatness properties of the spectrum of an audio signal within a given number of frequency bands.

5.3.10.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioSpectrumFlatness D                            -->
<!-- ################################################################ -->
<complexType name="AudioSpectrumFlatnessType">
  <complexContent>
    <extension base="mpeg7:AudioLLDVectorType">
      <attribute name="loEdge" type="float" default="250"/>
      <attribute name="hiEdge" type="float" default="16000"/>
    </extension>
  </complexContent>
</complexType>
```

5.3.10.2 Semantics

Name	Definition
`AudioSpectrumFlatnessType`	Description of the audio spectral flatness of the audio signal.
`loEdge`	Lower edge frequency (a default value of 250 is assumed)
`hiEdge`	Upper edge frequency (a default value of 16000 is assumed)

5.3.10.3 Usage, extraction and examples

5.3.10.3.1 **Purpose** (informative)

The `AudioSpectrumFlatnessType` describes the flatness properties of the short-term power spectrum of an audio signal. This descriptor expresses the deviation of the signal's power spectrum over frequency from a flat shape (corresponding to a noise-like or an impulse-like signal). A high deviation from a flat shape may indicate the presence of tonal components. The spectral flatness analysis is calculated for a number of frequency bands. It may be used as a feature vector for robust matching between pairs of audio signals.

5.3.10.3.2 **Extraction** (normative)

The extraction of the `AudioSpectrumFlatnessType` can be efficiently combined with the extraction of the `AudioSpectrumEnvelopeType` and is done in several steps:

a) A spectral analysis (windowing, DFT) of the input signal is performed using the same procedure and parameters specified for the extraction of the `AudioSpectrumEnvelopeType` part a-d, but with the window length, lw, corresponding to hop size (i.e. no overlap between subsequent calculations). Hence `hopSize` = 30ms is recommended for this descriptor.

b) A frequency range from `loEdge` to `hiEdge` is covered. Both limits must be chosen in quarter octave relation to 1kHz as described in the following equation, i.e.

$$edge = 2^{0.25m} \times 1KHz$$

where $m \in Z$ (i.e., m an integer).

In view of the limitations in available frequency resolution, use of `AudioSpectrumEnvelopeType` below 250 Hz is not recommended. A logarithmic frequency resolution of a 1/4 octave is used for all bands. Thus, all `AudioSpectrumFlatnessType` bands are commensurate with the frequency bands employed by `AudioSpectrumEnvelopeType`. In order to reduce the sensitivity against deviations in sampling frequency, the bands are defined in an overlapping fashion: For the calculation of the actual edge frequencies, the nominal lower edge and higher edge frequencies of each band are multiplied by the factors 0.95 and 1.05, respectively. Consequently, each band overlaps with its neighbor band by 10%. This results in band edges f_b as described in Table 2 (assuming the default `loEdge` value of 250 Hz).

The band edge frequencies are transformed to indices of power spectrum coefficients as follows: If DF is the frequency spacing of the DFT (DF = sampling rate / DFT size), the lower and higher edge of band b are defined by their power spectrum coefficient indices, il(b) and ih(b), respectively, which are derived from the edge frequencies by *nint*(f_b / DF), where *nint*() denotes rounding to the nearest integer.

For each frequency band, the flatness measure is defined as the ratio of the geometric and the arithmetic mean of the power spectrum coefficients (i.e. squared absolute DFT value, incl. grouping if required) *c(i)* within the band b (i.e. from coefficient index il to coefficient index ih, inclusive).

$$SFM_b = \frac{\sqrt[ih(b)-il(b)+1]{\prod_{i=il(b)}^{ih(b)} c(i)}}{\frac{1}{ih(b)-il(b)+1}\sum_{i=il(b)}^{ih(b)} c(i)}$$

If no audio signal is present (i.e. the mean power is zero), a flatness measure value of 1 is returned.

In order to reduce the computational effort and adapt the frequency resolution to “log” bands, all power spectrum coefficients in bands above the edge frequency of 1kHz are grouped, i.e. the above calculation is carried out using the average values over a group of power spectral coefficients rather than the single coefficients themselves. The grouping is defined in the following way:

— For all bands between nominal 1kHz and 2kHz, a grouping of 2 consecutive power spectrum coefficients is used. For all bands between nominal 2kHz and 4kHz, a grouping of 4 consecutive power spectrum coefficients is used. For all bands between nominal 4kHz and 8kHz, a grouping of 8 consecutive power spectrum coefficients is used and so on.

— For the last group of coefficients in each band, the following rule is applied: If at least 50% of the required coefficients for the group are available in that band, this last group is included using the necessary amount of additional coefficients from the successive band. Otherwise this group is not included, and the number of coefficients used from the particular band is reduced accordingly.

If the signal available to the extraction process does not supply proper signal content beyond a certain frequency limit (e.g. due to the signal sampling rate or other bandwidth limitations), no flatness values should be extracted for bands extending beyond this frequency limit. Instead, `hiEdge` should be reduced accordingly to signal the number of bands available with proper flatness data.

Table 2 — Band overlaps

Band edges f_b:	Nominal	250.0 Hz – 297.3 Hz	297.3 Hz – 353.6 Hz	353.6 Hz – 420.4 Hz	...
	Actual (overlapped)	237.5 Hz – 312.2 Hz	282.4 Hz – 371.2 Hz	335.9 Hz – 441.5 Hz	...

5.3.10.3.3 **Example** (informative)

The following is an example instantiation of `AudioSpectrumFlatnessType`. Consider a series of calculated flatness measurements of 128 consecutive analysis frames. The following instantiation shows how to summarize these values by using the `Min`, `Max`, `Mean` and `Variance` fields of the `SeriesOfVector` datatype for a 4-band flatness descriptor:

```
<AudioDescriptor xsi:type="AudioSpectrumFlatnessType" loEdge="500"
                 hiEdge="1000">
  <SeriesOfVector vectorSize="4" totalSampleNum="128">
    <Scaling ratio="64" elementNum="2"/>
    <Min dim="2 4"> 0.3 0.1 0.4 0.3 0.2 0.3 0.3 0.2 </Min>
    <Max dim="2 4"> 0.8 0.5 0.7 0.6 0.5 0.6 0.6 0.8 </Max>
    <Mean dim="2 4"> 0.6 0.4 0.5 0.4 0.4 0.5 0.4 0.4 </Mean>
    <Variance dim="2 4"> 0.1 0.11 0.06 0.08 0.07 0.1 0.09 0.07 </Variance>
  </SeriesOfVector>
</AudioDescriptor>
```

5.3.11 AudioSpectrumBasisType

The `AudioSpectrumBasis` D contains basis functions that are used to project high-dimensional spectrum descriptions into a low-dimensional representation. Spectrum dimensionality reduction plays a substantial role in automatic classification applications by compactly representing salient statistical information about audio segments. These features have been shown to perform well for automatic classification and retrieval applications. Applications to spectrogram summarization are also discussed below.

5.3.11.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioSpectrumBasis D                               -->
<!-- ################################################################ -->
<complexType name="AudioSpectrumBasisType">
  <complexContent>
    <extension base="mpeg7:AudioLLDVectorType">
      <attributeGroup ref="mpeg7:audioSpectrumAttributeGrp"/>
    </extension>
  </complexContent>
</complexType>
```

5.3.11.2 Semantics

Name	Definition
AudioSpectrumBasisType	Statistical basis functions of a spectrogram used for dimension reduction and summarization. Basis functions are stored in the Raw field of a SeriesOfVector, the dimensions of the series depend upon the usage model: For stationary basis components the dimension attribute is set to dim="N K" where N is the spectrum length and K is the number of basis functions. For time-varying basis components dim="M N K" where M is the number of blocks within the segment, N is the spectrum length and K is the number of basis functions per block. Block lengths must be at least K frames for K basis functions; default hopSize is PT500N1000F.
AudioSpectrumAttrGroup	Spectrum parameters for the basis vectors; see AudioSpectrumEnvelope.

5.3.11.3 Example (informative)

The following example shows an instance of five stationary basis functions spanning an audio segment derived from an AudioSpectrumEnvelope D with ¼-octave resolution.

```
<AudioDescriptor xsi:type="AudioSpectrumBasisType" loEdge="62.5" hiEdge="8000"
                 octaveResolution="1/4">
  <SeriesOfVector totalNumOfSamples="1" vectorSize="31 9">
    <Raw dim="31 9">
      0.082 -0.026 0.024 -0.093 0.010 -0.021 0.063 -0.103 0.057
      0.291 0.073 0.025 -0.039 0.026 -0.086 0.185 0.241 0.107
      0.267 0.062 0.030 -0.026 0.054 -0.115 0.171 0.266 0.240
      0.267 0.062 0.030 -0.026 0.054 -0.115 0.171 0.266 0.240
      0.271 -0.008 0.039 0.007 0.119 -0.067 0.033 0.165 0.175
      0.271 -0.008 0.039 0.007 0.119 -0.067 0.033 0.165 0.175
      0.269 -0.159 0.062 0.074 0.182 0.071 -0.194 0.054 -0.009
      <!-- more values here . . .                                    -->
    </Raw>
  </SeriesOfVector>
</AudioDescriptor>
```

5.3.11.4 Extraction (normative)

The following section defines the method for extracting basis functions from a spectrum. Some details of extraction are indicated to be informative with no loss of compatibility between implementations. To extract a reduced-dimension basis from an AudioSpectrumEnvelope spectrum the following steps shall be executed:

a) *Power spectrum*: instantiate an AudioSpectrumEnvelope descriptor using the extraction method defined in the AudioSpectrumEnvelope D. The resulting data will be a SeriesOfVectors with *M* frames and *N* frequency bins.

b) *Log-scale norming*: for each spectral vector, **x**, in AudioSpectrumEnvelope, convert the power spectrum to a decibel scale:

$$\chi = 10\log_{10}(\mathbf{x}_t)$$

and compute the *L2-norm* of the resulting vector:

$$r = \sqrt{\sum_{k=1}^{N} \chi_k^2}$$

the new unit-norm spectral vector is given by:

$$\tilde{\mathbf{x}} = \frac{\boldsymbol{\chi}}{r}$$

c) *Observation matrix*: place each normalized spectral frame *row-wise* into a matrix. The size of the resulting matrix is *M x N* where *M* is the number of time frames and *N* is the number of frequency bins. The matrix will have the following structure:

$$\tilde{\mathbf{X}} = \begin{bmatrix} \tilde{\mathbf{x}}_1^T \\ \tilde{\mathbf{x}}_2^T \\ \vdots \\ \vdots \\ \tilde{\mathbf{x}}_M^T \end{bmatrix}$$

d) *Basis extraction*: Extract a basis using a singular value decomposition (SVD), commonly implemented as a built-in function in many mathematical software packages using the command `[U,S,V] = SVD(X,0)`. Use the *economy* SVD when available since the row-basis functions are not required and this will increase extraction efficiency. The SVD factors the matrix from step (c) in the following way:

$$\tilde{\mathbf{X}} = \mathbf{USV}^T$$

where **X** is factored into the matrix product of three matrices; the row basis **U,** the diagonal singular value matrix **S** and the transposed column basis functions **V.** Reduce the spectral (column) basis by retaining only the first *k* basis functions, i.e. the first *k* columns of **V**:

$$\mathbf{V}_K = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_k \end{bmatrix}$$

k is typically in the range of 3-10 basis functions for sound classification and spectrum summarization applications.

To calculate the proportion of information retained for *k* basis functions use the singular values contained in matrix **S**:

$$I(k) = \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{j=1}^{N} S_{jj}}$$

where *I(k)* is the proportion of information retained for *k* basis functions and *N* is the total number of basis functions which is also equal to the number of spectral bins. The SVD basis functions are stored using a `SeriesOfVector` D in the `AudioSpectrumBasis` D.

e) *Statistically independent basis (Optional)*: after extracting the reduced SVD basis, **V,** a further step consisting of basis rotation to directions of maximal statistical independence is required for some applications. This is necessary for any application requiring maximum separation of features; for example, separation of source components of a spectrogram. A statistically independent basis is derived using an additional step of independent component analysis (ICA) after SVD extraction. The ICA basis is the same size as the SVD basis and is placed in the same `SeriesOfVector` field as the SVD basis.

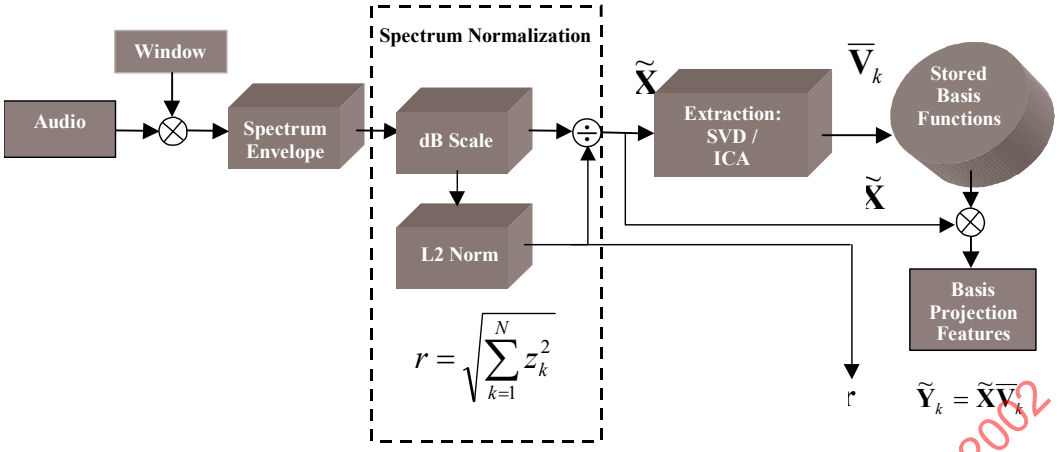

Figure 7 — Extraction method for `AudioSpectrumBasisType` and `AudioSpectrumProjectionType`

f) *Time varying components (Optional):* the extraction process (a)-(e) outlined above can be segmented into blocks over an `AudioSegment` thus providing a time-varying basis. To do this, the basis is sampled at regular intervals, default 500ms (`hopSize = PT500N1000F`), and a three-dimensional `SeriesOfVector` matrix results. The first dimension is the block index, the second is the spectral dimension and the third gives the number of basis vectors. This representation can track basis functions belonging to sources in an auditory scene.

5.3.12 AudioSpectrumProjectionType

The `AudioSpectrumProjection` D is the compliment to the `AudioSpectrumBasis D` and is used to represent low-dimensional features of a spectrum after projection against a reduced rank basis. These two types are always used together. The low-dimensional features of the `AudioSpectrumProjection D` consist of a `SeriesOfVectors`, one vector for each frame, *t,* of the normalized input spectrogram, $\tilde{\mathbf{x}}_t$. Each spectral frame from steps (a)-(c) above yields a corresponding projected vector, $\mathbf{y}_t$, that is stored in the `SeriesOfVector D`.

5.3.12.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioSpectrumProjection D                        -->
<!-- ################################################################ -->
<complexType name="AudioSpectrumProjectionType">
  <complexContent>
    <extension base="mpeg7:AudioLLDVectorType"/>
  </complexContent>
</complexType>
```

5.3.12.2 Semantics

Name	Definition
AudioSpectrumProjectionType	Low-dimensional representation of a spectrum using projection against spectral basis functions. The projected data is stored in a SeriesOfVector D. The dimensions of the SeriesOfVector D depend upon the usage model: For stationary basis components the dimension attribute is set to dim="N K+1" where N is the spectrum length and K is the number of basis functions. For time-varying basis components dim="M N K+1" where M is the number of blocks, N is the spectrum length and K is the number of basis functions per block.

5.3.12.3 Example (informative)

The following example shows projection coefficients corresponding to the AudioSpectrumBasis example given above. Note that the vectorSize attribute is set to the number of basis functions plus one.

```
<AudioDescriptor xsi:type="AudioSpectrumProjectionType">
  <SeriesOfVector hopSize="PT10N1000F" totalNumOfSamples="263"
                  vectorSize="10">
    <Raw dim="263 10">
      0.359 -0.693 0.345 -0.145 -0.129 -0.170 -0.117 -0.448 0.092 0.045
      0.364 -0.690 0.308 -0.147 -0.127 -0.184 -0.122 -0.476 0.130 0.206
      0.353 -0.656 0.382 -0.175 -0.137 -0.143 -0.167 -0.478 0.207 0.186
      <!-- more values here ...                                        -->
      0.998 -0.342 0.569 0.592 0.103 -0.280 0.159 -0.070 -0.293 -0.006
      1.000 -0.324 0.562 0.601 0.119 -0.273 0.165 -0.058 -0.305 -0.025
    </Raw>
  </SeriesOfVector>
</AudioDescriptor>
```

5.3.12.4 Extraction (normative)

The elements of each AudioSpectrumProjection vector shall represent, in order, the L2-norm value, r_t, obtained in step (b) of AudioSpectrumBasis extraction. This shall be followed by the inner product of the normalized spectral frame, $\tilde{\mathbf{x}}_t$, from step (b) above and each of the basis vectors, $\mathbf{v}_k$, from step (d) or (e) above. The resulting vector has *k+1* elements, where *k* is the number of basis components, and it is defined by:

$$\mathbf{y}_t = \begin{bmatrix} r_t & \tilde{\mathbf{x}}_t^T \mathbf{v}_1 & \tilde{\mathbf{x}}_t^T \mathbf{v}_2 & \cdots & \tilde{\mathbf{x}}_t^T \mathbf{v}_k \end{bmatrix}.$$

5.3.12.5 Usage (informative)

5.3.12.5.1 Automatic Sound Classification and Retrieval

The AudioSpectrumBasis D and AudioSpectrumProjection D are used in the Sound Classification and Indexing Tools for automatic classification of audio segments using probabilistic models. In this application, basis functions are computed for the set of training examples and are stored along with a probabilistic model of the training sounds. Using these methods, audio segments can be automatically classified into categories such as *speech*, *music* and *sound effects*. Another example is automatic classification of music genres such as *Salsa,*

HipHop, Reggae or *Classical.* For more information on automatic classification and retrieval of audio see the `SoundClassificationModel` DS below.

5.3.12.5.2 Spectrogram Summarization

The spectrum basis descriptors can be used to view independent subspaces of a spectrogram; for example, we may wish to view subspaces that contain independent source sounds in a mixture. To extract independent spectrogram subspaces for an audio segment, first perform extraction for `AudioSpectrumBasis`. Then the `AudioSpectrumProjection` is extracted as defined above.

Reconstruction of an independent spectrogram frame, $\mathbf{x}_t^T$, is calculated by taking the outer product of the *j*th vector in `AudioSpectrumBasis` and the *j*+1th vector in `AudioSpectrumProjection` and multiplying by the normalization coefficient *r*:

$$\mathbf{x}_t^T = r_t \, \mathbf{y}_t[j+1]\overline{\mathbf{v}}[j]^+$$

where the + operator indicates the pseudo-inverse. These frames are concatenated to form a new spectrogram. Any combination of spectrogram subspaces can be summed to obtain either individual source spectrograms or an approximation of the original spectrogram.

The salient features of a spectrogram may be efficiently represented with much less data than a full spectrogram using independent component basis functions. The following example is taken from a recording of a song featuring guitar, drums, hi-hat, bass guitar, and organ. Figure 8 shows the original full-bandwidth spectrogram and Figure 9 shows a 10-component reconstruction of the same spectrogram. The data ratio, *R*, between the reduced-dimension spectrum and the full-bandwidth spectrum is:

$$R = K\left(\frac{1}{M} + \frac{1}{N}\right)$$

where *K* is the number of basis components, *M* is the number of frames in the spectrogram and *N* is the number of frequency bins. For example, a 5-component summary of 500 frames of a 64-bin spectrogram leads to a data reduction of ~11:1.

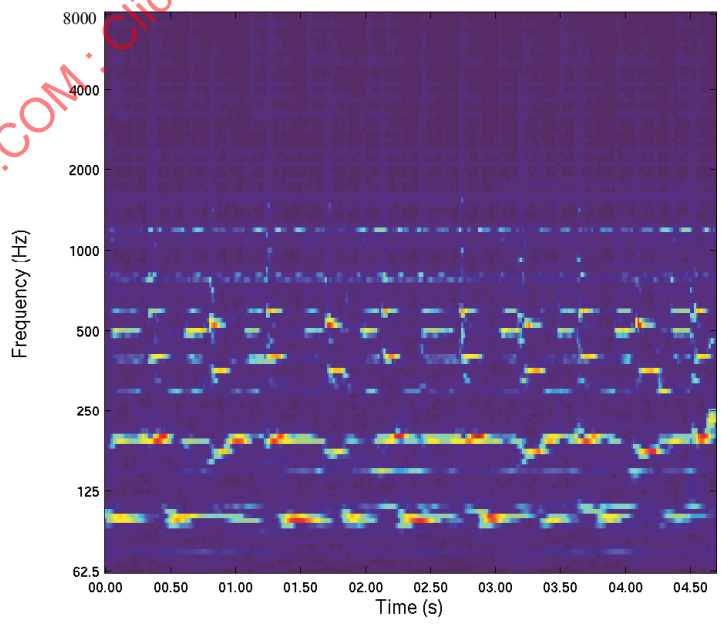

Figure 8 — `AudioSpectrumEnvelope` description of a pop song. The required data storage is *NM* values where *N* is the number of spectrum bins and *M* is the number of time points

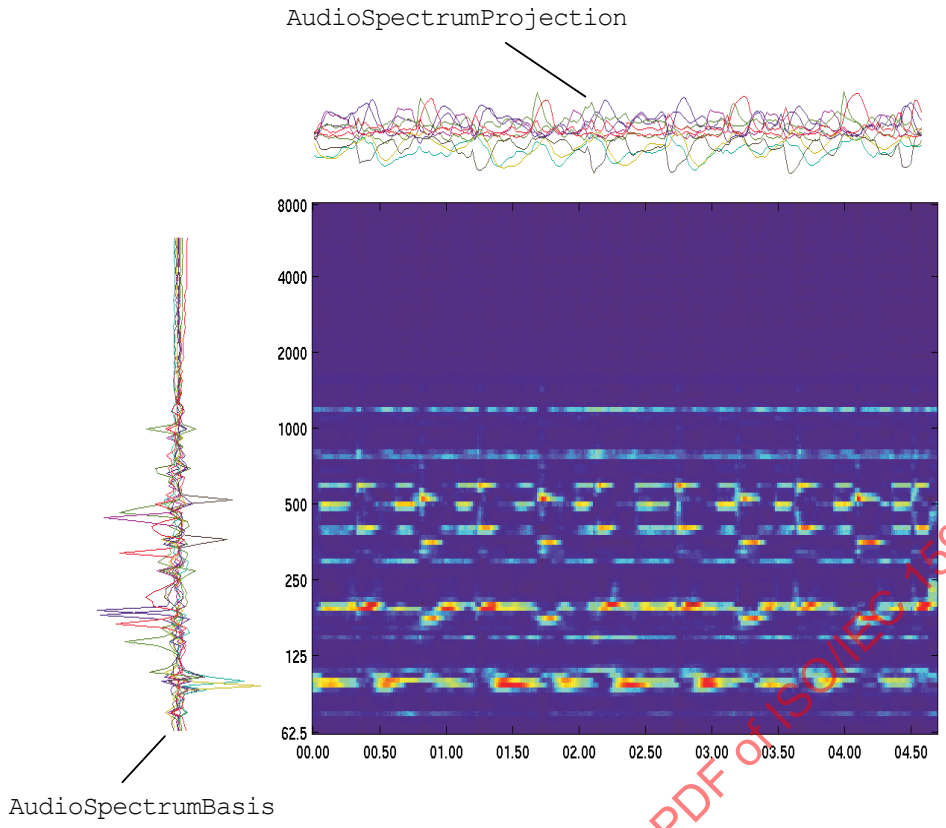

Figure 9 — 10-basis component reconstruction showing most of the detail of the original spectrogram including guitar, bass guitar, hi-hat and organ notes. The left vectors are an `AudioSpectrumBasis` D and the top vectors are the corresponding `AudioSpectrumProjection` D. The two vectors are combined using the reconstruction equation given above. The required data storage is 10*(M+N)* values

5.3.12.6 AudioFundamentalFrequencyType

`AudioFundamentalFrequency` D describes the fundamental frequency of the audio signal.

5.3.12.7 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioFundamentalFrequency D                        -->
<!-- ################################################################ -->
<complexType name="AudioFundamentalFrequencyType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType">
      <attribute name="loLimit" type="float" default="25"/>
      <attribute name="hiLimit" type="float" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

5.3.12.8 Semantics

Name	Definition
AudioFundamentalFrequencyType	Description of the fundamental frequency of the audio signal.
loLimit	Lower limit of search space, in Hz.
hiLimit	Upper limit of search space, in Hz.

5.3.12.9 Usage and extraction

5.3.12.9.1 Extraction

The extraction method is not specified in complete detail in order to promote choice of strategy. However the following shall be present in all cases.

The limits of the search range shall be specified using loLimit and hiLimit. The extraction method shall report a fundamental frequency for any signal that is periodic over the analysis interval with a fundamental within the search range.

The extraction method shall provide a confidence measure, between 0 and 1, to be used as a weight in scaling operations. Values of the estimate for which the weight is zero shall be considered non-periodic and ignored in similarity and scaling operations. The handling of non-zero values, that allow periodic values to be differentially weighted, is left up to the specific application.

One extraction method is detailed in the extraction of the AudioHarmonicity D. This is not the best method available but it gives reasonable estimates of the fundamental frequency in stationary signals.

5.3.12.9.2 Purpose

Fundamental frequency is a good predictor of musical pitch and speech intonation. As such it is an important descriptor of an audio signal. This descriptor is not designed to be a descriptor of melody, but it may nevertheless be possible to make meaningful comparisons between data labeled with a melody descriptor, and data labeled with fundamental frequency.

5.3.12.9.3 Motivation for the design

Fundamental frequency is complementary to the log-frequency logarithmic spectrum, in that, together with the AudioHarmonicity D, it specifies aspects of the detailed harmonic structure of periodic sounds that the logarithmic spectrum cannot represent for lack of resolution. The inclusion of a confidence measure, using the Weight field of the SeriesOfScalarType is an important part of the design, that allows proper handling and scaling of portions of signal that lack clear periodicity.

5.3.13 AudioHarmonicityType

AudioHarmonicity D describes the degree of harmonicity of an audio signal.

5.3.13.1 Syntax

```
<!-- ################################################################## -->
<!-- Definition of AudioHarmonicity D                                   -->
<!-- ################################################################## -->
<complexType name="AudioHarmonicityType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="HarmonicRatio" type="mpeg7:AudioLLDScalarType"/>
        <element name="UpperLimitOfHarmonicity"
                 type="mpeg7:AudioLLDScalarType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.3.13.2 Semantics

Name	Definition
AudioHarmonicityType	Combined measures of Harmonic Ratio and Upper limit of harmonicity.
HarmonicRatio	Description of the ratio of harmonic power to total power.
UpperLimitOfHarmonicity	The frequency beyond which the spectrum cannot be considered harmonic.

5.3.13.3 Usage and extraction

5.3.13.3.1 Extraction

AudioHarmonicity D contains two measures: HarmonicRatio, and UpperLimitOfHarmonicity.

HarmonicRatio is loosely defined as the proportion of harmonic components within the power spectrum. It is derived from the correlation between the signal and a lagged representation of the signal, lagged by the fundamental period of the signal. In order to avoid dependency on the actual fundamental frequency estimate, the algorithm produces its own estimate by searching for the maximum value in the normalized cross-correlation of the signal. The algorithm is:

a) Calculate $r(i,k)$, the normalised cross correlation of frame i with lag k:

$$r(i,k) = \sum_{j=m}^{m+n-1} s(j)s(j-k) \Big/ \left(\sum_{j=m}^{m+n-1} s(j)^2 * \sum_{j=m}^{m+n-1} s(j-k)^2 \right)^{0.5}$$

Where

s is the audio signal

$m = i * n$, where $i = 0, M-1 =$ frame index, $M =$ number of frames

$n = t * sr$, where $t =$ analysis window size (default 10ms) and $sr =$ sampling rate

$k = 1, K =$ lag, where $K = \omega * sr$, $\omega =$ maximum fundamental period expected (default 40ms)

b) The Harmonic Ratio $H(i)$ is chosen as the maximum $r(i,k)$ in each frame, i:

$$H(i) = \max_{k=1,n-1} r(i,k)$$

This value is 1 for a purely periodic signal, and it will be close to 0 for white noise. The estimate can be refined by replacing each local maximum of $r(i,k)$ by the maximum of a 3-point parabolic fit centered upon it.

`UpperLimitOfHarmonicity` is loosely defined as the frequency beyond which the spectrum cannot be considered harmonic. It is calculated based on the power spectra of the original and a comb-filtered signal. The algorithm is:

a) Determine the combed signal

$$c(j) = s(j) - \lambda s(j-K), j = m,(m+n-1)$$

where

$\lambda = \sum_{j=m}^{m+n-1} s(j)s(j-K) \Big/ \sum_{j=m}^{m+n-1} s^2(j-K)$ … is the optimal gain

K is the lag corresponding to the maximum cross correlation ($H(i) = r(i,K)$), and the fundamental period estimate. If K is fractional, s(j-K) is calculated by linear interpolation.

b) Calculate the DFTs of the signal, s(j), and the comb-filtered signal, c(j), using the technique described in `AudioSpectrumEnvelope D`. Calculate power spectra, and group the components below 62.5 Hz as explained for `AudioSpectrumCentroid D`.

c) For each frequency, $f_{\lim}$, calculate the sum of power beyond that frequency, for both the original and comb-filtered signal, and take their ratio.

$$a(f_{\lim}) = \sum_{f=f_{\lim}}^{f_{\max}} p'(f) \Big/ \sum_{f=f_{\lim}}^{f_{\max}} p(f)$$

where $p(f)$ and $p'(f)$ are the power spectra of the unfiltered and filtered signal respectively, and $f_{\max}$ is the maximum frequency of the DFT.

d) Starting from $f_{\lim} = f_{\max}$ and moving down in frequency, find the greatest frequency, $f_{u\lim}$, for which this ratio is smaller than a threshold (Threshold = 0.5).

e) Convert this value to an octave scale based on 1 kHz

`UpperLimitOfHarmonicity` = $\log_2(f_{u\lim}/1000)$

5.3.13.3.2 Purpose

A harmonicity measure allows distinguishing between sounds that have a harmonic spectrum (musical sounds, voiced speech, etc.) and those that have a non-harmonic spectrum (noise, unvoiced speech, dense mixtures of instruments, etc.).

5.3.13.3.3 Motivation for the design

Together with the `AudioFundamentalFrequency D`, `AudioHarmonicity D` describes the harmonic structure of sound. These features are orthogonal and complementary to a descriptor such as `AudioSpectrumEnvelope D`. The exact definitions of the measures (`HarmonicRatio` and `UpperLimitOfHarmonicity`) are designed to be easy to extract, and coherent with the definitions of other descriptors (most of which are based on power).

5.3.14 Timbre Descriptors

5.3.14.1 Introduction

The following descriptors (5.3.15 to 5.3.21) are distinct from the preceding low-level descriptors. In the context of the high-level Timbre description tools (section 5.2), they are intended to be descriptors that apply to an entire audio segment, rather than being primarily sampled types. However it is possible to retain the instantaneous sampled series for a number of the descriptors, using the `SeriesOfScalar` of the `AudioLLDScalarType`. `SeriesOfScalar` may not be chosen for the `LogAttackTime D`, the `SpectralCentroid D` and the `TemporalCentroid D`, as these descriptors are not defined as an instantaneous series.

5.3.14.2 Usage and Extraction

As many of the timbre descriptors rely on a previous estimation of the fundamental frequency and the harmonic peaks of the spectrum or on the temporal signal envelope (see Figure 10), the extraction of these is explained first rather than repeating it for each timbre descriptor.

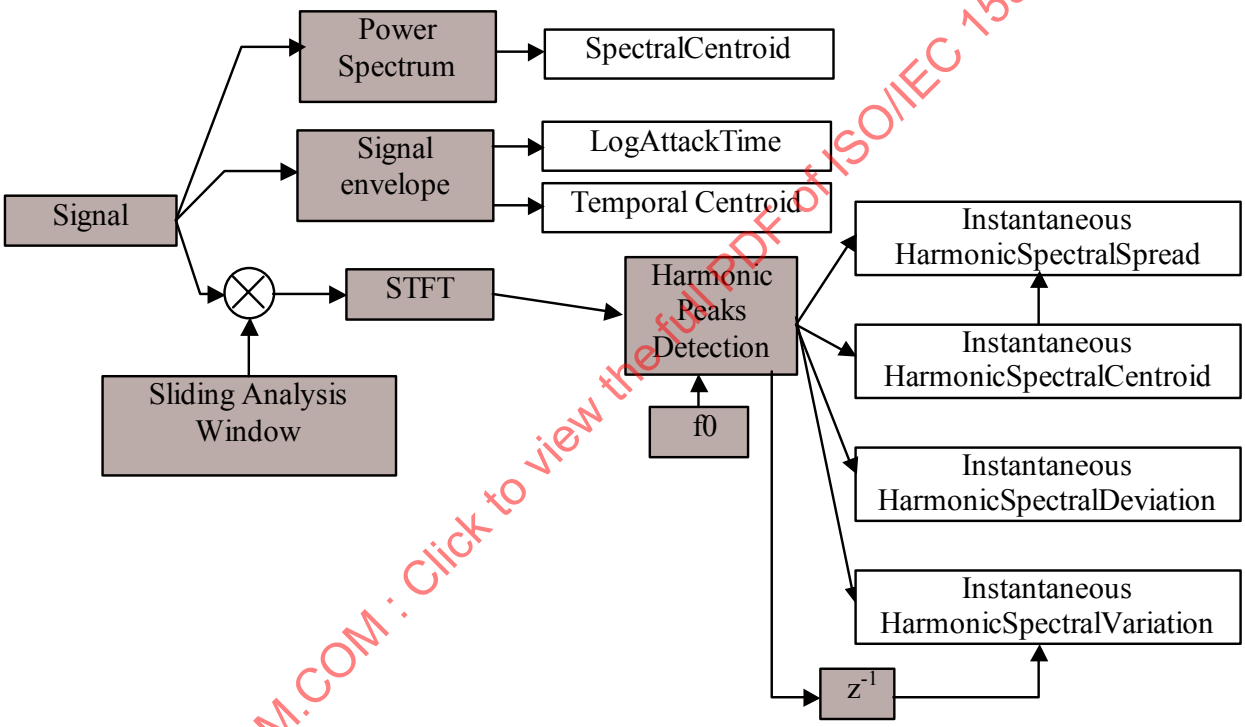

Figure 10 — Timbre Descriptor Estimation

5.3.14.3 Estimation of spectral parameters

The calculation of the fundamental frequency and the harmonic peaks is required before the calculation of each of the instantaneous harmonic spectral features, including centroid, deviation, spread and variation. Many of the timbre descriptors have been designed for specific use upon harmonic signals, such as a monophonic musical signal. Each descriptor describes a sound segment. An example of a sound segment would be a single note played on a clarinet.

5.3.14.3.1 **Recommended Analysis Parameters** (informative)

The recommended parameters for extraction of the timbre descriptors depend upon whether the global values alone are required or whether the instantaneous values are also required.

If only the global values of the Timbre descriptors are required then the recommended extraction parameters are:

— Analysis window type: hamming

— Analysis window size: 8 fundamental periods

— Hop size: 4 fundamental periods

If the instantaneous series of values are required then the requirement of the AudioLLDScalar D to restrict the `hopSize` attribute to integer multiples/divisors of 10ms applies. For these instances the recommended extraction parameters are:

— Analysis window type: hamming

— Analysis window size: 30ms

— Hop size: 10ms

5.3.14.3.2 Estimation of the fundamental frequency f0 (informative)

The fundamental frequency is the frequency that best explains the periodicity of a signal. While numerous methods have been proposed in order to estimate it, one can simply compute the local normalized auto-correlation function of the signal and take its first maximum in order to estimate the local fundamental period. The local fundamental frequency is then estimated by the inverse of the time corresponding to the position of this maximum.

5.3.14.3.3 Harmonic Peaks Detection (informative)

The harmonic peaks are the peaks of the spectrum located "around" the multiple of the fundamental frequency of the signal. The term "around" is used in order to take into account the slight variations of harmonicity of some sounds (piano for example). While numerous methods have been proposed in order to estimate the harmonic peaks, one can simply look for the maxima of the amplitude of the Short Time Fourier Transform (STFT) close to the multiples of the fundamental frequency. The frequencies of the harmonic peaks are then estimated by the positions of these maxima while the amplitudes of these maxima determine their amplitudes.

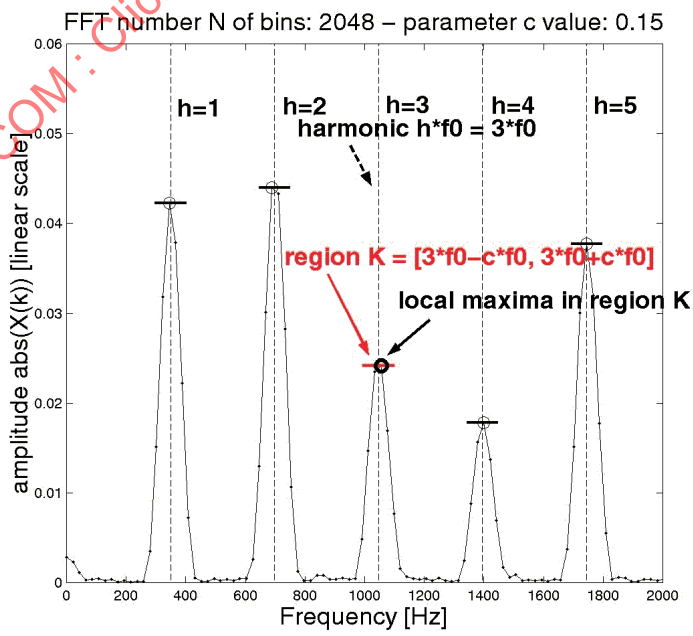

Figure 11 — Harmonic Peaks Detection (informative)

5.3.14.3.4 Suggested algorithm for harmonic peaks detection (informative)

To determine the amplitude, A, and frequency, f, of harmonic '*harmo*' in the frame '*frame*' do the following:

Let X(*k,frame*), *k* = 1,N be the STFT (of size N) of the frame, 'frame', of data (see Section 5.3.14.3.1 for extraction parameters) (See Figure 11)

$$A(frame, harmo) = \max_{m \in [a,b]} (|X(m, frame)|) = |X(M, frame)|$$

$$f(frame, harmo) = M \times DF$$

where

DF=sr/N is the frequency separation of coefficients

sr is the sampling rate

f0 is the estimated fundamental frequency

$$a = floor((harmo - c)\frac{f0}{DF}) \text{ and } b = ceil(harmo + c)\frac{f0}{DF})$$

where $c \in [0,0.5]$, determines the tolerated non-harmonicity. A value of c=0.15 is recommended.

5.3.14.4 Estimation of temporal parameters

5.3.14.4.1 Log-Attack-Time and Temporal Centroid: Signal envelope

While numerous methods have been proposed in order to compute the signal envelope, one can simply use a signal's power function over time. This function can be estimated by computing the local mean square value of the signal amplitude within a running window.

5.3.15 LogAttackTimeType

5.3.15.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of LogAttackTime D                                    -->
<!-- ################################################################ -->
<complexType name="LogAttackTimeType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType"/>
  </complexContent>
</complexType>
```

5.3.15.2 Semantics

Name	Definition
LogAttackTime	The LogAttackTime is defined as the logarithm (decimal base) of the time duration between the time the signal starts to the time it reaches its stable part. Unit: [$\log_{10}$ sec] Range: [$\log_{10}$(1/sr), determined by the length of the signal] Where sr stands for sampling rate

5.3.15.3 Usage and Extraction

5.3.15.3.1 Extraction

a) Estimate the temporal signal envelope over the time of the segment

b) Compute the `LogAttackTime`, LAT, as follows

$$LAT = \log_{10}(T1 - T0)$$

where

— *T0* is the time the signal starts;

— *T1* is the time the signal reaches its sustained part (harmonic space) or maximum part (percussive space).

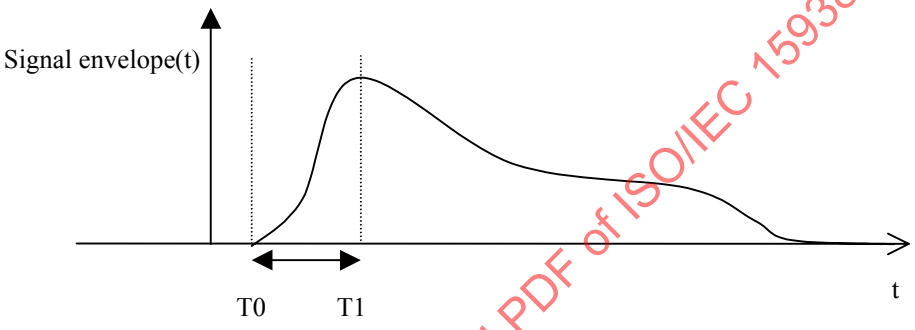

Figure 12 — Illustration of log-attack time

— (Informative) T0 can be estimated as the time the signal envelope exceeds 2% of its maximum value. T1 can be estimated, simply, as the time the signal envelope reaches its maximum value (as shown in Figure 12).

5.3.15.3.2 Motivation for the design

The 'attack' of a sound is the first part of a sound, before a real note develops.

5.3.16 HarmonicSpectralCentroidType

5.3.16.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of HarmonicSpectralCentroid D                         -->
<!-- ################################################################ -->
<complexType name="HarmonicSpectralCentroidType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType"/>
  </complexContent>
</complexType>
```

5.3.16.2 Semantics

Name	Definition
`HarmonicSpectralCentroid`	The `HarmonicSpectralCentroid` is computed as the average over the sound segment duration of the instantaneous `HarmonicSpectralCentroid` within a running window. The instantatneous `HarmonicSpectralCentroid` is computed as the amplitude (linear scale) weighted mean of the harmonic peaks of the spectrum. Unit: [Hz] Range: [0,sr/2]

5.3.16.3 Usage and Extraction

5.3.16.3.1 Extraction

The `HarmonicSpectralCentroid` may be extracted using the following algorithm

a) Estimate the harmonic peaks over the sound segment

b) Calculate the instantaneous `HarmonicSpectralCentroid`, IHSC, for each frame as follows:

$$IHSC(frame) = \frac{\sum_{harmo=1}^{nb\_harmo} f(frame, harmo) \cdot A(frame, harmo)}{\sum_{harmo=1}^{nb\_harmo} A(frame, harmo)}$$

where

— *A(frame,harmo)* is the amplitude of the harmonic peak number “harmo” at the frame number “frame”

— *f(frame,harmo)* is the frequency of the harmonic peak number “harmo” at the frame number “frame”

— *nb_harmo* is the number of harmonics taken into account

c) Calculate the `HarmonicSpectralCentroid`, HSC, for the sound segment as follows:

$$HSC = \frac{\sum_{frame=1}^{nb\_frames} IHSC(frame)}{nb\_frames}$$

where

— nb_frames is the number of frames in the sound segment

5.3.16.3.2 Motivation for the design

The use of a linear frequency scale instead of a logarithmic one is derived from experimental results on human perception of timbre similarity. The use of a linear scale instead of a logarithmic one significantly increases the explanation of the experimental results.

5.3.17 HarmonicSpectralDeviationType

5.3.17.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of HarmonicSpectralDeviation D                        -->
<!-- ################################################################ -->
<complexType name="HarmonicSpectralDeviationType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType"/>
  </complexContent>
</complexType>
```

5.3.17.2 Semantics

Name	Definition
HarmonicSpectralDeviation	The HarmonicSpectralDeviation is computed as the average over the sound segment duration of the instantaneous HarmonicSpectralDeviation within a running window. The instantaneous HarmonicSpectralDeviation is computed as the spectral deviation of log-amplitude components from a global spectral envelope. Unit: [-] Range: [0,1]

5.3.17.3 Usage and Extraction

5.3.17.3.1 Extraction

The HarmonicSpectralDeviation may be extracted using the following algorithm

a) Estimate the harmonic peaks over the sound segment

b) Estimate the spectral envelope (SE)

(Informative) To approximate the local Spectral Envelope take the mean amplitude of three adjacent harmonic peaks. To evaluate the ends of the envelope simply use the mean amplitude of two adjacent harmonic peaks.

For *harmo* = 1

$$SE(frame, harmo) = \frac{A(frame, harmo) + A(frame, harmo+1)}{2}$$

For *harmo* = 2 to *nb_harmo*-1

$$SE(frame, harmo) = \frac{\sum_{i=-1}^{1} A(frame, harmo+i)}{3}, harmo = 2, nb\_harmo - 1$$

For *harmo* = *nb_harmo*

$$SE(frame, harmo) = \frac{A(frame, harmo-1) + A(frame, harmo)}{2}$$

where

— *nb_harmo* is the number of harmonics taken into account

c) Calculate the instantaneous `HarmonicSpectralDeviation`, IHSD, for each frame as follows:

$$IHSD(frame) = \frac{\sum_{harmo=1}^{nb\_harmo} |\log_{10}(A(frame,harmo)) - \log_{10}(SE(frame,harmo))|}{\sum_{harmo=1}^{nb\_harmo} \log_{10}(A(frame,harmo))}$$

where

— *A(frame,harmo)* is the amplitude of the harmonic peak number “harmo” at the frame number “frame”

— *SE(frame,harmo)* is the local Spectral Envelope around the harmonic peak number harmo

— *nb_harmo* is the number of harmonics taken into account

d) Calculate the `HarmonicSpectralDeviation` , HSD, for the sound segment as follows:

$$HSD = \frac{\sum_{frame=1}^{nb\_frames} IHSD(frame)}{nb\_frames}$$

where

— *nb_frames* is the number of frames in the sound segment

5.3.17.3.2 Motivation for the design

The use of a logarithmic amplitude scale instead of a linear one is derived from experimental results on human perception of timbre similarity. The use of a logarithmic scale instead of a linear one significantly increases the explanation of these experimental results.

5.3.18 HarmonicSpectralSpreadType

5.3.18.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of HarmonicSpectralSpread D                           -->
<!-- ################################################################ -->
<complexType name="HarmonicSpectralSpreadType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType"/>
  </complexContent>
</complexType>
```

5.3.18.2 Semantics

Name	Definition
`HarmonicSpectralSpread`	The `HarmonicSpectralSpread` is computed as the average over the sound segment duration of the instantaneous `HarmonicSpectralSpread` within a running window. The instantaneous `HarmonicSpectralSpread` is computed as the amplitude weighted standard deviation of the harmonic peaks of the spectrum, normalized by the instantaneous `HarmonicSpectralCentroid`. Units: [-] Range: [0,1]

5.3.18.3 Usage and Extraction

5.3.18.3.1 Extraction

The HSS may be extracted using the following algorithm

a) Estimate the harmonic peaks over the sound segment

b) Estimate the instantaneous `HarmonicSpectralCentroid`, IHSC, of each frame

c) Calculate the instantaneous `HarmonicSpectralSpread`, IHSS, for each frame as follows:

$$IHSS(frame) = \frac{1}{IHSC(frame)} \sqrt{\frac{\sum_{harmo=1}^{nb\_harmo} A^2(frame,harmo) \cdot [f(frame,harmo) - IHSC(frame)]^2}{\sum_{harmo=1}^{nb\_harmo} A^2(frame,harmo)}}$$

where

— *A(frame,harmo)* is the amplitude of the harmonic peak number “harmo” at the frame number “frame”

— *f(frame,harmo)* is the frequency of the harmonic peak number “harmo” at the frame number “frame”

— *nb_harmo* is the number of harmonics taken into account

d) Calculate the `HarmonicSpectralSpread`, HSS, for each sound segment as follows:

$$HSS = \frac{\sum_{frame=1}^{nb\_frames} IHSS(frame)}{nb\_frames}$$

where

— *nb_frames* is the number of frames in the sound segment

5.3.18.3.2 Motivation for the design

As for the spectral centroid, there are many different ways to design a spectrum spread measure. This definition follows the same criteria as `HarmonicSpectralCentroid D`, with which it is coherent.

5.3.19 HarmonicSpectralVariationType

5.3.19.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of HarmonicSpectralVariation D                        -->
<!-- ################################################################ -->
<complexType name="HarmonicSpectralVariationType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType"/>
  </complexContent>
</complexType>
```

5.3.19.2 Semantics

Name	Definition
`HarmonicSpectralVariation`	The `HarmonicSpectralVariation` is defined as the mean over the sound segment duration of the instantaneous `HarmonicSpectralVariation`. The instantaneous `HarmonicSpectralVariation` is defined as the normalized correlation between the amplitude of the harmonic peaks of two adjacent frames. Units: [-] Range: [0,1]

5.3.19.3 Usage and Extraction

The HSV may be extracted using the following algorithm

a) Estimate the harmonic peaks over the sound segment

b) Calculate the instantaneous `HarmonicSpectralVariation`, IHSV, for each frame as follows:

$$IHSV(frame) = 1 - \frac{\sum_{harmo=1}^{nb\_harmo} A(frame-1, harmo) \cdot A(frame, harmo)}{\sqrt{\sum_{harmo=1}^{nb\_harmo} A^2(frame-1, harmo)} \cdot \sqrt{\sum_{harmo=1}^{nb\_harmo} A^2(frame, harmo)}}$$

where

— *A(frame,harmo)* is the amplitude of the harmonic peak number “harmo” at the frame number “frame”

— *nb_harmo* is the number of harmonics taken into account

c) Calculate the `HarmonicSpectralVariation`, HSV, for the sound segment as follows:

$$HSV = \frac{\sum_{frame=1}^{nb\_frames} IHSV(frame)}{nb\_frames}$$

where

— *nb_frames* is the number of frames in the sound segment

5.3.20 SpectralCentroidType

5.3.20.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SpectralCentroid D                                 -->
<!-- ################################################################ -->
<complexType name="SpectralCentroidType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType"/>
  </complexContent>
</complexType>
```

5.3.20.2 Semantics

Name	Definition
SpectralCentroid	The SpectralCentroid is computed as the power weighted average of the frequency of the bins in the power spectrum. Unit: [Hz] Range: [0,sr/2] where sr stands for sampling rate

5.3.20.3 Usage and Extraction

The SC may be extracted using the following algorithm

a) Determine the power spectrum over the sound segment. (Informative) While numerous methods have been proposed in order to compute the power spectrum, one can simply use the Welch method (averaged periodogram) both for harmonic and percussive sounds.

b) Calculate the SpectralCentroid, SC, for the segment as follows:

$$SC(frame) = \frac{\sum_{k=1}^{powerspectrum\_size} f(k) \cdot S(k)}{\sum_{k=1}^{powerspectrum\_size} S(k)}$$

where

— S(k) is the kth power spectrum coefficient

— f(k) stands for the frequency of the kth power spectrum coefficient

5.3.21 TemporalCentroidType

5.3.21.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of TemporalCentroid D                                 -->
<!-- ################################################################ -->
<complexType name="TemporalCentroidType">
  <complexContent>
    <extension base="mpeg7:AudioLLDScalarType"/>
  </complexContent>
</complexType>
```

5.3.21.2 Semantics

Name	Definition
TemporalCentroid	The TemporalCentroid is defined as the time averaged over the energy envelope. Unit: [sec] Range: [0,determined by the length of the signal]

5.3.21.3 Usage and Extraction

The TemporalCentroid may be extracted using the following algorithm

a) Calculate the Signal Envelope, SEnv, (as described in clause 5.3.14.4.1)

b) Calculate the TemporalCentroid, TC as follows:

$$TC = \frac{\sum_{n=1}^{length(SEnv)} n / sr \cdot SEnv(n)}{\sum_{n=1}^{length(SEnv)} SEnv(n)}$$

where

— SEnv is the Signal Envelope.

— sr is the Sampling Rate.

5.4 Silence

5.4.1 Introduction

The Silence D describes a perceptual feature of a sound track capturing the fact that no significant sound is occurring in this segment. It is useful for the segmentation of audio material into subparts, giving access to its physical structure.

The basic information of the description of a silent segment is the start time and the duration, which is given by times in the audio segment to which the silence descriptor is attached. The silence segments themselves are either given for the whole segment or as a time mask for the segment.

5.4.2 SilenceHeaderType

5.4.2.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SilenceHeader header                               -->
<!-- ################################################################ -->
<complexType name="SilenceHeaderType">
  <complexContent>
    <extension base="mpeg7:HeaderType">
      <attribute name="minDuration" type="mpeg7:mediaDurationType"
                 use="required"/>
    </extension>
  </complexContent>
</complexType>
```

5.4.2.2 Semantics

Name	Definition
SilenceHeaderType	Information shared by many silence descriptors
minDuration	The minDuration attribute is used to communicate a minimum temporal threshold determining whether a signal portion is identified as a silent segment (see information on extraction process). The minDuration element is usually applied uniformly to a complete segment decomposition as a parameter for the extraction algorithm.

5.4.3 SilenceType

5.4.3.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SilenceD                                           -->
<!-- ################################################################ -->
<complexType name="SilenceType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <attribute name="confidence" type="mpeg7:zeroToOneType" default="1.0"/>
      <attribute name="minDurationRef" type="anyURI" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

5.4.3.2 Semantics

Name	Definition
confidence	The confidence attribute measures how confident the detection process is that the segment is containing silence. Unit: None Range: [0,1]
minDurationRef	Reference to minDuration information in a SilenceHeader

5.4.4 Usage, examples and extraction (informative)

Silence can be regarded as a semantic concept. For example, a long silent period in a movie may signify that something important is about to happen, such as falling in love or increasing tension for an expected event. Together with speech, music and ambient sound descriptors, the silence descriptors captures the basic semantic events occurring in audio material.

Silence can be used to segment audio materials at varying levels of detail depending on the parameters used during the detection phase. For example, a high threshold may be used to detect long pauses between sentences as silence while tolerating shorter breaks between phrases. A lower threshold may be used to identify phrasal pauses or even pauses between words. Similarly, pauses in music pieces of varying length may be determined. In this way, audio material may be split up into parts that reflect its physical structure. Segmentation is an important prerequisite for further classification of the identified segments.

Target applications are:

— Direct access to semantic segmentations/events of audio material.

— Segmentation tools for annotation and retrieval, e.g. news story segmentation, captioning software.

The silence extractor can be implemented in various ways. One possible implementation of a confidence measure is to investigate the calculated (loudness) function over time with respect to an assumed (loudness) threshold, see informative description of an example silence detection algorithm. The confidence measure can e.g. be calculated from the ratio of the area under the (loudness) threshold and the area under the (loudness) curve, clipped to the range [0 ... 1].

5.4.4.1 Description of a silence detection algorithm

This section describes how a silence detector may be implemented by using an FFT-based algorithm to calculate the sliding loudness of the signal. A simple silence detection algorithm is described.

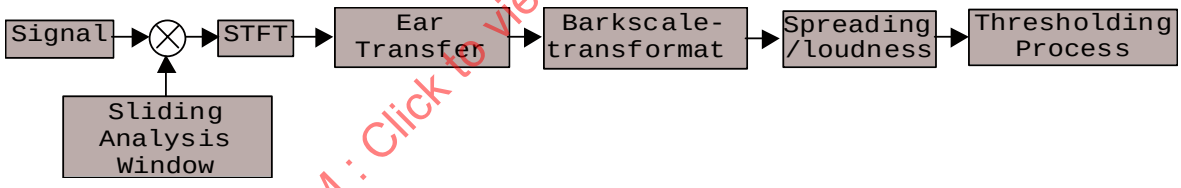

Figure 13 — Loudness based Silence Detector

The signal processing is carried out in several steps:

a) Sliding short Time Fourier Transform (STFT)

b) Filtering with outer ear transfer function

c) Summing the energy into 1 bark wide bands

d) Calculating masking threshold, considering threshold in quiet and temporal masking

e) Calculating partial loudness and overall loudness

f) Detecting silence segments by a suitable thresholding process

Descriptions for steps (a) – (d) can be found in ISO/IEC 11172-3, Annex D or ISO/IEC 13818-7, Annex B.

A thresholding process could employ several parameters, such as silence level thresholds and a minimum duration for which those thresholds are to be exceeded for a change in detector output. Such a scheme is illustrated by the following figure:

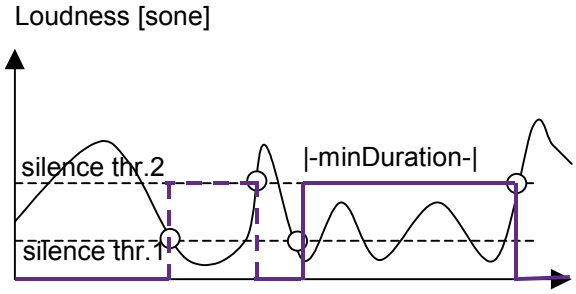

Figure 14 — Silence thresholds

Depending on the type and complexity of the audio signals, it may be necessary to automatically adapt the thresholding process to the characteristics of the background noise. An "intelligent" silence detector could additionally support perceptually-based concepts for silence, such as voice activity measures etc..

5.4.4.2 Example

```
<AudioSegment>
  <Header xsi:type="SilenceHeaderType" id="shOne">
    <minDuration>PT3N10F</minDuration>
  </Header>
  <MediaTime>
    <MediaRelTimePoint timeBase="MediaLocator[1]">P0S</MediaRelTimePoint>
    <MediaDuration>PT28S3N10F</MediaDuration>
  </MediaTime>
  <AudioDescriptor xsi:type="SilenceType" minDurationRef="shOne"
                   confidence="0.5"/>
</AudioSegment>
```


6 High Level Tools

6.1 Introduction

This section contains descriptors and description schemes which are broadly classed as "high level"; that is, they are either structural or application oriented. In some cases, they use tools defined in the audio framework section of this document. Much use is made of tools defined in the multimedia description schemes part of the standard.

The tools in this section cover a wide range of application areas and functionalities. They both provide functionality and serve as examples of how to use the low level framework.

6.2 Audio Signature

6.2.1 Introduction

The `AudioSignatureDS` is a condensed representation of an audio signal designed to provide a unique content identifier for the purpose of robust automatic identification of audio signals. The `AudioSignatureDS` uses statistical data summarization on a series of values of the `AudioSpectrumFlatnessType` to determine the signature.

6.2.2 AudioSignatureType

6.2.2.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of AudioSignature DS                                  -->
<!-- ################################################################ -->
<complexType name="AudioSignatureType">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="Flatness" type="mpeg7:AudioSpectrumFlatnessType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.2.2.2 Semantics

Name	Definition
AudioSignatureType	A structure containing a condensed representation as a unique content identifier for an audio signal for the purpose of robust automatic identification of audio signals (contains statistical summarization of data of AudioSpectrumFlatnessType)
Flatness	The spectrum flatness of the signal of AudioSpectrumFlatnessType data.

6.2.3 Instantiation requirements

In order to constitute a valid AudioSignature description, the following requirements have to be satisfied:

— The syntax of the Flatness part is restricted to SeriesOfVectorBinaryType.

— Both the Mean and the Variance fields of the Flatness part have to be instantiated, as provided by the SeriesOfVectorBinaryType syntax.

— The Scaling ratio (decimation factor), as provided by the SeriesOfVectorBinaryType syntax, must assume values between 2 and 128. The default value is 32.

— The loEdge parameter, as provided by the AudioSpectrumFlatnessType syntax, is fixed at 250 Hz.

— The hiEdge parameter, as provided by the AudioSpectrumFlatnessType syntax, must be at least 500 Hz. The default value is 4000 Hz.

6.2.4 Usage and examples (informative)

There are numerous examples of applications for the AudioSignature description scheme conceivable, including automatic identification of an unknown piece of audio based on a database of registered audio items. This is done by extracting AudioSignature descriptions from both the reference items and the item to be identified. The AudioSignature description of the item to be identified is then matched to all previously registered AudioSignature descriptions in the database. The best matching reference AudioSignature description is the most likely candidate to correspond to the unknown signal. A measure of confidence can be calculated from the matching error of the pair.

In a wider sense, the AudioSignature structure may be used to identify corresponding MPEG-7 descriptions for audio items which are delivered in formats not including descriptive data (“linking of legacy format audio data to MPEG-7 descriptions”). To this end, the MPEG-7 descriptions available at some server have to include an AudioSignature description of each described item. Again, an AudioSignature is extracted from the unknown audio item and the correct set of descriptive data is identified by matching the extracted AudioSignature to the registered signatures.

The signature essentially consists of a statistical summarization of frame by frame AudioSpectrumFlatness low level descriptor values over a period of time. This is obtained by using the mean and variance of the LLD values, as provided by the generic SeriesOfVectors construct with selectable degrees of decimation, and thus temporal resolution. Consequently, signatures may be rescaled (scaled down) in their temporal resolution according to the standard SeriesOfVectors scaling procedures as desired, e.g. in order to achieve a compatible temporal resolution between two signatures.

In addition, a second dimension of AudioSignature scalability is provided by the number of frequency bands present in the AudioSpectrumFlatness field of the signature descriptions. While signatures may provide different numbers of frequency bands, a meaningful comparison between them is always possible for the bands common to the compared signatures, since these relate to common fixed band definitions.

The combination of temporal and frequency band scalability provides a flexible trade-off between the compactness of the signatures and their ability to discriminate between many different audio stimuli. Please note that much more compact (but equivalent) representations of the AudioSignature can be derived in an application-specific context by converting the description to a condensed binary representation with an appropriate numeric precision. While the details of such formats are outside the scope of this standard, the explicit DDL description acts as the basic point of interoperability between these formats.

In order to match two AudioSignature descriptions, a standard mean square distance metric may be used for evaluating the degree of similarity between two signatures (after normalizing the feature variables to unit standard deviation).

6.3 Timbre

6.3.1 Introduction

Timbre Descriptors aim to describe perceptual features of instrument sounds. Timbre is currently defined in the literature as the perceptual features that make two sounds having the same pitch and loudness sound different. The aim of the Timbre DS is to describe these perceptual features with a reduced set of descriptors. The descriptors relate to notions such as “attack”, “brightness” or “richness” of a sound.

Families of sounds and the three TimbreType:

“Timbre similarity” refers to the way human listeners consider two sounds as close whatever classes they belong to and whatever pitch and loudness. Human perception of similarity between sounds is a rather complex mechanism, which involves taking into account several parameters that determine several perceptual dimensions in a possibly complex way. Human listeners will use different perceptual dimensions depending on the family of sounds being heard. For example, the harmonicity feature may be used to distinguish harmonic and non-harmonic sounds, but is unlikely to be important to distinguish sounds within each of these families.

For this reason we distinguish four different families of sounds for the monophonic, non-mixed, non-layered instrument sounds. These families correspond to the main perceptual categories of musical sounds distinguished by the following factors (see Table 3):

— Harmonic: relates to the property of periodicity of a signal (the harmonicity relations between the components of the spectrum a signal; distinguishes harmonic from inharmonic and noisy signals).

— Sustained: relates to the duration of excitation of the sound source (distinguishes sustained from impulsive signals).

— Coherent: relate to the temporal behaviour of the spectral component of a signal (distinguishes frequency spectra with prominent components from noisy spectra).

Table 3 — Sound families

Sound families	Harmonic Sounds	Inharmonic Sounds	Percussive Sounds	Non-coherent Sounds
Sounds characteristics	Sustained Harmonic Coherent	Sustained Non-Harmonic Coherent	Non-Sustained	Sustained Non-Coherent
Examples of sounds belonging to the family	violin, flute, …	bell, triangle	snare, claves, …	cymbals, white noise, …
TimbreType	Harmonic Instrument TimbreType		Percussive Instument TimbreType	

A reduced set of descriptors is established in order to describe the timbre perception within each family of sounds. So far, only two families are considered: the sustained, harmonic coherent sounds, and the non-sustained sounds.

For each of these families, a set of `TimbreDescriptors` is established in order to describe the timbre perception between sounds belonging to the same family. These sets are called the `HarmonicInstrumentTimbreType` and `PercussiveInstrumentTimbreType`.

Because some sounds may belong to other Timbre Families, a generic set of Timbre Descriptors is included with all of the `TimbreDescriptors` of the `HarmonicInstrumentTimbreType` and `PercussiveInstrumentTimbreType`.

Target applications are:

— Authoring Tools for sound designers or musicians (Music Sample database management) and

— Retrieval Tools for producers (“Query by example search” based on perceptual features).

6.3.2 InstrumentTimbreType

6.3.2.1 Introduction

The `InstrumentTimbreType` is a set of `TimbreDescriptors` established in order to describe the timbre perception among sounds belonging simultaneously to the Harmonic and Percussive sound families. An example of such a sound is that produced by a harp.

6.3.2.2 Syntax

```
<!-- ################################################################ -->
<!-- Definition of InstrumentTimbre DS                                -->
<!-- ################################################################ -->
<complexType name="InstrumentTimbreType">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="LogAttackTime" type="mpeg7:LogAttackTimeType"
                 minOccurs="0"/>
        <element name="HarmonicSpectralCentroid"
                 type="mpeg7:HarmonicSpectralCentroidType" minOccurs="0"/>
        <element name="HarmonicSpectralDeviation"
                 type="mpeg7:HarmonicSpectralDeviationType" minOccurs="0"/>
        <element name="HarmonicSpectralSpread"
                 type="mpeg7:HarmonicSpectralSpreadType" minOccurs="0"/>
        <element name="HarmonicSpectralVariation"
                 type="mpeg7:HarmonicSpectralVariationType" minOccurs="0"/>
        <element name="SpectralCentroid" type="mpeg7:SpectralCentroidType"
                 minOccurs="0"/>
        <element name="TemporalCentroid" type="mpeg7:TemporalCentroidType"
                 minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.3.2.3 Semantics

Name	Definition
LogAttackTime (LAT)	A LogAttackTime Descriptor
HarmonicSpectralCentroid (HSC)	A HarmonicSpectralCentroid Descriptor
HarmonicSpectralDeviation (HSD)	A HarmonicSpectralDeviation Descriptor
HarmonicSpectralSpread (HSS)	A HarmonicSpectralSpread Descriptor
HarmonicSpectralVariation (HSV)	A HarmonicSpectralVariation Descriptor
SpectralCentroid (SC)	A SpectralCentroid Descriptor
TemporalCentroid (TC)	A TemporalCentroid Descriptor

6.3.3 HarmonicInstrumentTimbreType

6.3.3.1 Introduction

The HarmonicInstrumentTimbreType is a set of TimbreDescriptors established in order to describe the timbre perception among sounds belonging to Harmonic sound family. An example is the sound of a violin.

6.3.3.2 Syntax

```
<!-- ################################################################ -->
<!-- Definition of HarmonicInstrumentTimbre DS                        -->
<!-- ################################################################ -->
<complexType name="HarmonicInstrumentTimbreType">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="LogAttackTime" type="mpeg7:LogAttackTimeType"/>
        <element name="HarmonicSpectralCentroid"
                 type="mpeg7:HarmonicSpectralCentroidType"/>
        <element name="HarmonicSpectralDeviation"
                 type="mpeg7:HarmonicSpectralDeviationType"/>
        <element name="HarmonicSpectralSpread"
                 type="mpeg7:HarmonicSpectralSpreadType"/>
        <element name="HarmonicSpectralVariation"
                 type="mpeg7:HarmonicSpectralVariationType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.3.3.3 Semantics

Name	Definition
`LogAttackTime (LAT)`	A `LogAttackTime` Descriptor
`HarmonicSpectralCentroid (HSC)`	A `HarmonicSpectralCentroid` Descriptor
`HarmonicSpectralDeviation (HSD)`	A `HarmonicSpectralDeviation` Descriptor
`HarmonicSpectralSpread (HSS)`	A `HarmonicSpectralSpread` Descriptor
`HarmonicSpectralVariation (HSV)`	A `HarmonicSpectralVariation` Descriptor

6.3.4 PercussiveInstrumentTimbreType

6.3.4.1 Introduction

The `PercussiveInstrumentTimbreType` is a set of `TimbreDescriptors` established in order to describe the timbre perception among sounds belonging to Percussive sound family. An example is the sound of a drum.

6.3.4.2 Syntax

```
<!-- ################################################################ -->
<!-- Definition of PercussiveInstrumentTimbre DS                      -->
<!-- ################################################################ -->
<complexType name="PercussiveInstrumentTimbreType">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="LogAttackTime" type="mpeg7:LogAttackTimeType"/>
        <element name="SpectralCentroid" type="mpeg7:SpectralCentroidType"/>
        <element name="TemporalCentroid" type="mpeg7:TemporalCentroidType"/>
```

```
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.3.4.3 Semantics

Name	Definition
`LogAttackTime (LAT)`	A `LogAttackTime` Descriptor
`SpectralCentroid (SC)`	A `SpectralCentroid` Descriptor
`TemporalCentroid (TC)`	A `TemporalCentroid` Descriptor

6.3.5 Usage, extraction and examples (informative)

6.3.5.1 Distance measures

Timbre descriptors can be combined in the following suggested way in order to allow a comparison of sounds according to perceptual features:

For sound family 1 one obtains the following expression for the distance:

$$dist = \sqrt{8(\Delta LAT)^2 + 3e-5(\Delta HSC)^2 + 3e-4(\Delta HSD)^2 + (10\Delta HSS - 60\Delta HSV)^2}$$

For sound family 3 one obtains the following expression for the distance:

$$dist = \sqrt{(-0.3\Delta LAT - 0.6\Delta TC)^2 + (-1e-4\Delta SC)^2}$$

In both cases, Δ is the difference between the values of the same acoustical parameter for the two sounds considered.

It should be noted that the exact coefficients may be different from one database to another depending on the set of sounds included. The above coefficients are approximations to values derived from a specific database used for experimental verification of the descriptors; they are provided only as an example for informative purposes, and may not be appropriate for arbitrary data sets.

6.3.5.2 Example of InstrumentTimbre

This example represents the sound of a harp.

```
<AudioDescriptionScheme xsi:type="InstrumentTimbreType">
  <LogAttackTime>
    <Scalar>-1.660812</Scalar>
  </LogAttackTime>
  <HarmonicSpectralCentroid>
    <Scalar>698.586713</Scalar>
  </HarmonicSpectralCentroid>
  <HarmonicSpectralDeviation>
    <Scalar>-0.014473</Scalar>
  </HarmonicSpectralDeviation>
  <HarmonicSpectralSpread>
    <Scalar>0.345456</Scalar>
```

```
    </HarmonicSpectralSpread>
    <HarmonicSpectralVariation>
      <Scalar>0.015437</Scalar>
    </HarmonicSpectralVariation>
    <SpectralCentroid>
      <Scalar>867.486074</Scalar>
    </SpectralCentroid>
    <TemporalCentroid>
      <Scalar>0.231309</Scalar>
    </TemporalCentroid>
  </AudioDescriptionScheme>
```

6.3.5.3 Example of HarmonicInstrumentTimbre

This example represents the sound of a violin.

```
  <AudioDescriptionScheme xsi:type="HarmonicInstrumentTimbreType">
    <LogAttackTime>
      <Scalar>-0.150702</Scalar>
    </LogAttackTime>
    <HarmonicSpectralCentroid>
      <Scalar>1586.892383</Scalar>
    </HarmonicSpectralCentroid>
    <HarmonicSpectralDeviation>
      <Scalar>-0.027864</Scalar>
    </HarmonicSpectralDeviation>
    <HarmonicSpectralSpread>
      <Scalar>0.550866</Scalar>
    </HarmonicSpectralSpread>
    <HarmonicSpectralVariation>
      <Scalar>0.001877</Scalar>
    </HarmonicSpectralVariation>
  </AudioDescriptionScheme>
```

6.3.5.4 Example of PercussiveInstrumentTimbre

This example represents the sound of a side drum.

```
  <AudioDescriptionScheme xsi:type="PercussiveInstrumentTimbreType">
    <LogAttackTime>
      <Scalar>-1.683017</Scalar>
    </LogAttackTime>
    <SpectralCentroid>
      <Scalar>1217.341518</Scalar>
    </SpectralCentroid>
    <TemporalCentroid>
      <Scalar>0.081574</Scalar>
    </TemporalCentroid>
  </AudioDescriptionScheme>
```


6.4 General Sound Recognition and Indexing

6.4.1 Introduction

The tools defined in this section support applications in general audio classification and content indexing. For example, automatic classification and segmentation of audio into broad classes such as *speech*, *music,* and *background* or into narrower classes such as *male, female, laughter*, *telephones, reggae*, *classical* or *violin*. The description schemes consist of sound models that are based on the `ContinuousHiddenMarkovModel DS` and `ProbabilityClassificationModel` DS defined in ISO/IEC 15938 part 5.

In addition to automatic classification, audio segments may be indexed using the series of states generated by a sound model (`SoundModel D`). The pattern of states through time is used to index audio and retrieve similar audio segments by matching query and target state activation patterns. The `SoundModelStatePath` D and `SoundModelStateHistogram` D encapsulate this functionality.

6.4.2 SoundModelType

The `SoundModel` DS contains a sound class label and a continuous hidden Markov model (CHMM) that is used for automatic classification and indexing of audio segments, see Figure 15. Hidden Markov model parameters are calculated using well-known learning algorithms, such as the Baum-Welch algorithm, operating on a training set of sound data. Once trained, a hidden Markov model can be used to compare new sounds with the model to determine the goodness of fit.

The default descriptor for sound classification is `AudioSpectrumProjection`. A set of basis functions is stored with each model that are used to calculate the spectrum projection of audio segments, see the definition of `AudioSpectrumBasis` D and `AudioSpectrumProjection` D above. Other descriptors may also be used with the `SoundModel` DS; their use is signaled by the `DescriptorModel` DS contained within the `ContinuousHiddenMarkovModel` DS and `SoundClassificationModel` DS.

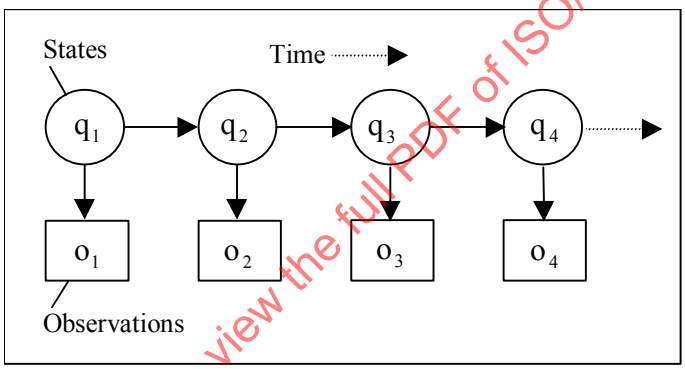

Figure 15 — The `SoundModel` DS consists of a hidden Markov model that generates a sequence of hidden states, *q*. Each state generates an observation, *o*, for each time step, *t*. The model contains state transition probabilities as well as parameters for the observation probability distributions

6.4.2.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SoundModel DS                                      -->
<!-- ################################################################ -->
<complexType name="SoundModelType">
  <complexContent>
    <extension base="mpeg7:ContinuousHiddenMarkovModelType">
      <sequence minOccurs="0">
        <element name="SoundClassLabel" type="mpeg7:TermUseType"/>
        <element name="SpectrumBasis" type="mpeg7:AudioSpectrumBasisType"/>
      </sequence>
      <attribute name="SoundModelRef" type="anyURI" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

6.4.2.2 Semantics

Name	Definition
SoundModelType	Description scheme containing parameters to a Continuous Hidden Markov Model (CHMM), sound class labels or references, extraction metadata and spectral basis functions for the sound class. The hidden Markov model (HMM) consists of three components, $\theta_j = \{A_j, B_j, \pi_j\}$; corresponding to the Initial state distribution, $\pi_i = P(q_1 = i)$ with $q_t \in \{1 \ldots K\}$, the state Transitions matrix $A_{ij} = P(q_t = j \mid q_{t-1} = i)$, and an ObservationDistributionType: $b_j(\mathbf{y}) = P(\mathbf{y} \mid q_t = j)$, defined for each state. The initial state distribution and transition probabilities characterize the dynamic behavior of the states through time. This DS extends ContinuousHiddenMarkovModelType defined in ISO/IEC 15938 part 5.
SoundClassLabel	A unique label, or reference to a label from a classification scheme, that describes the sound class of the model. See mpeg7:TermUseType in ISO/IEC 15938 part 5.
SpectrumBasis	Data-derived basis functions for the sound class. See definition of AudioSpectrumBasis.
SoundModelRef	Optional reference to a SoundModel pointing to an instance that provides the model definition.

6.4.2.3 Example (informative)

The following example is an instance of the SoundModel DS defining a model of Trumpet sounds. The model uses a HMM with states that are set to the GaussianDistributionType with mean $\boldsymbol{\mu}_j$ and covariance matrix $\mathbf{K}_j$, giving $B_j = \{\boldsymbol{\mu}_j, \mathbf{K}_j\}$ for state *j*.

```
<SoundModel id="IDInstrument:Trumpet">
  <SoundClassLabel>
    <Term id="ID16">Instrument:Trumpet</Term>
  </SoundClassLabel>
  <Initial dim="1 6"> 0.000 0.068 0.074 0.716 0.142 0.000 </Initial>
  <Transitions dim="6 6">
    1.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.994 0.000 0.000 0.000 0.006
    0.000 0.000 0.993 0.007 0.000 0.000
    0.014 0.000 0.095 0.818 0.000 0.074
    0.000 0.000 0.000 0.005 0.995 0.000
    0.056 0.000 0.000 0.000 0.000 0.944
  </Transitions>
  <DescriptorModel>
    <Descriptor xsi:type="mpeg7:AudioSpectrumProjectionType"/>
    <Field>SeriesOfVector</Field>
  </DescriptorModel>
```

```
    <State>
      <Label>
        <Term id="IDState1">State1</Term>
      </Label>
      <ObservationDistribution xsi:type="mpeg7:GaussianDistributionType">
        <Mean dim="1 10"> 8.004 -4.805 4.850 5.738 1.261 -2.198 2.076 -0.324
              -2.052 -0.022 </Mean>
        <Covariance dim="10 10">
          0.744 0.008 2.526 0.324 -0.049 -0.297 0.159 -0.074 -0.260 0.029
          0.008 1.387 -1.087 1.906 0.838 -0.134 0.640 -0.321 -0.113 -0.053
          2.526 -1.087 12.525 0.026 0.012 -0.002 0.009 -0.004 -0.002 -0.001
          0.324 1.906 0.026 6.004 -0.020 0.003 -0.015 0.008 0.003 0.001
          -0.049 0.838 0.012 -0.020 4.870 0.001 -0.007 0.003 0.001 0.001
          -0.297 -0.134 -0.002 0.003 0.001 3.402 0.001 -0.001 -0.000 -0.000
          0.159 0.640 0.009 -0.015 -0.007 0.001 3.157 0.003 0.001 0.000
          -0.074 -0.321 -0.004 0.008 0.003 -0.001 0.003 1.816 -0.000 -0.000
          -0.260 -0.113 -0.002 0.003 0.001 -0.000 0.001 -0.000 0.923 -0.000
          0.029 -0.053 -0.001 0.001 0.001 -0.000 0.000 -0.000 -0.000 0.494
        </Covariance>
      </ObservationDistribution>
    </State>
    <!-- Remaining States similar to above . . .                          -->
    <SpectrumBasis loEdge="62.5" hiEdge="8000" octaveResolution="1/4">
      <SeriesOfVector totalNumOfSamples="1" vectorSize="31 9">
        <Raw mpeg7:dim="31 9">
          0.082 -0.026 0.024 -0.093 0.010 -0.021 0.063 -0.103 0.057
          0.291 0.073 0.025 -0.039 0.026 -0.086 0.185 0.241 0.107
          0.267 0.062 0.030 -0.026 0.054 -0.115 0.171 0.266 0.240
          0.267 0.062 0.030 -0.026 0.054 -0.115 0.171 0.266 0.240
          0.271 -0.008 0.039 0.007 0.119 -0.067 0.033 0.165 0.175
          0.271 -0.008 0.039 0.007 0.119 -0.067 0.033 0.165 0.175
          0.269 -0.159 0.062 0.074 0.182 0.071 -0.194 0.054 -0.009
          0.246 -0.306 0.048 0.148 0.199 0.163 -0.324 -0.048 -0.065
          0.216 -0.356 -0.037 0.137 0.059 0.215 -0.242 -0.035 -0.052
          0.187 -0.359 -0.183 0.067 -0.343 0.223 0.023 -0.002 0.000
          <!-- Remaining values here . . .                               -->
        </Raw>
      </SeriesOfVector>
    </SpectrumBasis>
  </SoundModel>
```

6.4.3 SoundClassificationModelType

This DS combines a set of sound models into a multi-way classifier for automatic labeling of audio segments using terms from a classification scheme. Probabilistic classifiers can recognize broad sound classes, such as *speech* and *music*, or they can be trained to identify narrower content categories such as *male, female, trumpet* or *violin*. Other applications include music genre classification and voice recognition. See Figure 16 for an example classification scheme for general audio content indexing. For more information on classification schemes see the `ClassificationScheme` DS defined in ISO/IEC 15938 part 5.

6.4.3.1 Syntax

```
  <!-- ################################################################ -->
  <!-- Definition of SoundClassificationModel DS                        -->
  <!-- ################################################################ -->
  <complexType name="SoundClassificationModelType">
    <complexContent>
      <extension base="mpeg7: ClassificationModelType">
        <sequence>
          <element name="SoundModel" type="mpeg7:SoundModelType"
```

```
                        minOccurs="1" maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.4.3.2 Semantics

Name	Definition
SoundClassificationModelType	A collection of sound models that are used for automatic classification and indexing of audio.
SoundModel	A sequence of SoundModel DS instances that define the model choices for the classifier.

6.4.3.3 Example

The example below shows a number of SoundClassificationModel DS instances corresponding to the classification scheme shown in Figure 16. The classifiers are organized hierarchically with higher level classes pre-selecting the classifiers for low-level classes.

```
<AudioDescriptionScheme xsi:type="SoundClassificationModelType"
                        id="IDClassifier:GeneralAudio">
  <SoundModel SoundModelRef="IDAnimals"/>
  <SoundModel SoundModelRef="IDMusic"/>
  <SoundModel SoundModelRef="IDPeople"/>
  <SoundModel SoundModelRef="IDFoley"/>
</AudioDescriptionScheme>
<AudioDescriptionScheme xsi:type="SoundClassificationModelType"
                        id="IDClassifier:Animals">
  <SoundModel SoundModelRef="IDAnimals:BirdCalls"/>
  <SoundModel SoundModelRef="IDAnimals:DogBarks"/>
</AudioDescriptionScheme>
<AudioDescriptionScheme xsi:type="SoundClassificationModelType"
                        id="IDClassifier:Music">
  <SoundModel SoundModelRef="IDInstrument:AltoFlute"/>
  <SoundModel SoundModelRef="IDInstrument:Bosendorfer"/>
  <SoundModel SoundModelRef="IDInstrument:Cello"/>
  <SoundModel SoundModelRef="IDInstrument:EnglishHorn"/>
  <SoundModel SoundModelRef="IDInstrument:Guitar"/>
  <SoundModel SoundModelRef="IDInstrument:Trumpet"/>
  <SoundModel SoundModelRef="IDInstrument:Violins"/>
</AudioDescriptionScheme>
<AudioDescriptionScheme xsi:type="SoundClassificationModelType"
                        id="IDClassifier:People">
  <SoundModel SoundModelRef="IDSpeech:Male"/>
  <SoundModel SoundModelRef="IDSpeech:Female"/>
  <SoundModel SoundModelRef="IDCrowds:Applause"/>
  <SoundModel SoundModelRef="IDPeople:FootSteps"/>
  <SoundModel SoundModelRef="IDPeople:Laughter"/>
  <SoundModel SoundModelRef="IDPeople:ShoeSqueaks"/>
</AudioDescriptionScheme>
<AudioDescriptionScheme xsi:type="SoundClassificationModelType"
                        id="IDClassifier:Foley">
  <SoundModel SoundModelRef="IDTelephones"/>
  <SoundModel SoundModelRef="IDGunshots:Pistols"/>
  <SoundModel SoundModelRef="IDExplosions"/>
  <SoundModel SoundModelRef="IDGlass:Smashes"/>
</AudioDescriptionScheme>
```

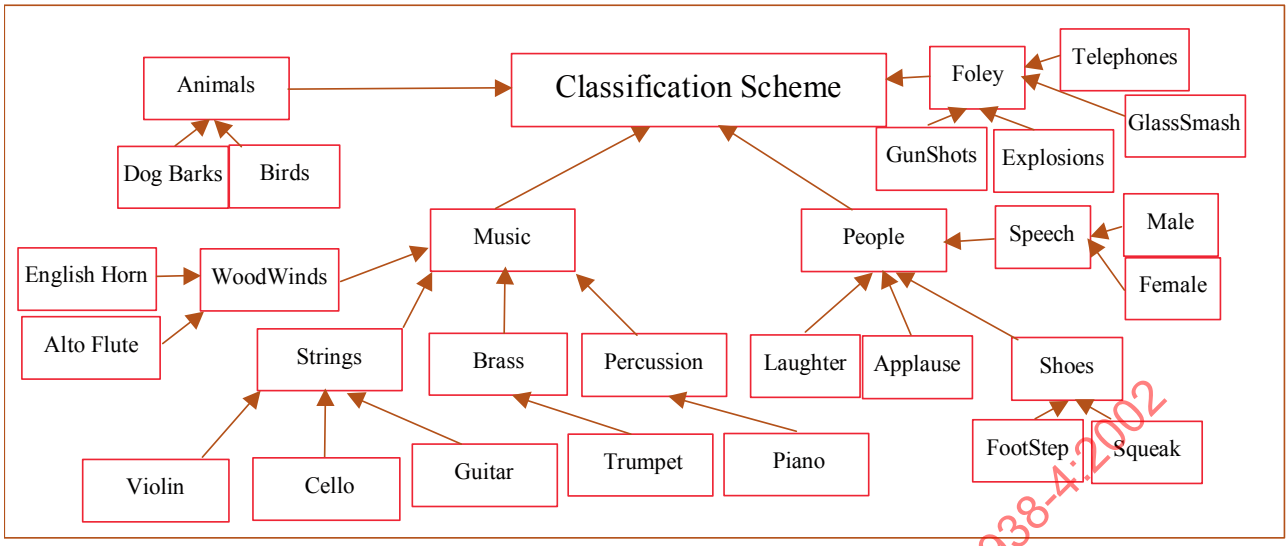

Figure 16 — Example Classification Scheme for General Audio Content

6.4.4 SoundModelStatePathType

This descriptor consists of the sequence of states generated by a `SoundModel` given an audio segment. A series of state indices, that reference continuous hidden Markov model states from a `SoundModel`, is stored using the `AudioLLDScalar D` within the descriptor.

6.4.4.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SoundModelStatePath DS                             -->
<!-- ################################################################ -->
<complexType name="SoundModelStatePathType">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="StatePath" type="mpeg7:AudioLLDScalarType"/>
        <element name="SoundModelRef" type="anyURI"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.4.4.2 Semantics

Name	Definition
`SoundModelStatePathType`	Describes the series of states generated by a `SoundModel` for a given audio segment
`StatePath`	Regularly sampled series of state indices described as integers ranging from 1..K, where K is the number of states. The value represents the ordinality of a state from a hidden Markov model.
`SoundModelRef`	Reference to the `SoundModel` DS instance that generated the given `StatePath`.

6.4.4.3 **Extraction** (informative)

The Viterbi algorithm generates the most likely state path given a sequence of observations and a hidden Markov model. For details on the Viterbi algorithm see the usage section on automatic audio classification below. The output is a sequence of state indices, one for each input vector. Figure 17 shows the `AudioSpectrumEnvelope` of a dog barking and the resulting `SoundModelStatePath` sequence generated by a dog bark `SoundModel`.

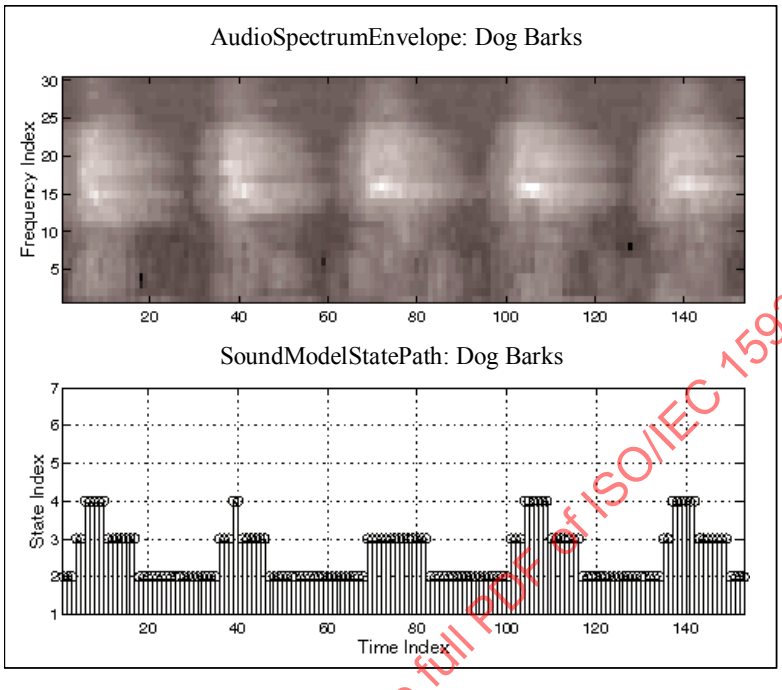

Figure 17 — Spectrum of a dog barking and and the state-path generated by a `SoundModel`

6.4.4.4 Example

```
<AudioDescriptor xsi:type="SoundModelStatePathType">
  <StatePath>
    <SeriesOfScalar totalNumOfSamples="154" hopSize="PT10N1000F">
      5 5 5 5 5 6 6 6 6 3 3 3 3 3 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
      5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6
      6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
      5 5 5 5 6 6 6 6 6 6 6 6 3 3 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 5
      5 5 5 5 5 5 5 5 5 5 5 5 5 5
    </SeriesOfScalar>
  </StatePath>
  <SoundModelRef>IDDogBarks</SoundModelRef>
</AudioDescriptor>
```

6.4.5 SoundModelStateHistogramType

This descriptor consists of a normalized histogram of the state sequence generated by a `SoundModel`. The descriptor is used to compare sound segments using histograms of their state activation patterns. For an example matching algorithm and applications see the informative usage section below.

6.4.5.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SoundModelStateHistogram D                         -->
<!-- ################################################################ -->
<complexType name="SoundModelStateHistogramType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <sequence minOccurs="1" maxOccurs="unbounded">
          <element name="StateRef" type="anyURI"/>
          <element name="RelativeFrequency" type="mpeg7:nonNegativeReal"/>
        </sequence>
        <element name="SoundModelRef" type="anyURI"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.4.5.2 Semantics

Name	Definition
SoundModelStateHistogramType	Normalized histogram of the state sequence generated by a SoundModel over a given audio segment.
StateRef	Reference to a unique identifier for each state in a continuous hidden Markov model. For more information on states see FiniteStateModelType defined in ISO/IEC 15938 part 5.
RelativeFrequency	Relative frequency of a state in an audio segment. Frequencies are normalized counts in the range 0..1 obtained by dividing the counts for each state by the total number of samples in the state sequence: $hist_a(j) = \frac{N(j)}{\sum_{i=1}^{K} N(i)}, 1 \le j \le K,$ where N(j) is the count (frequency) for state j for a given audio segment.
SoundModelRef	Reference to the SoundModel DS instance used to generate the state histogram.

6.4.5.3 Usage (informative)

One similarity measure for SoundModelStateHistogram descriptions generated by the same SoundModel is the sum-of-square-errors (SSE):

$$\delta(a,b) = \sum_{j=1}^{k} (hist_a(j) - hist_b(j))^2$$

This distance metric will be zero if the two histograms are equivalent and will be non-zero if they are different with higher values indicating a greater degree of dissimilarity.

6.4.5.4 Example

```
<AudioDescriptor xsi:type="SoundModelStatePathType">
  <SoundModelRef>IDDogBarks</SoundModelRef>
  <StateRef>IDState1</StateRef>
  <RelativeFrequency> 0.000</RelativeFrequency>
  <StateRef>IDState2</StateRef>
  <RelativeFrequency> 0.000</RelativeFrequency>
  <StateRef>IDState3</StateRef>
  <RelativeFrequency> 0.045</RelativeFrequency>
  <StateRef>IDState4</StateRef>
  <RelativeFrequency> 0.000</RelativeFrequency>
  <StateRef>IDState5</StateRef>
  <RelativeFrequency> 0.442</RelativeFrequency>
  <StateRef>IDState6</StateRef>
  <RelativeFrequency> 0.513</RelativeFrequency>
</AudioDescriptor>
```

6.4.6 General Sound Classification and Indexing Applications (informative)

The following sections outline two applications of the sound classification and indexing tools described above.

6.4.6.1 Automatic Audio Classification

In the first example the `SoundClassificationModel` DS is used to classify audio segments into categories from a classification scheme. Figure 16 shows an example classification scheme consisting of *Animals, Music, People,* and *Foley* (Background) at the highest levels and related sub-categories at the lowest levels of the tree.

6.4.6.1.1 The Viterbi Algorithm

Automatic classification of audio uses a collection of hidden Markov models, category labels and basis functions defined by `SoundModel` DS instances and collected into a `SoundClassificationModel` DS. Here, the Viterbi algorithm is used to compute the most likely state sequence for each model in the classifier given the observed data. The algorithm consists of the following steps where $\theta_j = \{A_j, B_j, \pi_j\}$ are the hidden Markov model parameters for model *j*, M is the number of observation vectors for the given audio segment, *K* is the number of states in the given sound model, o_t is an observation vector at time *t* and q_t is the state index at time *t* :

a) *Preprocessing of* $\theta_j = \{A_j, B_j, \pi_j\}$.

$$\tilde{\pi}_i = \log(\pi_i) \qquad 1 \le i \le K$$

$$\tilde{b}_i(\mathrm{o}_t) = \log(b_i(\mathrm{o}_t)), \quad 1 \le i \le K, \quad 1 \le t \le M$$

$$\tilde{a}_{ij} = \log(a_{ij}) \qquad 1 \le i, j \le K$$

b) *Initialization.*

$$\tilde{\delta}_1(i) = \log(\delta_1(i)) = \tilde{\pi}_i + \tilde{b}_i(\mathrm{o}_1), \quad 1 \le i \le K$$

$$\psi_t(i) = 0, \qquad 1 \le i \le K$$

c) *Recursion*

$$\tilde{\delta}_t(j) = \log(\delta_t(j)) = \max_{1 \le i \le K}\left[\tilde{\delta}_{t-1} + \tilde{a}_{ij}\right] + \tilde{b}_j(\mathrm{o}_t)$$

$$\psi_t(j) = \log(\delta_t(j)) = \arg\max_{1 \le i \le K}\left[\tilde{\delta}_{t-1} + \tilde{a}_{ij}\right] + \tilde{b}_j(\mathrm{o}_t), \quad 2 \le t \le M, \quad 1 \le j \le K$$

d) *Termination*

$$\tilde{\mathrm{P}}^* = \max_{1 \le i \le K}\left[\tilde{\delta}_M(i)\right]$$

$$\mathrm{q}_M^* = \arg\max_{1 \le i \le K}\left[\tilde{\delta}_M(i)\right]$$

e) *Backtracking*

$$\mathrm{q}_t^* = \psi_{t+1}\left(\mathrm{q}_{t+1}^*\right), \qquad M - 1 \ge t \ge 1$$

This yields the most likely state path, $Q_j = \{\mathrm{q}_1, \mathrm{q}_2, \cdots, \mathrm{q}_M\}$, for each model *j* given observed data $O = \{\mathrm{o}_1, \mathrm{o}_2, \cdots, \mathrm{o}_M\}$ and hidden Markov model parameters θ_j. The likelihood, $\tilde{\mathrm{P}}_j^*$, of the observed data given each model, *j*, is used to choose the best-fit model amongst *L* competing models such that:

$$j^* = \arg\max_{1 \le j \le L}\left[\mathrm{P}_j^* = P(\mathrm{O}, \mathrm{Q}_j \mid \theta_j)\right],$$

this method completes *maximum likelihood* classification for an HMM classifier, see Figure 18.

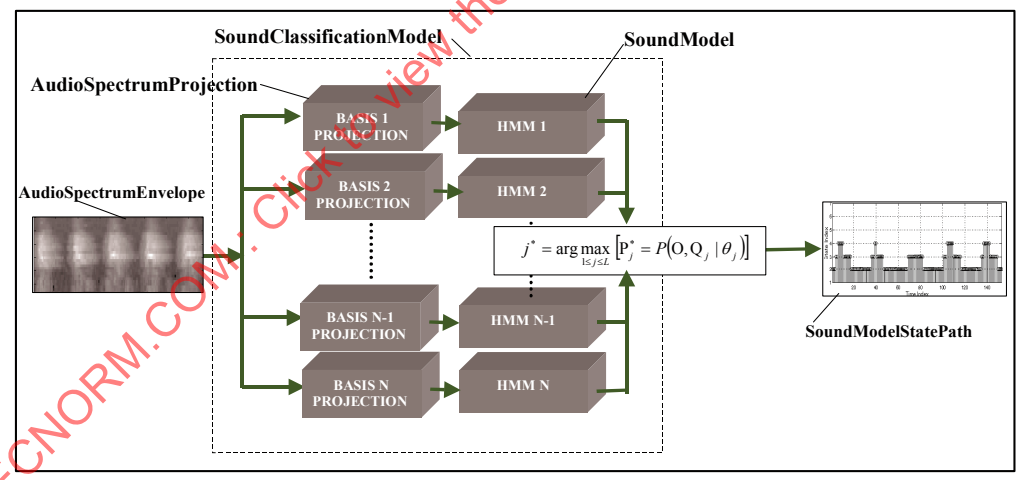

Figure 18 — Use of multiple HMM models for automatic classification and indexing of audio content

6.4.6.2 Audio Query-by-Example

Figure 19 shows a query-by-example application that uses both the `SoundClassificatoinModel` DS and the `SoundModelStateHistogram` D. Given an audio query, the most likely model is selected using automatic classification as described above. The state path generated by the selected model is used to compute a `SoundModelStateHistogram` D. Distances are calculated between the query histogram and a pre-computed database of histogram descriptions using the sum of square errors distance metric described above. These distances are used to sort the results in ascending order thereby yielding the best matches for the given audio query, see Figure 20.

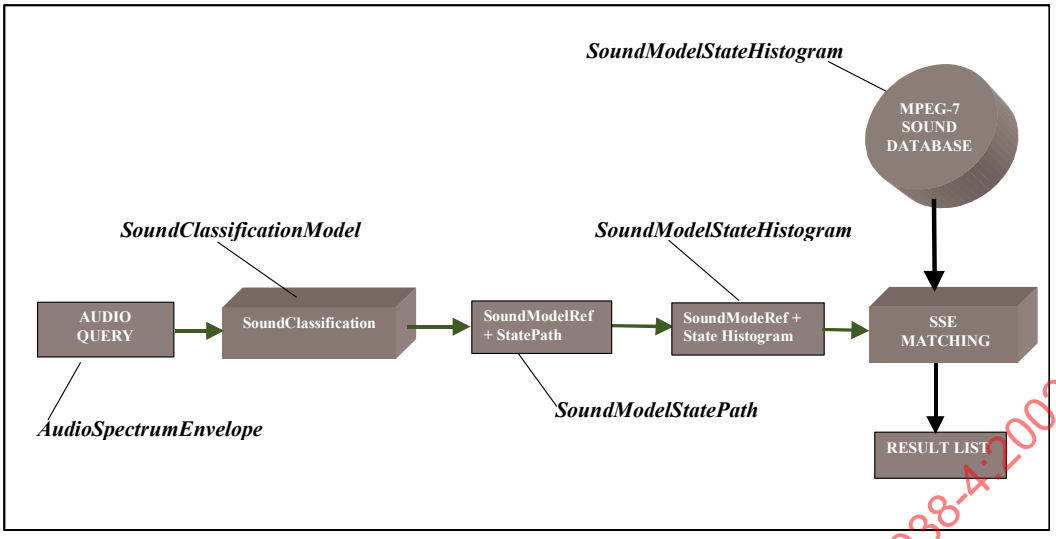

Figure 19 — Audio query-by-example application utilizing a `SoundClassificationModel` DS and `SoundModelStateHistogram` D

Figure 20 — Query-by-example results calculated using the sum of square errors between query and target state histograms

6.5 Spoken Content

6.5.1 Introduction

The Spoken Content DSs are a representation of the output of Automatic Speech Recognition (ASR). The `SpokenContentLatticeType` represents the actual decoding produced by an ASR engine, whereas the `SpokenContentHeaderType` contains information about the recogniser itself and the people (or "Speakers") being recognised.

The Spoken Content DSs consist of combined word and phone lattices for each speaker in an audio stream. By combining the lattices, the problem of out-of-vocabulary (OOV) words is greatly alleviated and retrieval may still be carried out when the original decoding was in error. The DSs can be used for two broad classes of retrieval scenario: indexing into and retrieval of an audio stream, and indexing of multimedia objects annotated with speech. The DSs attempt to be memory efficient while also retaining the flexibility to accommodate currently unforeseen uses.

In the context of the Spoken Content DSs, the word “phone” is used to refer to the sub-word units used in automatic speech recognition.

Example applications include

a) *Recall of audio/video data by memorable spoken events*. An example would be a film or video recording where a character or person spoke a particular word or sequence of words. The source media would be known, and the query would return a position in the media.

b) *Spoken Document Retrieval*. In this case, there is a database consisting of separate spoken documents. The result of the query is the relevant documents, and optionally the position in those documents of the matched speech.

c) *Annotated Media Retrieval*. This is similar to spoken document retrieval, but the spoken part of the media would generally be quite short (a few seconds). The result of the query is the media which is annotated with speech, and not the speech itself. An example is a photograph retrieved using a spoken annotation.

6.5.2 SpokenContentHeaderType

6.5.2.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SpokenContentHeader header                         -->
<!-- ################################################################ -->
<complexType name="SpokenContentHeaderType">
  <complexContent>
    <extension base="mpeg7:HeaderType">
      <sequence>
        <choice minOccurs="1" maxOccurs="unbounded">
          <!-- Information about the word and phone lexicons used to   -->
          <!-- represent the speech                                    -->
          <element name="WordLexicon" type="mpeg7:WordLexiconType"/>
          <element name="PhoneLexicon" type="mpeg7:PhoneLexiconType"/>
        </choice>
        <element name="ConfusionInfo" type="mpeg7:ConfusionCountType"
                 minOccurs="0" maxOccurs="unbounded"/>
        <element name="DescriptionMetadata"
                 type="mpeg7:DescriptionMetadataType" minOccurs="0"/>
        <!-- Information about the speakers in the audio              -->
        <element name="SpeakerInfo" type="mpeg7:SpeakerInfoType" minOccurs="1"
                 maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.5.2.2 Semantics

Name	Definition
SpokenContentHeaderType	Header information for Spoken Content in a description.
WordLexicon	A list of words. For a complete description, see the definition of WordLexiconType.
PhoneLexicon	A list of phones. For a complete description, see the definition of PhoneLexiconType.
ConfusionInfo	A data structure of phone confusion information. Although separate, the confusion information must map onto the phone lexicon with which it is associated via the SpeakerInfo.
DescriptionMetadata	Information about the extraction process used to generate the lattice(s). Specifically, this data structure can store the name and settings of the speech recognition engine used. For more information see ISO/IEC 15938 part 5.
SpeakerInfo	Information about the speakers, ie. the people speaking in the audio.

6.5.3 SpeakerInfoType

6.5.3.1 Syntax

```
<!-- ################################################################## -->
<!-- Definition of SpeakerInfo header                                   -->
<!-- ################################################################## -->
<complexType name="SpeakerInfoType">
  <complexContent>
    <extension base="mpeg7:HeaderType">
      <sequence>
        <element name="SpokenLanguage" type="language"/>
        <element name="Person" type="mpeg7:PersonType" minOccurs="0"/>
        <element name="WordIndex" minOccurs="0">
          <complexType>
            <sequence>
              <element name="WordIndexEntry" maxOccurs="unbounded">
                <complexType>
                  <sequence>
                    <element name="IndexEntry"
                             type="mpeg7:SpokenContentIndexEntryType"
                             minOccurs="1" maxOccurs="unbounded"/>
                  </sequence>
                  <attribute name="key" use="required">
                    <simpleType>
                      <list itemType="mpeg7:WordLexiconIndexType"/>
                    </simpleType>
                  </attribute>
                </complexType>
              </element>
            </sequence>
            <attribute name="defaultLattice" type="anyURI" use="required"/>
          </complexType>
        </element>
```

```
        <element name="PhoneIndex" minOccurs="0">
          <complexType>
            <sequence>
              <element name="PhoneIndexEntry" maxOccurs="unbounded">
                <complexType>
                  <sequence>
                    <element name="IndexEntry"
                             type="mpeg7:SpokenContentIndexEntryType"
                             minOccurs="1" maxOccurs="unbounded"/>
                  </sequence>
                  <attribute name="key" use="required">
                    <simpleType>
                      <list itemType="mpeg7:PhoneLexiconIndexType"/>
                    </simpleType>
                  </attribute>
                </complexType>
              </element>
            </sequence>
            <attribute name="defaultLattice" type="anyURI" use="required"/>
          </complexType>
        </element>
      </sequence>
      <attribute name="phoneLexiconRef" type="anyURI" use="optional"/>
      <attribute name="wordLexiconRef" type="anyURI" use="optional"/>
      <attribute name="confusionInfoRef" type="anyURI" use="optional"/>
      <attribute name="descriptionMetadataRef" type="anyURI" use="optional"/>
      <attribute name="provenance" use="required">
        <simpleType>
          <restriction base="NMTOKEN">
            <enumeration value="unknown"/>
            <enumeration value="ASR"/>
            <enumeration value="manual"/>
            <enumeration value="keyword"/>
            <enumeration value="parsing"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
```

6.5.3.2 Semantics

Name	Definition
SpeakerInfoType	Speaker information for a speaker in a Spoken Content description scheme. This is actually more of a collection point for information about a lattice. It contains a "Person DS" element representing the person who is speaking, but also contains indexes and references to confusion information and lexicons.
SpokenLanguage	The language which the speaker is speaking. This is distinct from the language in which the description is written. It is implicitly assumed that the word and phone lexicons are applicable for the spoken language.
DescriptionMetadata	Information about the extraction process, and any settings that may be pertinent to the retrieval process. For more information, see part 5.
Person	An individual person who is speaking. This field is optional in that the Person may be unknown, but can be used to store the name of the speaker.

Name	Definition
`WordIndex`	An Index for the words in the lattice. The index consists of a list of words or word "n-grams" (sequences of consecutive words), together with pointers to where each word or word n-gram occurs in the lattices. The design is such that each speaker has a single word index.
`WordIndexEntry`	An entry in the word index
`IndexEntry`	A lattice/block/node triple determining a point in a lattice where the key occurs. This has the same meaning in both the word and phone indexes
`key`	The index key, which is an n-gram of words (in the word index) or phones (in the phone index). A single word corresponds to a basic keyword index. In the phone index, a phone triple (3-gram) is likely to be the most useful form, but any size n-gram is allowed
`PhoneIndex`	An Index for the phones in the lattice. The index consists of a list of phones or phone "n-grams" (sequences of consecutive phones), together with pointers to where each phone or phone n-gram occurs in the lattices. The design is such that each speaker has a single phone index.
`PhoneIndexEntry`	An entry in the phone index
`defaultLattice`	The default lattice for the lattice entries in the index. This has the same meaning in both the word and phone indexes.
`phoneLexiconRef`	A reference to the phone lexicon used by this speaker. Many speakers are likely to share the same phone lexicon
`wordLexiconRef`	A reference to the word lexicon used by this speaker. Many speakers are likely to share the same word lexicon
`confusionInfoRef`	A reference to the confusion information for the phone lexicon. This attribute is not required, but if used must tally with the phone lexicon, that is, the lexicon provides the labels for the confusion information
`descriptionMetadataRef`	A reference to a `DescriptionMetadataType` for this speaker. It could be the same as that used for other speakers, or may be different
`provenance`	The provenance of this decoding. — `unknown`: The provenance is unknown — `ASR`: The decoding came from an Automatic Speech Recognition system. This is the most likely value. — `manual`: The lattice is manually derived rather than automatic — `keyword`: The lattice consists only of keywords rather than full text. This means that either the ASR was used in word spotting mode (treating the majority of the speech as garbage), or that a manual annotation only chose selected words. Each word should appear as it was spoken in the data, subject only to ASR errors. — `parsing`: The lattice is the result of a higher level parse, perhaps to discern topic or a summary. In this case, a word in the lattice might not correspond directly to words spoken in the data.

6.5.4 SpokenContentIndexEntryType

6.5.4.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SpokenContentIndexEntry datatype                   -->
<!-- ################################################################ -->
<complexType name="SpokenContentIndexEntryType">
  <attribute name="node" type="mpeg7:unsigned16" use="required"/>
  <attribute name="block" type="mpeg7:unsigned16" use="required"/>
  <attribute name="lattice" type="anyURI" use="optional"/>
</complexType>
```

6.5.4.2 Semantics

Name	Definition
SpokenContentIndexEntryType	The format of an entry in the word or phone index. This is a node/block/lattice triple
node	The number of the node at which the index key begins in a particular block
block	The number of the block containing the above node
lattice	The ID of the lattice containing the above node and block. If omitted, the defaultLattice attribute is used from the index

6.5.5 ConfusionCountType

6.5.5.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of ConfusionCount header                              -->
<!-- ################################################################ -->
<complexType name="ConfusionCountType">
  <complexContent>
    <extension base="mpeg7:HeaderType">
      <sequence>
        <element name="Insertion" type="mpeg7:integerVector"/>
        <element name="Deletion" type="mpeg7:integerVector"/>
        <element name="Substitution" type="mpeg7:IntegerMatrixType"/>
      </sequence>
      <attribute name="numOfDimensions" type="positiveInteger"
                 use="required"/>
    </extension>
  </complexContent>
</complexType>
```

6.5.5.2 Semantics

Confusion statistics characterise a particular ASR engine, possibly in the context of a particular speaker, and are calculated using a sequence of speech for which two "decodings" are available:

a) A canonical "decoding" reflecting the actual pronunciations of the words spoken. This is referred to as sequence A.

b) A real decoding from the ASR engine, of the same form as sequence A, but incorporating corruption characterised as insertion, deletion and substitution of phones. This is referred to as sequence B.

A (dynamic programming) string alignment between the two sequences yields the confusion statistics.

If confusion statistics are provided in a description, then a corresponding `PhoneLexicon` must also be provided. The correspondence between indices of the confusion is determined by assigning each phone in the phone lexicon a number based on the order of appearance in the phone lexicon. The first phone appearing in the phone lexicon is assigned an index value of zero, the second phone an index of one, and so on, continuing to assign increasing sequential numbers to phones in the order that they appear in the phone lexicon.

Name	Definition
`numOfDimensions`	The dimensionality of the vectors and matrix in the `ConfusionCountType`. This number must correspond to the size of the `PhoneLexiconType` to which the data applies.
`Insertion`	A vector (of length `numOfDimensions`) of counts, being the number of times each phone was inserted in sequence B.
`Deletion`	A vector (of length `numOfDimensions`) of counts, being the number of times each phone was deleted in sequence B.
`Substitution`	A square matrix (dimension `numOfDimensions`) of counts, being the number of times each phone d in sequence B was substituted in place of each phone p in sequence A. The leading diagonal represents a phone being substituted for itself, i.e., a correct decoding. Each row corresponds to a p and each column to a d, that is, each row represents a canonical phone and each column represents a decoded phone.

6.5.5.3 Usage of ConfusionCountType (informative)

Although the confusion statistics are stored as pure counts, their use is more likely to be as probabilities. There are many different ways to calculate such probabilities using Bayesian or maximum entropy techniques. A simple example is presented which is based upon maximum likelihood.

Represent the counts in the `ConfusionCountType` as follows:

— Substitutions: S_{dp} is the number of times that phone d in sequence B was substituted for phone p in sequence A.

— Insertions: I_d is the number of times that phone d was inserted in sequence B when there was nothing in sequence A at that point.

— Deletions: D_p is the number of times that phone p in sequence A was deleted in sequence B.

The following numbers can easily be calculated:

— N_p is the number of times that phone p occurs in sequence A.

— I is the total number of insertions, that is, the number of times any phone appeared in sequence B where there was nothing in sequence A at that point.

Assume also that sequence A is T^p phones in length, and sequence B is T^d phones in length.

The following probabilities can now be calculated:

The unconditional probability of a phone, d, being inserted: $\frac{I}{T^d}$.

The probability of a phone, d, being inserted, given an insertion took place: $\frac{I_d}{I}$

The probability of phone, p, being inserted: $\frac{D_p}{N_p}$.

The probability of confusing phone p as phone d: $\frac{S_{dp}}{N_p}$.

6.5.6 WordType, PhoneType, WordLexiconIndexType and PhoneLexiconIndexType

6.5.6.1 Syntax

```
<!-- ################################################################ -->
<!-- Definitions of SpokenContent Word and Phone datatypes            -->
<!-- ################################################################ -->
<simpleType name="WordType">
  <restriction base="string"/>
</simpleType>
<simpleType name="PhoneType">
  <restriction base="string"/>
</simpleType>
<simpleType name="WordLexiconIndexType">
  <restriction base="mpeg7:unsigned32"/>
</simpleType>
<simpleType name="PhoneLexiconIndexType">
  <restriction base="mpeg7:unsigned16"/>
</simpleType>
```

6.5.6.2 Semantics

Name	Definition
`WordType`	A type definition defining what a word is. In XML, this is a string. The `WordType` must not contain whitespace characters as this precludes the use of word N-Grams as index keys.
`PhoneType`	As above, but for phones. Again, just a string. The `PhoneType` must not contain whitespace characters as this precludes the use of phone N-Grams as index keys.
`WordLexiconIndexType`	An integral type representing an index into a `WordLexiconType`. The first token in the lexicon is indexed 0, and the second 1 and so on.
`PhoneLexiconIndexType`	An integral type representing an index into a `PhoneLexiconType`. The first token in the lexicon is indexed 0, and the second 1 and so on. Notice that the `PhoneLexiconIndexType` is a 16 bit number, whereas the `WordLexiconIndexType` is a 32 bit number.

6.5.7 LexiconType

6.5.7.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of Lexicon header                                     -->
<!-- ################################################################ -->
<complexType name="LexiconType" abstract="true">
  <complexContent>
    <extension base="mpeg7:HeaderType">
      <attribute name="numOfOriginalEntries" type="positiveInteger"
                 use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

6.5.7.2 Semantics

Name	Definition
LexiconType	An abstract base type representing a lexicon. A lexicon is a list of tokens. The tokens should be added by extension of this type.
numOfOriginalEntries	The original size of the lexicon. In the case of a word lexicon, this should be the number of words originally known to the ASR system, whereas the actual size of the lexicon need only consist of those words decoded in the lattice

6.5.8 WordLexiconType

6.5.8.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of WordLexicon header                                 -->
<!-- ################################################################ -->
<complexType name="WordLexiconType">
  <complexContent>
    <extension base="mpeg7:LexiconType">
      <sequence>
        <!-- The maxOccurs is the upper limit of WordLexiconIndexType  -->
        <element name="Token" type="mpeg7:WordType" minOccurs="1"
                 maxOccurs="4294967295"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.5.8.2 Semantics

Name	Definition
WordLexiconType	A lexicon of words. Each entry represents one orthographic transcription – i.e. a spelling — of a word. Therefore, the lexicon is not a phonetic (pronunciation) dictionary.
Token	An entry in the lexicon

6.5.9 phoneticAlphabetType

6.5.9.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of phoneticAlphabet datatype                          -->
<!-- ################################################################ -->
<simpleType name="phoneticAlphabetType">
  <!-- This defines an [enumerated] type covering the phone sets. It  -->
  <!-- essentially distinguishes between IPA based systems and everything -->
  <!-- else.                                                          -->
  <restriction base="NMTOKEN">
    <enumeration value="sampa"/>
    <enumeration value="ipaSymbol"/>
    <enumeration value="ipaNumber"/>
    <enumeration value="other"/>
  </restriction>
</simpleType>
```

6.5.9.2 Semantics

Name	Definition
phoneticAlphabetType	The name of the phonetic alphabet: — sampa. The speech assessment methods phonetic alphabet. This class also subsumes derivations of SAMPA such as XSAMPA, SAMPROSA and SAMPA-C. The SAMPROSA use of "..." as silence is encouraged. — ipaSymbol. Symbol strings from the international phonetic association alphabet (IPA). This is encoded as unicode — ipaNumber. Numbers from the IPA of the form "XXX", where XXX is the 3 digit IPA index. A phone made up from two or more such numbers should concatenate the numbers thus: XXXYYY... — other. A (possibly proprietary) encoding that does not map onto any of the above

6.5.10 PhoneLexiconType

6.5.10.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of PhoneLexicon header                                -->
<!-- ################################################################ -->
<complexType name="PhoneLexiconType">
  <complexContent>
    <extension base="mpeg7:LexiconType">
      <sequence>
        <!-- The maxOccurs is the upper limit of WordLexiconIndexType  -->
        <element name="Token" type="mpeg7:PhoneType" minOccurs="1"
                 maxOccurs="65536"/>
      </sequence>
      <attribute name="phoneticAlphabet" type="mpeg7:phoneticAlphabetType"
                 default="sampa"/>
    </extension>
  </complexContent>
</complexType>
```

6.5.10.2 Semantics

Name	Definition
PhoneLexiconType	A lexicon of phones
Token	An entry in the lexicon of type PhoneType
phoneticAlphabet	The name of the encoding scheme of the phone lexicon

6.5.11 SpokenContentLatticeType

6.5.11.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of the SpokenContentLattice DS                        -->
<!-- ################################################################ -->
<complexType name="SpokenContentLatticeType">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="Block" minOccurs="1" maxOccurs="65536">
          <complexType>
            <sequence>
              <element name="MediaTime" type="mpeg7:MediaTimeType"/>
              <element name="Node" minOccurs="1" maxOccurs="65536">
                <complexType>
                  <sequence>
                    <element name="WordLink" minOccurs="0" maxOccurs="127">
                      <complexType>
                        <complexContent>
                          <extension base="mpeg7:SpokenContentLinkType">
                            <attribute name="word"
                                       type="mpeg7:WordLexiconIndexType"
                                       use="required"/>
                          </extension>
                        </complexContent>
                      </complexType>
                    </element>
                    <element name="PhoneLink" minOccurs="0" maxOccurs="127">
                      <complexType>
                        <complexContent>
                          <extension base="mpeg7:SpokenContentLinkType">
                            <attribute name="phone"
                                       type="mpeg7:PhoneLexiconIndexType"
                                       use="required"/>
                          </extension>
                        </complexContent>
                      </complexType>
                    </element>
                  </sequence>
                  <attribute name="num" type="mpeg7:unsigned16"
                             use="required"/>
                  <attribute name="timeOffset" type="mpeg7:unsigned16"
                             use="required"/>
                  <attribute name="speakerInfoRef" type="anyURI"
                             use="optional"/>
```

```
            </complexType>
          </element>
        </sequence>
        <attribute name="defaultSpeakerInfoRef" type="anyURI"
                   use="required"/>
        <attribute name="num" type="mpeg7:unsigned16" use="required"/>
        <attribute name="audio" default="speech">
          <simpleType>
            <!-- This gives an approximate measure of how noisy the    -->
            <!-- speech signal is with respect to the speech           -->
            <restriction base="NMTOKEN">
              <enumeration value="unknown"/>
              <enumeration value="speech"/>
              <enumeration value="noise"/>
              <enumeration value="noisySpeech"/>
            </restriction>
          </simpleType>
        </attribute>
      </complexType>
    </element>
  </sequence>
 </extension>
</complexContent>
</complexType>
```

6.5.11.2 Semantics

Name	Definition
SpokenContentLatticeType	The main container for the ASR information. The lattice core is a series of nodes and links. Each node contains timing information and each link contains a word or phone. The nodes are partitioned into blocks to speed access.
Block	A SpokenContentLatticeType consists of blocks, each block consisting of nodes. A block is defined as a lattice with an upper limit on the number of nodes that it can contain. The upper limit is to enable a compact representation for the data types which administrate the block. For instance, restricting the number of nodes in a block to 65536 enables the use of a 16 bit data type for the node number. In addition, the block represents a suitable granularity at which to represent audio quality.
MediaTime	The start time and, optionally, the duration of the block. For a description of the MediaTimeType, see ISO/IEC 15938 part 5.
Node	A node within the block
WordLink	A link between two nodes representing word information
word	The word represented by the link
PhoneLink	As a WordLink, but representing phone information
phone	The phone represented by the link
num	The number of this node. Node numbers range from 0 to 65535
timeOffset	The time offset of this node, measured in one-hundredths of a second, from the beginning of the containing block. The absolute time is obtained by combining the block time with the node offset

Name	Definition
speakerInfoRef	A reference to the SpeakerInfo corresponding to this node.
defaultSpeakerInfoRef	A reference to a SpeakerInfoType describing the default speaker. This reference is used where the speaker entry on a node in this lattice is blank. A typical use would be where there is only one speaker represented in the lattice, in which case it would be wasteful to put the same information on each node. In the extreme case that every node has a speaker reference, the defaultSpeakerRef is not used, but must contain a valid reference. Note that a reference outside the current description placed on every node may lead to a very large description.
num	The number of this block. Block numbers range from 0 to 65535.
audio	A measure of the audio quality pertinent to this block, which facilitates a crude segmentation: — unknown: No information is available. — speech: The signal is known to be clean speech, suggesting a high likelihood of a good transcription. — noise: The signal is known to be non-speech. This may arise when segmentation would have been appropriate but inconvenient. — noisySpeech: The signal is known to be speech, but with facets making recognition difficult. For instance, there could be music in the background.

6.5.12 SpokenContentLinkType

6.5.12.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of SpokenContentLink datatype                         -->
<!-- ################################################################ -->
<complexType name="SpokenContentLinkType">
  <attribute name="probability" type="mpeg7:zeroToOneType" default="1.0"/>
  <attribute name="nodeOffset" type="mpeg7:unsigned16" default="1"/>
</complexType>
```

6.5.12.2 Semantics

Name	Definition
SpokenContentLinkType	The structure of a word or phone link in the lattice
probability	The probability of this link. In a crude sense, this is to indicate which links are more likely than others, with larger numbers indicating higher likelihood.
nodeOffset	The node to which this link leads, specified as a relative offset and defaulting to 1. A node offset leading out of the current block implicitly refers to the next block. A node offset cannot span a whole block, ie., a link from a node in block 3 must lead to a node in block 3 or block 4.

6.5.13 Usage, extraction and examples (informative)

6.5.13.1 Extraction

The Spoken Content set of DSs is designed to be a superset of the output capabilities of most ASR systems on the market at the time of publication. In this sense, the extraction method is highly non-normative; distinct from the Low Level Descriptors in this document.

Commercial large vocabulary speech recognition tends to be aimed at the dictation application area. The ideal output from a user's point of view for this area is a "best pass" – a single hypothesis about what was spoken. This can be stored trivially as a lattice containing a single path. Some commercial recognisers also provide "word alternatives"; these can be represented as a lattice by assuming that each word alternative begins and ends at the same time node. ASR systems used in research organisations tend to be capable of producing lattices as a basic output format.

The structure of the Spoken Content is designed to accommodate some of the errors inherent in ASR. Such errors fall into three different categories:

a) *Basic decoding errors*. State of the art ASR is currently limited by the underlying (usually Hidden Markov Model) technology. Even though the technology improves each year, ASR systems still make basic mistakes in recognition. It is usually difficult to determine the reason for an individual mistake, but it is generally accepted that factors such as background noise, non-canonical pronunciation, strong intonation, dialects and accents all influence recognition performance negatively. The fundamental problem, however, is that ASR systems can not yet utilise all the semantic and pragmatic knowledge that humans use to disambiguate between the many different ways in which a speech signal can be interpreted. A lattice can be used to efficiently represent multiple hypotheses about the correct transcription and their relative likelihoods.

b) *Multiple hypotheses*. Without sufficient contextual knowledge, the recogniser is unable to distinguish some phrases with either the same or similar phonetic content (see, e.g., Figure 21). The phrase "Recognise speech" is often cited as being misrecognised as "wreck a nice beach". To work around this problem, state of the art ASR systems often use a word lattice representation to allow all hypothesised word sequences within the context about which they are confident. The MPEG-7 representation caters for such a lattice.

c) *Unknown words*. A typical state of the art ASR system has a dictionary of perhaps 20,000 to 50,000 words, and cannot produce words outside that vocabulary. Unfortunately, many discriminative (in the sense of information retrieval) words are very uncommon, and will be out of vocabulary (OOV). Typically, names of people and places fit into this category. In general, the ASR system will either omit or replace an OOV word with a phonetically similar within vocabulary word, often corrupting nearby decodings. To prevent OOV words being simply replaced by within vocabulary words, a phonetic representation is available as a lower level semantic representation (see, e.g., Figure 22).

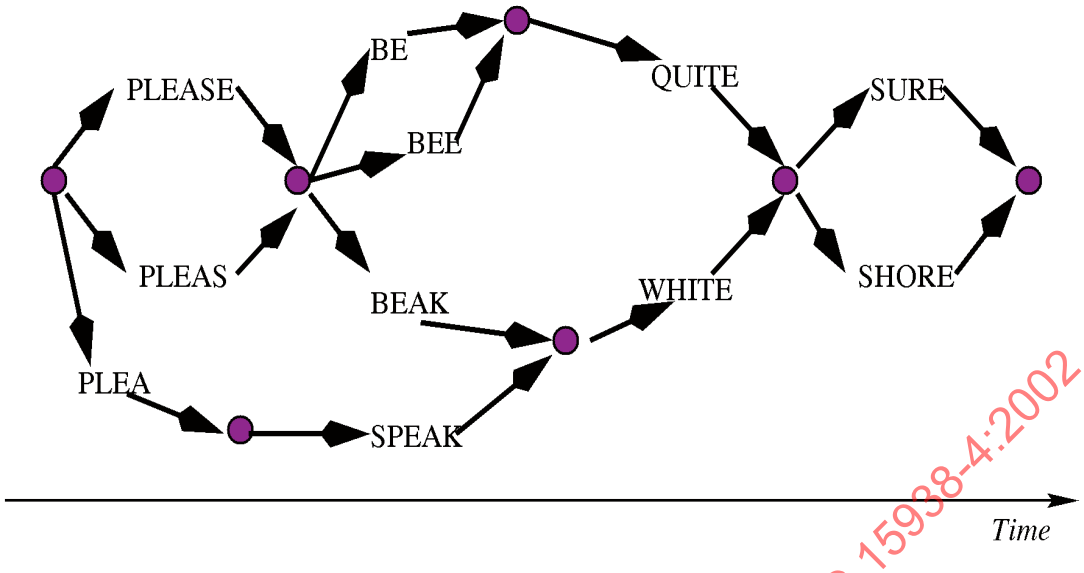

Figure 21 — A hypothetical decoding of the phrase "Please be quite sure". The phrases "be quite" and "beak white" have identical phonetic representations and so may only be distinguished by higher level semantic knowledge

The underlying assumption is that ASR will not provide perfect decodings within the foreseeable future.

The Spoken Content DSs allow representation of various types of valuable information:

a) *Linguistic*: Multiple speakers are supported. In addition, the DS is capable of supporting different languages as well as people mixing languages during dialogue.

b) *Semantic*: The combination of words and phones allows retrieval at different semantic levels. Simple keyword searching over any annotation is unlikely to handle complexities such as anaphoric pronouns or deixis. However, it is possible to perform, e.g., anaphoric resolution on the output of an ASR to produce an annotation stream that may be more readily searched. Although storable in any scheme defined for the ASR output, it is essential that it be marked as coming from a different source. Likewise, were hand-annotated metadata available, this too can be stored in the same scheme as the ASR output. A provenance indicator is provided to distinguish between these data sources.

6.5.13.2 Structure of Spoken Content

The DS structure consists of a number of combined word and phone lattices. A lattice is an acyclic directed graph. It represents different parses of a stream of (audio) data. It is not simply a sequence of words with alternatives. One phone can be substituted for one or more phones, one word can be substituted for one or more words or phones, and that whole structure can form a substitution for one or more words or phones. The lattice structure is designed to handle certain common scenarios:

— Single best path word decoding, e.g., a film annotation derived from the script.

— N-Best word list, e.g., the output of some ASRs

— N-Best utterance list[1)], the output of most ASRs

1) The only difference between an N-Best utterance list and a lattice is the density of the data. The former grows exponentially with the number of hypotheses at any stage, whereas the latter is linear. For long utterances, and especially continuous annotation, the N-Best list is prohibitively large.

— Single best path phone decoding, e.g., a phonetic representation of a word

— Word lattice

— A pure phone lattice, e.g., this may be produced by annotation performed on low CPU devices where full ASR is not possible.

— Combined word and phone lattice to support later retrieval of OOV words.

— Topic labels/predefined taxonomy/or textual summaries of the speech stream, e.g., obtained from parsing methods or textual précis.

The particular level of detail retained in the lattice is up to the annotator. For cases where most words have a high confidence but a few have low confidence (possibly indicating OOV words) we could imagine a relatively thin word lattice with occasional phone additions. For cases of decoding performed on low CPU mobile devices, a purely phone lattice may be appropriate.

Figure 22 shows a simple lattice structure for a single speaker. Each textual entry represents a word or phone, and the small circles represent nodes. Each link between nodes has associated with it a probability, or score, which represents the confidence in that particular decoding (not shown). The lattice is parsed from left to right.

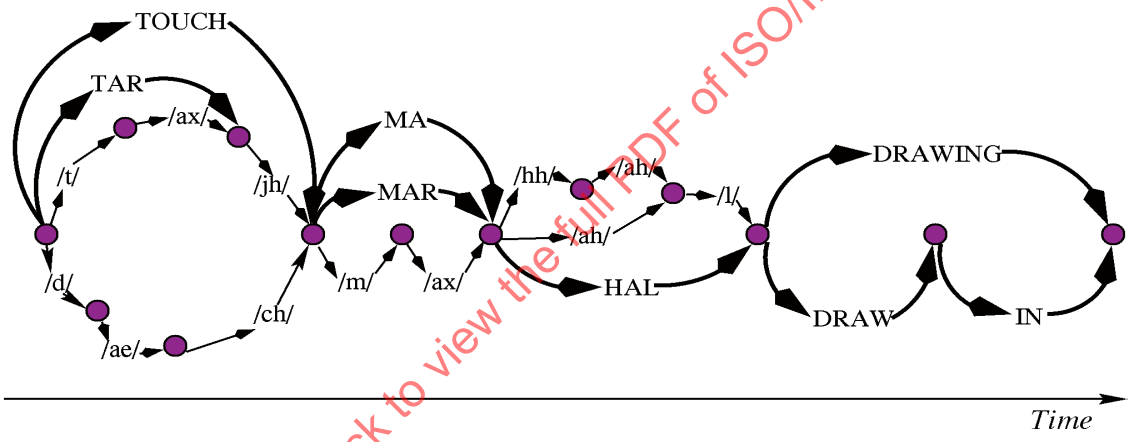

Figure 22 — A lattice structure for an hypothetical (combined phone and word) decoding of the expression "Taj Mahal drawing …". It is assumed that the name 'Taj Mahal' is out of the vocabulary of the ASR system

6.5.13.3 Example

```
<Header xsi:type="SpokenContentHeaderType">
  <!-- This is the (first) lexicon                                    -->
  <WordLexicon id="wlZero" numOfOriginalEntries="1000">
    <Token>draw</Token>
    <Token>drawing</Token>
    <Token>hal</Token>
    <Token>in</Token>
    <Token>ma</Token>
    <Token>mar</Token>
    <Token>tar</Token>
    <Token>touch</Token>
  </WordLexicon>
  <!-- This is an abbreviated phone set. A phone set would normally   -->
  <!-- contain more entries                                           -->
  <PhoneLexicon id="plZero" numOfOriginalEntries="10"
                phoneticAlphabet="other">
    <Token>ae</Token>
    <Token>ah</Token>
```

```
    <Token>ax</Token>
    <Token>ch</Token>
    <Token>d</Token>
    <Token>hh</Token>
    <Token>jh</Token>
    <Token>l</Token>
    <Token>m</Token>
    <Token>t</Token>
  </PhoneLexicon>
  <ConfusionInfo id="ciZero" numOfDimensions="10">
    <Insertion> 10 10 10 2 50 5 5 105 10 10 </Insertion>
    <Deletion> 1 9 21 2 9 2 1 5 1 10 </Deletion>
    <Substitution dim="10 10">
      10 10 10 10 10 10 10 10 10 10
      10 10 10 10 10 10 10 10 10 10
      10 10 10 10 10 10 10 10 10 10
      10 10 10 10 10 10 10 10 10 10
      10 10 10 10 10 10 10 10 10 10
      10 10 10 10 10 10 10 10 10 10
      10 10 10 10 10 10 10 10 10 10
      10 10 10 10 10 10 10 10 10 10
      10 10 10 10 10 10 10 10 10 10
      10 10 10 10 10 10 10 10 10 10
    </Substitution>
  </ConfusionInfo>
  <DescriptionMetadata id="infoZero">
    <Instrument>
      <Tool>
        <FreeTerm>Some speech recognition engine</FreeTerm>
      </Tool>
      <Setting name="matchFactor" value="0.5"/>
    </Instrument>
  </DescriptionMetadata>
  <!-- Information about the person who is speaking                    -->
  <SpeakerInfo id="sZero" phoneLexiconRef="#plZero" confusionInfoRef="#ciZero"
               wordLexiconRef="#wlZero" descriptionMetadataRef="#infoZero"
               provenance="ASR">
    <SpokenLanguage>en-UK</SpokenLanguage>
    <Person>
      <Name>
        <GivenName initial="J">Jason</GivenName>
        <GivenName initial="P">Peter</GivenName>
        <FamilyName>Charlesworth</FamilyName>
        <Title>Dr</Title>
      </Name>
    </Person>
    <!-- Word index. For example, the lattice is explicit               -->
    <WordIndex defaultLattice="#lZero">
      <WordIndexEntry key="1">
        <IndexEntry node="0" block="1" lattice="#lZero"/>
      </WordIndexEntry>
    </WordIndex>
    <!-- Phone index. For example, the lattice is implicit              -->
    <PhoneIndex defaultLattice="#lZero">
      <PhoneIndexEntry key="4 0 3">
        <IndexEntry node="0" block="0"/>
      </PhoneIndexEntry>
      <PhoneIndexEntry key="2 6 8">
        <IndexEntry node="2" block="0"/>
      </PhoneIndexEntry>
    </PhoneIndex>
```

```
  </SpeakerInfo>
</Header>
<AudioDescriptionScheme xsi:type="SpokenContentLatticeType" id="lZero">
  <!-- StartTime="15:02:35 21-02-2000"                                    -->
  <Block num="0" audio="noisySpeech" defaultSpeakerInfoRef="#sZero">
    <MediaTime>
      <MediaTimePoint>2000-02-21T15:02:35</MediaTimePoint>
    </MediaTime>
    <Node num="0" timeOffset="0">
      <WordLink nodeOffset="5" probability="0.5" word="7"/>
      <WordLink nodeOffset="4" probability="0.5" word="6"/>
      <PhoneLink nodeOffset="2" probability="0.45" phone="9"/>
      <PhoneLink probability="0.45" phone="4"/>
    </Node>
    <Node num="1" timeOffset="21">
      <PhoneLink nodeOffset="2" probability="0.45" phone="0"/>
    </Node>
    <Node num="2" timeOffset="25">
      <PhoneLink nodeOffset="2" probability="0.45" phone="2"/>
    </Node>
    <Node num="3" timeOffset="32">
      <PhoneLink nodeOffset="2" probability="0.45" phone="3"/>
    </Node>
    <Node num="4" timeOffset="40">
      <PhoneLink probability="0.45" phone="6"/>
    </Node>
    <Node num="5" timeOffset="50">
      <WordLink nodeOffset="2" probability="0.5" word="4"/>
      <WordLink nodeOffset="2" probability="0.5" word="5"/>
      <PhoneLink probability="0.45" phone="8"/>
    </Node>
    <Node num="6" timeOffset="60">
      <PhoneLink probability="0.45" phone="2"/>
    </Node>
    <Node num="7" timeOffset="70">
      <WordLink nodeOffset="3" probability="0.5" word="2"/>
      <PhoneLink probability="0.45" phone="5"/>
      <PhoneLink nodeOffset="2" probability="0.45" phone="1"/>
    </Node>
    <Node num="8" timeOffset="80">
      <PhoneLink probability="0.45" phone="1"/>
    </Node>
    <Node num="9" timeOffset="90">
      <PhoneLink probability="0.45" phone="7"/>
    </Node>
  </Block>
  <!-- StartTime="15:02:36 21-02-2000"                                    -->
  <Block num="1" audio="speech" defaultSpeakerInfoRef="#sZero">
    <MediaTime>
      <MediaTimePoint>2000-02-21T15:02:36</MediaTimePoint>
    </MediaTime>
    <Node num="0" timeOffset="0" speakerInfoRef="#sZero">
      <WordLink nodeOffset="2" probability="0.5" word="1"/>
      <WordLink probability="0.5" word="0"/>
    </Node>
    <Node num="1" timeOffset="20" speakerInfoRef="#sZero">
      <WordLink probability="0.5" word="3"/>
    </Node>
    <Node num="2" timeOffset="40" speakerInfoRef="#sZero"/>
  </Block>
</AudioDescriptionScheme>
```

6.6 Melody

6.6.1 Introduction

The `Melody` DS is a rich representation for monophonic melodic information to facilitate efficient, robust, and expressive melodic similarity matching. The `Melody` DS includes tools for extremely terse, efficient melody contour representation, and tools for a more verbose, complete, expressive melody representation. Both tools support matching between melodies, and can support optional supporting information about the melody that may further aid search.

6.6.2 MelodyType

6.6.2.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of Melody DS                                          -->
<!-- ################################################################ -->
<complexType name="MelodyType">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="Meter" type="mpeg7:MeterType" minOccurs="0"/>
        <element name="Scale" type="mpeg7:scaleType" minOccurs="0"/>
        <element name="Key" type="mpeg7:KeyType" minOccurs="0"/>
        <choice>
          <element name="MelodyContour" type="mpeg7:MelodyContourType"/>
          <element name="MelodySequence" type="mpeg7:MelodySequenceType"
                   minOccurs="1" maxOccurs="unbounded"/>
        </choice>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.6.2.2 Semantics

Name	Definition
`MelodyType`	A structure containing optional elements that support the description of melody, and either a compact description of melody contour or a verbose description of the melody.
`Meter`	The time signature(s) of the melody of `MeterType`.
`Scale`	An array of intervals representing the (chromatic) scale steps to the point at which the scale repeats
`Key`	A container type containing degree, alteration, and mode.
`MelodyContourType`	A structure containing a compact representation of the melody of the referenced segment (inherited from `AudioSegment`) with interval contour, meter, and beat information.
`MelodySequenceType`	A structure containing a verbose representation of the melody of the referenced segment with precise interval and timing information, and with optional starting note and lyric information.

6.6.3 Meter

The Meter defines the time signature(s) of an audio segment, which is necessary for determining rhythmic information (strong vs. weak beats) of `MelodyType` data. A time signature is indicated by two values. The numerator defines the number of beats per measure. The denominator defines the chosen unit of measurement of beats (whole note=1, half-note=2, quarter-note=4, etc.). Examples of common time signatures are 4/4, 2/4, and 3/4. A few examples of less common time signatures are 11/8 and 19/16.

6.6.3.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of Meter D                                            -->
<!-- ################################################################ -->
<complexType name="MeterType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="Numerator">
          <simpleType>
            <restriction base="integer">
              <minInclusive value="1"/>
              <maxInclusive value="128"/>
            </restriction>
          </simpleType>
        </element>
        <element name="Denominator">
          <simpleType>
            <restriction base="integer">
              <enumeration value="1"/>
              <enumeration value="2"/>
              <enumeration value="4"/>
              <enumeration value="8"/>
              <enumeration value="16"/>
              <enumeration value="32"/>
              <enumeration value="64"/>
              <enumeration value="128"/>
            </restriction>
          </simpleType>
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.6.3.2 Semantics

Name	Definition
`MeterType`	The time signature(s) of the melody.
`Numerator`	Includes integer values (from 1 to 128) for the numerator of the time signature.
`Denominator`	Includes integer powers of two (1, 2, 4, …, 128) for the denominator of the time signature.