INTERNATIONAL STANDARD

ISO/IEC 15938-4

First edition
2002-06-15

AMENDMENT 2
2006-10-01

Information technology — Multimedia content description interface —

Part 4: **Audio**

AMENDMENT 2: High-level descriptors

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

Partie 4: Audio

AMENDEMENT 2: Descripteurs de haut niveau

Reference number
ISO/IEC 15938-4:2002/Amd.2:2006(E)

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 2.

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.

Amendment 2 to ISO/IEC 15938-4:2002 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Information technology — Multimedia content description interface —

Part 4:
Audio

AMENDMENT 2: High-level descriptors

Remove subclauses 5.2.3. and add following subclauses:

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"/>
        <element name="LogBase" type ="float" default="1.0" 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.
`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.
`LogBase`	In the case, its value is different to the default value, a logarithm has to be performed on the input data, before calculating any series (mean, variance…).The value is the base of a logarithm that is performed on the input data. Note that the value of LogBase must be greater than 0.

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.

In the case, that the value of LogBase is different from its default value, a logarithm must be performed on any input data before series calculation. In case, it is equal to the default value, no logarithm must be performed on the input data. The following formula shows the rule for this calculation. *Base* contains the base of the logarithm and is defined in LogBase. In case the logarithmic calculation is invalid (for e.g. log 0) or the calculated output is smaller than -1.0e^2 the output value is fixed to -1.0e^2.

$$outputValue = \log_{base}(inputValue)$$

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 AMD2.1 illustrates the scalability property. This scaled series can be derived indifferently from the original series by applying the scaling operation with the `ratio`s shown, or from the scaled Series of Figure AMD2.1 by applying the appropriate rescaling operation. The result is identical. Scaling operations are chosen among those for which this property can be enforced.

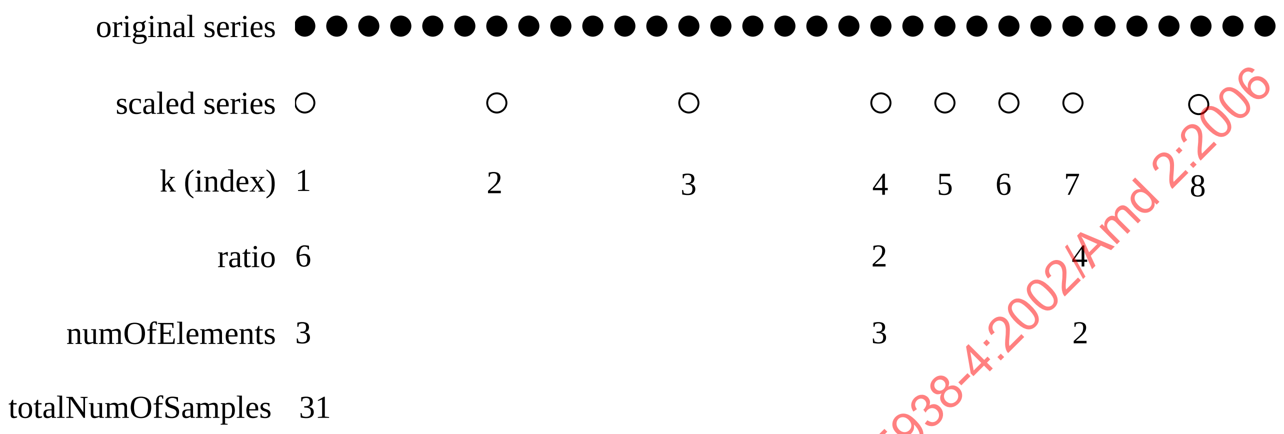


Figure AMD2.1 — An illustration of the scalability property

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. In case *logBase* is not equal to the default value, X is the logarithm of the input data, otherwise the raw input data. 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.

Replace subclause 5.2.5 with the following:

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="LogBase" type ="float" default="1.0" 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.

Name	Definition
`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.
`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.
`LogBase`	Base of a logarithm that is performed on the input data. If the value is equal the default value, no logarithm is performed. Note that the value of LogBase must be greater than 0.
`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`	$MSD_k = \max_{i=1+(k-1)N}^{kN}\left\lVert x_i - \bar{x}_k \right\rVert^2$	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.

In the case, that the value of *LogBase* is different from the default value, a logarithm must be performed on any input data before series calculation. In case, it is equal to the default value, no logarithm must be performed on the input data. The following formula shows the rule for this calculation. *Base* contains the base of the logarithm and is defined in *LogBase*. In case the logarithmic calculation is invalid (for e.g. log 0) or the calculated output is smaller than -1.0e^2 the output value is fixed to -1.0e^2.

$$outputValue = \log_{base}(inputValue)$$

In case *logBase* is equal to the default value, $\bar{x}_i^j$ is the mean of N samples and X are the raw input data.

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

Add at the end of subclause 6.8.3.3.3:

6.9 Rhythmic Pattern

6.9.1 RhythmicBaseType

This base descriptor contains a description of one single rhythmic pattern. The pattern is represented in a way that the parts of the bar are sorted in order of their importance. This is based on the fact, that the importance decreases with the order of the prime index. Regarding any further classification or matching of rhythmic patterns, this representation allows a setting of several grades in the resolution. Therefore for every pattern the most compact representation can be provided, resulting in an efficient comparison of the patterns and minimal memory needed for the storage.

6.9.1.1 Syntax

```
<!-- ############################################################ -->
<!-- Definition of RhythmicBaseD  -->
<!-- ############################################################ -->
<complexType name="RhythmicBaseType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="PrimeIndex" type="mpeg7:integerVector" minOccurs="1"
                 maxOccurs="1"/>
        <element name="Velocity" type="mpeg7:integerVector" minOccurs="1"
                 maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.9.1.2 Semantics

Name	Definition
`RhythmicBaseType`	The RhythmicBaseType contains elements of a single rhythmic pattern with different degrees of resolution.
`PrimeIndex`	The Integer vector indicating the initial index of the rhythmic pattern.
`Velocity`	The Integer vector indicating the velocity of the elements.

6.9.1.3 Usage

The RhythmicBaseType contains elements of a single rhythmic pattern with different degrees of resolution (i.e. on any different hierarchic levels). A representation of a rhythmic pattern requires indexing of the rhythmic grid with respect to the rhythmic significance of the grid position.

The calculation of each *PrimeIndex* of the example pattern may be done in the following manner:

1. Vector of prime factorization of the top part of the meter:
   `nomVec = { nom`$_1$`… nom`$_k$`}` sorted by size with the largest value first

2. Vector of prime factorization of the number of divisions per beat (tick):
   `mtVec = { mt`$_1$`… mt`$_k$`}`

3. calculation of the prime indices from the grid positions:

```
patternLength = product(nomVec) * product(mtVec)
                     ( product: multiply each value within the vector)
vector primeVec()  ( initialized with the size of patternLength )
primeVec()= 0      ( set all Elements of the vector to 0 )
primeIndex = 1
primeProduct = 1
count = 0

for i=1 : length(nomVec)
{
  primeProduct *= nomVec(i)
  while (count < patternLength)
  {
    if (((count/(patternLength/primeProduct)) modulo 1 ) == 0)
    {
      if (primeVec[count]==0)
      {
        primeVec[count] = primeIndex;
        primeIndex++;
      }
    }
    count++;
  }
}
for i=1 : length(mtVec)
{
  primeProduct *= mtVec (i)
  while (count < patternLength)
  {
    if (((count/(patternLength/primeProduct)) modulo 1 ) == 0)
    {
      if (primeVec[count]==0)
      {
        primeVec[count] = primeIndex;
        primeIndex++;
      }
    }
    count++;
  }
}
```

The successive prime factorization of the nominator and the micro time is necessary, because a joint prime factorization of the maximum number of elements can lead to comparisons of patterns with different time signature (but same length) in cases of reduced rhythmic resolution.

Example 1: meter = 4/4; micro time = 2; resulting size = 4 * 2 = 8 ;

The following table shows a rhythmic pattern with a binary feeling notated as commonly done in a score-like representation:

part of the bar	1	1+	2	2+	3	3+	4	4+
grid position	1	2	3	4	5	6	7	8
prime index	1	5	3	6	2	7	4	8
velocity	100	0	112	0	150	68	120	0

No elements will be applied for any part of the bar with velocity equal to zero:

prime index	1	3	2	7	4
velocity	100	112	150	68	120

According to the ascending order of the prime indexes the elements will be rearranged, resulting in the final representation:

prime index	1	2	3	4	7
velocity	100	150	112	120	68

Example 2: meter = 4/4; micro time = 3; resulting size = 4 * 3 = 12 ;

The following table shows a rhythmic pattern with a ternary feeling notated as commonly done in a score-like representation:

part of the bar	**1**	**1**+	**1**++	**2**	**2**+	**2**++	**3**	**3**+	**3**++	**4**	**4**+	**4**++
grid position	1	2	3	4	5	6	7	8	9	10	11	12
prime index	1	5	6	3	7	8	2	9	10	4	11	12
velocity	180	0	100	200	0	99	190	0	97	205	0	101

No elements will be applied for any part of the bar with velocity equal to zero.

prime index	1	6	3	8	2	10	4	12
velocity	180	100	200	99	190	97	205	101

According to the ascending order of the prime indexes the elements will be rearranged, resulting in the final representation:

grid position	1	2	3	4	6	8	10	12
velocity	180	190	200	205	100	99	97	101

The following example demonstrates how two representations exhibiting different grades of resolution can be easily compared to each other. Only the number of elements of the shorter representation is taken into account. It is advantageous to compare only patterns with similar meter.

Pattern 1: meter: 4/4;

grid position	1	2	3	4	7
velocity	100	150	112	120	68

Pattern2: meter: 4/4;

prime index	1	2
velocity	120	145

The examples demonstrate how the use of a specific order allows the specification of a rhythmic hierarchy without any additional information.

6.9.2 AudioRhythmicPatternDS

The AudioRhythmicPatternType provides a more comprehensive description of a rhythmical structure of a whole song. The internal structure of the representation is dependent on the underlying rhythmical structure of the pattern that has been defined in RhythmicPatternType. Additionally to the rhythmic information of the pattern, this descriptor contains meter, instrument information, number of recurrences and segments as well.

6.9.2.1 Syntax of AudioRhythmicPatternType

```
<complexType name="AudioRhythmicPatternDS">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="SinglePattern">
          <complexType>
            <sequence>
              <element name="Instrument" type="mpeg7:CreationToolType"/>
              <element name="Recurrences">
                <simpleType>
                  <restriction base="integer">
                    <minInclusive value="0"/>
                  </restriction>
                </simpleType>
              </element>
              <element name="RhythmPattern" type="mpeg7:RhythmicBaseType"/>
              <element name="Meter" type="mpeg7:MeterType"/>
              <element name="AudioSegment" type="mpeg7:AudioSegmentType"/>
            </sequence>
          </complexType>
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.9.2.2 Semantics

Name	Definition
`AudioRhythmicPatternDS`	A description scheme providing a compact and efficient representation of rhythmical patterns.
`SinglePattern`	Element that describes one single rhythmic pattern.
`Instrument`	Describes the devices/procedure and settings used for the creation of the metadata, such as the tools used to extract the metadata or the extraction parameters. `Instrument` is of type `CreationToolType`. Musical instruments should be only drum instruments and they must have been previously defined.
`Recurrences`	The number of recurrences of the same pattern.
`RhythmPattern`	The rhythmical pattern of an audio signal of *PatternType* data.
`Meter`	The meter of the music pattern.
`AudioSegment`	The time when the pattern starts and the duration of it.

6.9.2.3 Instantiation requirements

In order to guarantee a proper instantiation of this description scheme, the following requirements have to be fulfilled:

- The number of elements of *Velocity* and *PrimeIndex*, as provided by *RhythmicBaseType* must be equal and not 0.

6.9.2.4 Usage

The *AudioRhythmicPattern* DS aggregates rhythmic information from different fields of the song. Single patterns have been specified in *RhythmicBaseType*. To define a drum instrument that plays the actual pattern the attribute *Instrument* has been introduced. When using the *CreationToolType* for specifying an instrument it must be assured, that only drum instruments have been used. *Recurrences* describes the number of times the same pattern is played consecutively. To describe the currently played pattern *RhythmPattern* must be used. The start of the rhythmic pattern and its length (including the number of recurrences) must be indicated by *AudioSegment*.

The first step of extracting rhythmic patterns from a polyphonic music signal would be to transcribe percussive instruments. A commonly used technique is template matching by performing a differentiation, a halfway-rectification and a Principal Component Analysis on the input data to find spectral characteristics of un-pitched percussive instruments and to transcribe the actual drum tracks.

After transcribing the input data and obtaining the drum tracks, the actual pattern could be calculated. At first, the audio signal might be segmented into similar and characteristic regions using a self-similarity method. The segmentation is motivated by the assumption, that within each region not more than one representative drum pattern occurs, and that the rhythmic features are nearly invariant. Subsequently, the temporal positions of the events are quantized on a tatum grid. The term tatum grid refers to the pulse series on the lowest metric level. Tatum period and phase is computed by means of a two-way mismatch error procedure. The pattern length might be estimated by searching for the prominent periodicity in the quantized score with periods equal to an integer multiple of the bar length. The periodicity function is obtained by calculating a similarity measure between the signal and its time shifted version. The similarity between two score representations is calculated as weighted sum of the number of simultaneously occurring notes and rests in the score. The pattern is calculated by means of a histogram representation measuring the occurrence of notes on each metrical position within the pattern for each instrument. By comparing the histogram values with an arbitrary threshold the pattern elements are chosen as frequently occurring notes.

6.9.2.5 Applications

6.9.2.5.1 Automatic Retrieval and Recommendation

When using rhythmic patterns to refer to a particular musical style or genre it is possible to query for musical content that is also characterized by one ore more representative rhythmic patterns. This mechanism can serve as a search criterion in applications proposing a number of musical titles belonging to a particular musical style or genre or sounding similar to a title being proposed by the user as a “reference sound”.

6.10 Chord Pattern

6.10.1 ChordBaseType

This descriptor contains a description of a single musical chord. A chord is a musical structure formed when at least two tones having different notes are played simultaneously. If the default ISO/IEC 15938-4 scale descriptor is used, numbers 0 to 11 indicate the half tones in the scale. The representation of the *ChordBaseType* element consists of a *RelativeChordNumber*, which represents the base note of a chord relative to the active key, the type of triad, and additional tones. For example, a *RelativeChordNumber* of 4, in the Key of C, represents a base note of E because it is four half tones higher than the base key.

6.10.1.1 Syntax

```
<!-- ############################################################ -->
<!-- Definition of ChordBaseD  -->
<!-- ############################################################ -->
<complexType name="ChordBaseType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="NoteAdded" minOccurs="0">
          <simpleType>
            <restriction base="integer">
              <minInclusive value="0"/>
              <maxInclusive value="11"/>
            </restriction>
          </simpleType>
        </element>
        <element name="NoteRemoved" minOccurs="0">
          <simpleType>
            <restriction base="integer">
              <minInclusive value="0"/>
              <maxInclusive value="11"/>
            </restriction>
          </simpleType>
        </element>
      </sequence>
      <attribute name="RelativeChordNumber" type="positiveInteger"
                 use="required"/>
      <attribute name="Triad" use="optional" default="major">
        <simpleType>
          <restriction base="string">
            <enumeration value="major"/>
            <enumeration value="minor"/>
            <enumeration value="augmented"/>
            <enumeration value="diminished"/>
            <enumeration value="sus2"/>
            <enumeration value="sus4"/>
          </restriction>
        </simpleType>
      </attribute>
      <attribute name="SeventhChord" use="optional" default="none">
        <simpleType>
          <restriction base="string">
            <enumeration value="none"/>
            <enumeration value="major7"/>
            <enumeration value="dominant7"/>
          </restriction>
        </simpleType>
      </attribute>
      <attribute name="AlternateBass" type="positiveInteger" use="optional"/>
    </extension>
  </complexContent>
</complexType>
```

6.10.1.2 Semantics

Name	Definition
`ChordBaseType`	This type contains a description of the actual chord.
`RelativeChordNumber`	This element is an integer that describes the base note of a currently played chord. The value of *RelativeChordNumber* indicates the numbers of half tones in the scale from the root note of the active Key and it must not be greater than the number of elements of the scale that is used.
`Triad`	This string contains an enumeration of all possible types of the triad.
`SeventhChord`	This enumeration extends the already existing chord to a septimal harmony.
`NoteAdded`	*NoteAdded* is an integer that describes a note that additionally appears within the chord. Its value is the number of half tones above the base note from the currently played chord. *Note that this is relative to RelativeChordNumber*. For Example a "C major" chord is extended by a "D", *NoteAdded* would be 2. Its value must not be greater than twice the number of items in the scale.
`NoteRemoved`	In the case the base chords contains notes that are not desired, they may be removed with *NoteRemoved*. The value of NoteRemoved is the number of half tones above the base note of the currently played chord. *Note that this is relative to RelativeChordNumber*. For Example to remove the "E" from a "C major" chord *NoteRemoved* would be 4. Its value must not be greater than the number of items in the scale.
`AlternateBass`	Some chords might have an alternate bass note This attribute indicates the base note to be played with the chord. Its value is the number of half tones higher than the base note of the currently played chord. For Example to add an alternate "G" to a "C major" chord *AlternateBass* would be 7. Its value must not be greater than the number of items in the scale.

6.10.1.3 Usage and Extraction

The descriptor ChordBaseType contains the description of one single chord. Its representation consists of a base note, the type of the triad and possible additional notes.

6.10.1.3.1 Representation

Musically, a tone interval can be described as a simultaneous appearance of at least two individual notes with different tones. At least three different tones would build a chord. In musical practice it is very common to describe a chord as a triad writing the base note and type of triad. Additionally one can add special information for notes, which are different from the triad, such as 7th chords. In case of the *Chord Pattern*, depending on the scale, by default only the common western music style with enharmonic change is taken into consideration. The examples of the following text refer to the standard music scale as described in ISO/IEC 15938-4:2002, *ScaleType* with enharmonic change and a range 12 half tones. Furthermore the *RelativeChordNumber* is one and refers to note "C". The following example image (Figure AMD2.2) shows the scale and referring key numbers on a usual piano keyboard.

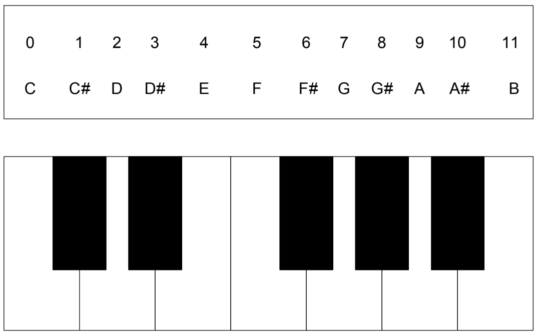


Figure AMD2.2 – Example piano keyboard and above example key and referring notes

To describe a musical chord using the *ChordBase* descriptor, one base note and five different extensions have been designed to cover all note possibilities. *RelativeChordNumber* contains the number referring to a base note of the triad. It represents the base note of a chord relative to the active key, the type of triad, and additional tones. For example, a RelativeChordNumber of 4, in the Key of C, represents a base note of E. In this example *RelativeChordNumber* is 0 and refers to note "C". The attribute *Triad* declares the mode of the triad. The default mode is a major chord. Annex D shows a list containing 13 rows. The first row refers to the mode. The next 12 rows state the occurrence of notes in this chord by a "1" in the row, "0" otherwise. For example: The C major chord contains the notes "C", "E", and "G". Annex D.2 contains at row "major" a number only in columns 0, 4 and 7, meaning the chord consists of the base note (0→ "C"), the major third (4 half tones above the base note →"E") and the fifth (7 half tones above the base note). The same procedure would be fulfilled when creating the other triads. *SeventhChord* describes an appearance of a septimal harmony, where a tone 10 or 11 half tones higher than the base tone. It can be distinguished between a 7th major (10 half tones higher than the base tone) and a 7th dominant (11 half tones higher than the base tone) chord. The default value is "none", meaning, there is no 7th chord existing. Annex D.1 shows the additional note that needs to occur to fulfil the requirements of this chord. For Example, when the C major triad contains an additional dominant seventh, a "B*b*" is added. The chord would change to Cmajor dom7, or commonly expressed C dom7. In this case the *BaseNote* is "C", *Triad* is "major" and *SeventhChord* is "dominant7".

There might be special tones that occur additionally, that are relevant only to some music information retrieval applications. In this case the *NoteAdded* can be used to add more tones to the chord. The value specified is the number of half tones above the base note of the chord. Therefore, if somebody wants to add a "D" to a "C major" chord, its value would be 2. Note that *NoteAdded* must not be used to indicate tones that are already indicated by the Triad or *SeventhChord* descriptors. The element *NoteRemoved* is embedded in a sequence as well. With this it is possible to delete tones from the chord. Its values are specified as the number of half tones above the base tone. If somebody wants to discard the major third (for example "E" at the "C major" triad) from the chord, the value of this descriptor would be 4. Note that *NoteRemoved* must not be used to indicate tones that are not indicated by Triad or *SeventhChord*. *NoteRemoved* must also not be used to indicate tones that are added using *NoteAdded*.

Possible way of Chord Extraction from a polyphonic audio signal

The tonal components might be extracted from the frequency domain representation of the signal under investigation. The separation of those tonal components from noise and percussive occurrences is crucial to the effectiveness of the consecutive process. From the result of the previous stages, the notes are determined by eliminating harmonic overtones that stem from the characteristic of the instrument playing the note under consideration. The individual notes are then mapped to chords in order to determine the underlying chord pattern and to find the dominant chords for a segment of music.

Extraction of Prominent Tonal Components

It is recommended using transformations with a variable frequency resolution providing higher density of frequency bins in the low frequency range and a lower density at high frequencies, respectively. The extraction method that has been implemented and tested is a derivative of the Warped FFT.

Separating Tonal Components from Transients and Erasing Overtones

The tonal components of interest usually possess a longer temporal duration than the transient components and a focussed occurrence in the frequency range. Thus a straightforward approach is to perform the separation based on these properties.

As a first step, all local maxima in the frequency range are identified that lie above a certain amplitude threshold. Secondly, a minimum length criterion in temporal direction is applied to the candidates found in step one. It must be stated, that in some cases, tonal components are erased also, e.g. if they are extremely short.

To perform a reliable harmonic analysis from a frequency domain representation of a musical signal, it is critical to identify the overtones and to use only the fundamentals for the further processing.

Determination of Chords

The chord analysis is based on a pattern matching method. The tonal events are compared with reference vectors representing different chord types. They consist of ones where tones exist in the related chord type and zeros elsewhere, e.g. the vector for major chords has a one in the first, fifth and eighth cell.

A matrix of twelve rows representing the tones of the western scale is built from the input. For every time frame the amplitudes of the tonal events are written into a column of this matrix. If a tone appears in more than one octave within one frame, the amplitudes for the corresponding row and column are summed up giving this tone a higher weight for calculation. Every column of the matrix is then successively compared with all reference vectors by calculating the scalar product. The input columns are shifted, so that all nonzero elements are in the first cell once for processing the comparison. This way all possible chord inversions are checked. The chord type belonging to the reference vector that yields the highest scalar product is chosen and the respective key-note is deduced from the number of shifting operations.

The presented method has the significant advantage of being very easily upgradeable to other chord types by simply adding their reference vectors to the algorithm.

6.10.2 AudioChordPatternDS

The AudioChordPattern DS is a description scheme which allows a compact representation of chord patterns. The internal structure of the representation is dependent on the proper chord structure of the pattern that has to be represented. Chords have been defined relative to the key in which they are played. In this way a comparison is simplified, because the chord progression itself is independent of the key.

6.10.2.1 Syntax

```
<!-- ################################################################## -->
<!-- Definition of AudioChordPatternDS                                  -->
<!-- ################################################################## -->
<complexType name="AudioChordPatternDS">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="Key" type="mpeg7:KeyType"/>
      <sequence>
        <element name="Chord" maxOccurs="unbounded">
          <complexType>
            <sequence>
              <element name="ChordBase" type="mpeg7:ChordBaseType" minOccurs="1"
                       maxOccurs="1"/>
              <element name="AudioSegment" type="mpeg7:AudioSegmentType"
                       minOccurs="1" maxOccurs="1"/>
            </sequence>
          </complexType>
        </element>
      </sequence>
    </sequence>
  </extension>
</complexContent>
</complexType>
```

6.10.2.2 Semantics

Name	Definition
`AudioChordPatternDS`	A representation of a chord pattern of an audio signal.
`Key`	A container type containing degree, alteration, and mode of a popular western music note.
`Chord`	Chord is the upper element containing a chord and the according audiosegment.
`ChordBase`	The actual chord element, defined in ChordBaseType
`AudioSegment`	The time when the pattern starts and the duration of it. The TimePoint must be set relative to the beginning of the song.

6.10.2.3 Instantiation requirements

In order to guarantee a proper instantiation of this description scheme, the following requirements have to be fulfilled:

- The number of elements of *Chord*, as provided by *ChordBaseType* must be greater than 0.
- The number of elements of the scale used in Key must be the same scale as in ChordType
- The value of the first sequence in *ChordType::RelativeChordNumber* must be “1”.
- The element *Key* must implement only one element from *KeyNote*.

6.10.2.4 Usage and Extraction

6.10.2.4.1 Representation of Chord Pattern

In common western popular music often chord progression patterns that repeat but change slightly can be found, for example the chords C major, F major, G major and again C major. Musically this issue is called cadence. But the same cadence can be reached with the chords “D major”, “G major” and “A major” because it consists of keynote, dominant, subdominant and again the keynote and is independent of its actual key. To compare two different chord patterns it is therefore mostly not relevant whether the base chord is “C major” or “D major”. It is more important that the successive chords have the same distance from the first chord. Therefore *AudioChordPatternDS* has been designed not to state the absolute key number in *ChordType* but its relative distance to the first occurred chord. That means if the chord progression starts with “D major”, the *RelativeChordNumber* in *ChordType* must be 0. In this case Key would be “D”. If the following chord is “A major”, *RelativeChordNumber* in *ChordType* must be 7, because in this scale “A” is in the default scale 7 half tones above the “D”. Therefore it is easy to compare chord progression patterns independent of their keynote. The following image (Figure AMD2.3) shows the issue in comparison to a musical chord progression pattern with a C major cadence in example.

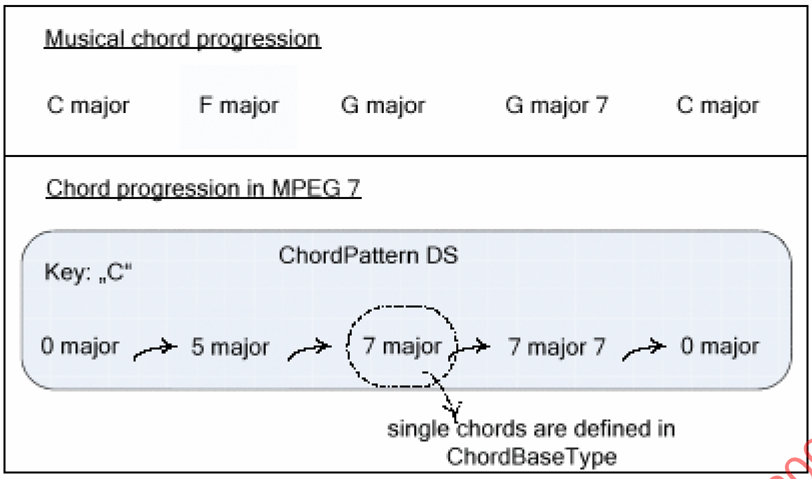


Figure AMD2.3 – Chord progression in MPEG 7 in comparison to a musical chord progression

6.10.2.4.2 Chord Pattern example

The following example shows an XML schema using the Audio*ChordPattern* description scheme with the chords "C major", F major, "G major" and then again "C major" (cadence of C major).

```
<AudioDescriptionScheme xsi:type="AudioChordPatternDS">
  <Key>
    <KeyNote>C</KeyNote>
  </Key>
  <Chord>
    <ChordBase RelativeChordNumber="0"/>
    <AudioSegment>
      <MediaTime>
        <MediaTimePoint>T00:00:00</MediaTimePoint>
        <MediaDuration>PT1M30S</MediaDuration>
      </MediaTime>
    </AudioSegment>
  </Chord>
  <Chord>
    <ChordBase RelativeChordNumber="5"/>
    <AudioSegment>
      <MediaTime>
        <MediaTimePoint>T00:00:10</MediaTimePoint>
        <MediaDuration>PT1M30S</MediaDuration>
      </MediaTime>
    </AudioSegment>
  </Chord>
  <Chord>
    <ChordBase RelativeChordNumber="7"/>
    <AudioSegment>
      <MediaTime>
        <MediaTimePoint>T00:00:20</MediaTimePoint>
        <MediaDuration>PT1M30S</MediaDuration>
      </MediaTime>
    </AudioSegment>
  </Chord>
  <Chord>
    <ChordBase RelativeChordNumber="0"/>
    <AudioSegment>
      <MediaTime>
        <MediaTimePoint>T00:00:30</MediaTimePoint>
        <MediaDuration>PT1M30S</MediaDuration>
      </MediaTime>
    </AudioSegment>
  </Chord>
</AudioDescriptionScheme>
```

6.10.2.4.3 Automatic Retrieval and Classification

A similarity matching of chord progression patterns could easily be done by comparing the chord progression patterns without considering the actual key. A standard mean square metric may then be used for evaluating the degree of similarity between the base keys of two chord patterns.

6.10.2.4.4 Applications

When using chord patterns to describe a certain musical mood, it is possible to query for musical content that is also characterized by one ore more representative chord patterns. This mechanism can serve as a search criterion in applications proposing a number of musical titles belonging to a particular musical style or genre or sounding similar to a title being proposed by the user as a “reference sound”.

6.11 Scales & Modes

6.11.1 Pitch Profile DS

The *PitchProfile* DS is an enhanced representation of monophonic melodies, based on the *Melody* DS and extended to allow a fine description of musical scales and modes on which melodies are built. The *PitchProfile* DS integrates the *Scale* type (part *of Melody* DS), which provides a description of the tuning system used by the melody, and extend it to capture the relative importance of the various components of a scale, such that the scale can be identified.

6.11.1.1 Syntax

```
<!-- ####################################### -->
<!-- Definition of PitchProfile DS           -->
<!-- ####################################### -->
<complexType name="PitchProfileDSType">
  <complexContent>
    <extension base="mpeg7:MelodyType">
      <sequence>
        <element name="ReferencePitch" type="mpeg7:ReferencePitchType"
                 minOccurs="0"/>
        <element name="TransposingRatio" type="float" default="12"
                 minOccurs="0"/>
        <element name="ProfileWeights" type="mpeg7:ProfileWeightsType"
                 minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.11.1.2 Semantics

Name	Definition
`PitchProfileType`	A structure containing optional elements that support the description of the mode on which the melody is built. The mode can be either a traditional basis scale (e.g. the major scale) or any mode derived from this basis scale (e.g. Dorian, Phrygian, etc.).
`ReferencePitch`	A container for the base pitch of the mode, either in absolute frequency expressed in unit of Hz, or a pitch marker using a semitone scale, in case the mode is related to a precise western music temperament. The reference pitch is to the mode what the key is to the scale, i.e. the only way to distinguish a mode transposed from another, for instance C Mixolydian and G Mixolydian. For definition of ReferencePithType see subclause 6.11.2. *Example:* The major scale has seven notes and seven different modes. In C major the mode starting on the major sixth (a 9 semitones interval) is the A Aeolian mode. A melody built on this mode should therefore have “A” as reference pitch.
`TransposingRatio`	A ratio of frequency expressed in semitones, specifying the point at which the mode repeats. The default value of this descriptor is taken to be 12. A *TransposingRatio* value of 0 (zero) should be understood as describing a musical mode that does not repeat.
`ProfileWeights`	An array containing the relative prevalence of each pitch class (or each degree) of the mode. The prevalence of a given degree is its cumulative duration in the melody. For instance in a melody using the A Dorian mode, the prevalence of the major third (a two semitones interval) would be the cumulative duration of all Bs in the melody. The relative prevalence is obtained by normalizing the prevalence by the base pitch prevalence. (The prevalence is bounded to 1.0) The following formula (informative) is applicative: $ProfileWeights\left[n\right]=\max\left(\frac{Prevalence\left[n\right]}{Prevalence\left[0\right]},1\right)$, where *Prevalence*[*i*] is the cumulative duration of ith degree (starting from index zero). For definition of ProfileWeightsType, see subclause 6.11.3.

6.11.1.3 Instantiation requirements

In order to guarantee a proper instantiation of this description scheme, the following requirements have to be fulfilled:

- In any case the *Scale* D of the *Melody* DS must be instantiated as soon as the *ProfileWeights* D is instantiated.
- Although the *Scale* D and the *ProfileWeights* D are both float vectors addressing melody degrees features, they do not have necessarily the same length, but the i^{th} element of the *ProfileWeights* vector corresponds to the i^{th} element of the Scale vector. In most cases, the last element of the *Scale* D is the *TransposingRatio*, which is equivalent to the base pitch regarding its hierarchical position in the scale or in the mode. There is therefore no need to attach a weight to this last element. For an instantiation example see subclause 6.11.1.5.

6.11.1.4 Usage and extraction

6.11.1.4.1 Automatic retrieval and recommendation

When using musical scales and/or modes to refer to a particular musical style or genre it is possible to query for musical content that is also characterized by one ore more representative scales and/or modes. This mechanism can serve as a search criterion in applications proposing a number of musical titles belonging to a particular musical style or genre or sounding similar to a title being proposed by the user as a “reference sound”.

6.11.1.4.2 Usage of the transposing ratio

In most cases the tuning system will be the equal-tempered scale and the transposing ration will be 12. In other words a given musical scale or a given musical mode repeats itself after 12 semitones, i.e. after an octave. However the last value of the Scale vector should *only* be understood as the frequency ratio after which the *tuning system* repeats itself, whereas the transposing ratio of a given mode built on this tuning system can be different:

- Considering for instance a scale built on the repetition of a minor second and a major second. Such a scale has been used and defined by the 20th century French composer Olivier Messiaen as a “limited-transposition mode”, meaning that it can only be transposed (semitone by semitone) a small number of times, in opposition to traditional scales (e.g. the major scale that can be transposed 12 times). Such a mode can be described with the *PitchProfile* DS by assigning a non-zero weight to degrees 1, 3, 4, 6, 7, 9, and 10, and using a *TransposingRatio*’s value of 12. An alternative description consists of using a *TransposingRatio*’s value of 3 and a *ProfileWeights* vector of length 2, with a non-zero weight assigned to its first element. With this latter representation the degrees 0 (base degree), 3, 6 and 9 on one hand, and 1, 4, 7 and 10 on the other hand, are considered as equivalents, and the *TransposingRatio* reflects the “limited-transposition mode” property, in accordance to the composer’s thought.

- Considering now a scale built on the following repetitive intervals: a minor third, a major third and another minor third (or in semitones intervals, 3-4-3). Starting from C1 such a scale would be: C1, Eb1, G1, Bb1, Db2, F2, Ab2, B2, D#3, etc, and would reach its octave transposition only after 6 octaves. It can easily be described with the *PitchProfile* DS, by using a 9-length *ProfileWeights* vector, assigning a non-zero weight to degrees 3 and 7, and using a *TransposingRatio*’s value of 10.

6.11.1.4.3 Playlist generation

Playlist generation based on similarity according to scales and modes is an interesting functionality in new intelligent HIFI systems. Such systems will give end-users the possibility to build their own playlists according to similarity measures, a measure that is enabled by the scales and modes description scheme.

6.11.1.4.4 Extraction

Whereas automatic extraction from polyphonic data is still an open issue, the *PitchProfile* DS can be extracted from monophonic audio files, under the assumption of the equal-tempered tuning system with of *TransposingRatio*’s value of 12. The extraction procedure consists in the following steps:

1. *Extraction of instantaneous fundamental frequencies.* For 44.1 kHz sampled files significant results have been obtained with a hop size of 256 and a 2940-length analysis window, for a frequency output range of 60 Hz to 11.025 kHz. The fundamental frequency is therefore estimated 172 times per second. This extraction can be easily performed with the YIN algorithm[1].

1) de Cheveigné, A., and Kawahara, H. (2002). "*YIN, a fundamental frequency estimator for speech and music*", J. Acoust. Soc. Am., 111, 1917-1930.

2. *Detection of attacks and pitch segmentation.* The pitch standard deviation is computed inside a fixed-length window, moving along the output of step 1. An attack is detected each time the standard deviation falls under a given threshold; a release is detected each time the standard deviation rises above the threshold. The output of step 1 is therefore indexed with attack and release timestamps. . To each attack a note is assigned, whose duration is defined as the time interval between the attack and the release. The pitch is of the note is defined at the next step.

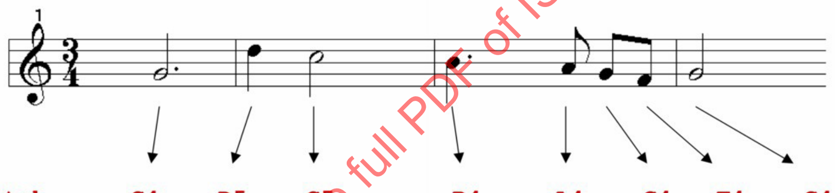
3. *Pitch quantization and pitch class prevalence computation.* To each attack-release interval the closest theoretic pitch to the interval's median frequency is assigned. The spectrum of pitches is then reduced to pitch classes modulo the *TransposingRatio*. For each pitch class the *prevalence* (i.e. the cumulative duration – 6.11.2.2) is computed; the set of all pitch classes and the corresponding prevalence vector is called *pitch profile*.

4. Selection of the *base pitch*. The *ReferencePitch* is chosen to be the "most important pitch of the profile". The pitch profile is compared to theoretic profiles in order to select the reference. For major and minor melodies, Krumhansl and Kessler's key finding profiles[2] give satisfying results.

5. Computation of *ProfileWeights*. Once the base pitch has been identified, the *ProfileWeights* descriptor is instantiated according to the formula given in subclause 6.11.1.2.

6.11.1.5 Example

The following example is an excerpt from the song "Moon River" by Henry Mancini:



```
Pitch       G4   D5   C5        B4    A4   G4   F4   G4
Duration   3.0  1.0  2.0       1.5   0.5  0.5  0.5  2.0

Base Pitch : G
Pitch classes  : G# A    A# B    C    C# D    D# E F    F#
Rel prevalence : 0  0.09 0  0.27 0.36 0  0.18 0  0 0.09 0
```

Figure AMD2.4 – PitchProfile description of "Moon River"

Using *PitchProfile DS*, it could be instantiated in the following manner:

```
<AudioDescriptionScheme xsi:type="PitchProfileType">
  <Scale>1 2 3 4 5 6 7 8 9 10 11 12</Scale>
  <ReferencePitch>
    <BasePitchNumber>7</BasePitchNumber>
  </ReferencePitch>
  <TransposingRatio>12</TransposingRatio>
  <ProfileWeights>0 0.09 0 0.27 0.36 0 0.18 0 0 0.09 0</ProfileWeights>
</AudioDescriptionScheme>
```

2) Krumhansl, C. L. (1990). "*Cognitive Foundations of Musical Pitch*". New York: Oxford University Press.

6.11.2 ReferencePitch

The *ReferencePitch* contains information about the “base pitch” or “base frequency” of the mode, i.e. the note on which the whole mode is built. The reference can be described either in absolute frequency expressed in unit of Hz, or with a pitch marker using a semitone scale. For instance, the reference of the Phrygian mode derived from the C-major scale (C D E F G A B) will be E; the reference of the minor pentatonic mode derived from the C-major pentatonic scale (C D E G A) would be A.

6.11.2.1 Syntax

```
<!-- ################################### -->
<!-- Definition of ReferencePitch D      -->
<!-- ################################### -->
<complexType name="ReferencePitchType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="BaseFrequency" minOccurs="0">
          <simpleType>
            <restriction base="float">
              <minInclusive value="0.0"/>
            </restriction>
          </simpleType>
        </element>
        <element name="BasePitchNumber" minOccurs="0">
          <simpleType>
            <restriction base="integer">
              <minInclusive value="0"/>
            </restriction>
          </simpleType>
        </element>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.11.2.2 Semantics

Name	Definition
`ReferencePitchType`	The pitch (or frequency) on which the mode is built (i.e. the fundamental degree of the mode), expressed either in absolute frequency or with a positive integer referring to a given pitch class (see below).
`BaseFrequency`	The absolute frequency of the first degree of the mode, expressed in units of Hertz, when the tuning system used is NOT the equal-tempered context.
`BasePitchNumber`	The pitch class of the first degree of the mode, when the tuning system used is the equal-tempered context. The *BasePitchNumber* is an MPEG-7 positive integer, with the following correspondence: 0=”C”, 1=”Db”, 2=”D”, etc. Note that enharmonic features are therefore not considered here.

6.11.2.3 Usage and extraction

6.11.2.3.1 Content retrieval

The *ReferencePitch* offers a new criterion for music browsing. Music composers for instance might have a strong interest in searching for musical excerpts with a particular reference pitch. Consider a composer writing a piece in Bb major, he/she might ask a search engine to find music files in which the melody uses Bb as the reference pitch.

Another good example is novice instrumentalists looking for small pieces for music studying. Considering for instance that a young violoncellist, who only knows the first position, is unable to play an A-flat without any finger extensions. The *ReferencePitch*, combined with the *ProfileWeights*, can help him finding music pieces where the A-flat never appears. Indeed the following example indicates the presence in the melody of pitches D, F, G, A and C, in other words the D minor pentatonic scale, which does not use any A-flat. Typically such a scale can be played on the first position of the cello, without any extensions. The following example shows the usage of PitchProfileType by extending the MelodyType:

```
<AudioDescriptionScheme xsi:type="PitchProfileType">
   <MelodyContour>
      <Contour>2 0 0</Contour>
      <Beat>1 2 2 3</Beat>
   </MelodyContour>
   <ReferencePitch>
      <BasePitchNumber>7</BasePitchNumber>
   </ReferencePitch>
   <TransposingRatio>12</TransposingRatio>
   <ProfileWeights>0.09 0 0.27 0.36 0 0.18 0 0 0.09 0</ProfileWeights>
</AudioDescriptionScheme>
```

6.11.2.3.2 Extraction

See subclause 6.11.1.4.4.

6.11.3 ProfileWeights D

The *ProfileWeights* descriptor is intended to describe which degrees of a given tuning system are used by a melody, and the relative importance of each degree to the melody’s reference degree.

6.11.3.1 Syntax

```
<!-- ############################################### -->
<!-- Definition of ProfileWeights D                  -->
<!-- ############################################### -->
<simpleType name="ProfileWeightsType">
  <restriction base="mpeg7:floatVector"/>
</simpleType>
```

6.11.3.2 Semantics

Name	Definition
`ProfileWeightsType`	Sequence of weights describing the relative importance of each degree in the mode. Each weight is defined as the relative prevalence (or cumulative duration) of each pitch class (or each degree) of the mode. The following formula (informative) is applicative: $$ProfileWeights[n] = \max\left(\frac{prevalence[n]}{prevalence[0]}, 1\right),$$ where *Prevalence*[*i*] is the cumulative duration of ith degree of the mode (starting from index zero).

6.11.3.3 Usage and Extraction

6.11.3.3.1 Designing distances on the melodies' space

In the equal-tempered context a straightforward distance on the descriptors' space is the *cosine* distance. In this case the mode descriptor is reduced to an 11-dimensional vector of weights and the following formula (informative) has to be applied for a cosine distance between melodies A & B:

$$d(A,B) = \frac{\sum_{i=1}^{11} a_i b_i}{\sqrt{\sum_{i=1}^{11} a_i^2 \sum_{i=1}^{11} b_i^2}}$$

where $(a_i)_{1 \le i \le 11}$ and $(b_i)_{1 \le i \le 11}$ are instances of the *ProfileWeights* descriptor for melodies A & B.

With such a distance melodies may easily be grouped into clusters and sorted by order of similarity. The following issues are addressed:

- "Given an input melody, I want an engine search to recover all the melodies in a given database that bear some significant resemblance with it, on a scale or mode similarity criterion."
- "Given a database of melodies, can I extract some clusters that have a musical meaning in terms of scale or mode similarity?"
- "Given a set of melodies, can I construct relevant semantic links on this set? On which criterias can I say that melody A is closer to melody B than melody C?"

6.11.3.3.2 Extraction

See subclause 6.11.1.4.4.

6.12 Enhanced AudioSignature

The recognition performance of the AudioSignatureType descriptor is sufficient for the vast majority of applications like broadcast monitoring, recognition of personal audio on PCs etc. There are, however, commercially interesting applications which require a reliable recognition of extremely distorted audio signals, like GSM-coded cell phone signals, which are not accommodated by the AudioSignatureType. This is addressed by using the Enhanced AudioSignature DS that provides a higher recognition rate on strong distorted signals as by using the AudioSignature Type.

6.12.1 Syntax

```
<!-- ################################################################## -->
<!-- Definition of EnhancedAudioSignatureDS                              -->
<!--################################################################## -->
<complexType name="EnhancedAudioSignatureType">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="Envelope" type="mpeg7:AudioSpectrumEnvelopeType"
                 minOccurs="1" maxOccurs="1"/>
        <element name="Flatness" type="mpeg7:AudioSignatureType"
                 minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.12.2 Semantics

Name	Definition
`EnhancedAudioSignatureType`	A structure containing a condensed representation as a unique content identifier for an audio signal for the purpose of automatic identification for extremely distorted audio signals
`Envelope`	The spectrum envelope of the signal of `AudioSpectrumEnvelopeType` data.
`Flatness`	The spectrum flatness of the signal of `AudioSignatureType` data

6.12.3 Instantiation requirements

In order to constitute a valid EnhancedAudioSignatureType descriptor, the following requirements have to be satisfied:

- The syntax of the `Envelope` part is restricted to `SeriesOfVectorType`.
- The `log` field of the `Envelope` part has to be instantiated, as provided by the `SeriesOfVectorType` syntax.
- The `logBase` parameter, as provided by the `SeriesOfVectorType` syntax, is fixed at 10.0
- The `loEdge` parameter, as provided by the `AudioSpectrumAttributeGrp` syntax, is fixed at 250 Hz.

- The `hiEdge` parameter, as provided by the `AudioSpectrumAttributeGrp` syntax, is fixed at 4000 Hz.
- The `octaveResolution` parameter, provided by the `AudioSpectrumAttributeGrp` syntax, must be 1 or less.
- There must be no overlap between subsequent calculations. Therefore the `HopSize` is fixed at 30 ms.
- The Scaling Ratio as provided by `ScalableSeriesType` can adopt values of 1, 2 or 4.
- It is strongly recommended to use the underlying `AudioSpectrumEnvelope` descriptor with a `hopSize` = 30ms (i.e. a hop size equal to the window length, without overlap between subsequent frames).

6.12.4 Usage and Examples

There are numerous examples of applications for the *EnhancedAudioSignature* description scheme conceivable, including automatic identification of an unknown piece of audio based on a database of registered audio items. The description scheme is inherited from the AudioSignature description scheme and the basic innovation is the introduction of the AudioSignatureEnvelope type. That enables a broader range of application and quality improvement of previous applications.

The AudioSignature description scheme may be used for identifying slightly distorted signals, but the enhanced descriptor performs much better on strongly distorted signals as for e.g. recorded music from cellular phones. The element "Envelope" is based on the feature AudioSpectrumEnvelope and its feature space is orthogonal to that of AudioSignatureDS used feature AudioSpectrumFlatness which enables a lot of other application areas.

A similarity matching of undistorted or slightly distorted query items might be done as described in ISO/IEC 15938-4:2002. In order to match similarity between strongly distorted signals and the reference signal it is recommended to use the data of AudioSpectrumEnvelopeType. A standard mean square metric may then be used for evaluating the degree of similarity between two signatures. In order to improve the performance of recognizing music the difference of adjacent Scalable Series might be calculated.

6.13 Perceptual Attributes

Perceptual attributes are concerned with classification of musical genre, instrumentation, lyrics, ensemble type, emotional content, tempo and energy. The representation of each of these attributes cannot be uniform, as they have different semantic representations. For example, whereas a piece of music is perceived to have a single tempo, which can therefore be represented on a numerical scale, it is not generally perceived to belong to a single musical genre, but rather has degrees of membership in multiple genres.

To accommodate these different semantic representations of perceptual attributes, we use three different syntactical representations, each of which matches a different semantic representation. These three types are Distribution, Classification, and Linear.

The schema types presented in this subclause should be used only for describing the perceptual attributes or perceived qualities of a song.

6.13.1 PerceptualGenreDistributionElementType D

Distribution perceptual attributes are those attributes that take a distribution of values. Each element within the distribution contains a classification, representing the specific class of that element, and a score, representing the relative weighting of that distribution element within the distribution. The score value ranges from 1 to 255 and is understood to have an inherent ordering, such that a score value of 1 represents a low weighting within the distribution, while a score value of 255 represents a high weighting within the distribution.

The four distribution perceptual attributes are Perceptual Genre, Perceptual Mood, Perceptual Lyrics, and Perceptual Instrumentation.

Perceptual Genre describes the general style of a piece of music. A particular piece of music is understood to have potential degrees of membership in more than one genre or style of music, and so the full range of genres that describe a particular song is represented by a distribution of Perceptual Genres. For example, a particular song may be perceived to be predominantly a Rock song with elements of Country so its Perceptual Genre will be represented with values for both Rock and Country.

Perceptual Genre is useful for finding songs which embody the same or similar styles of music.

An instance of PerceptualGenreDistributionElementType represents a single Perceptual Genre value within a distribution of Perceptual Genres. The class of a particular PerceptualGenreDistributionElementType instance is represented by its id, which is a reference to a term in a classification scheme.

The degree of relevance of each instance within a distribution is represented by its score. The score value ranges from 1 to 255, with a value of 1 indicating a minimal degree of membership in the perceptual genre and a value of 255 representing a maximal degree of membership in the perceptual genre.

6.13.1.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of PerceptualGenreDistributionElementType D           -->
<!-- ################################################################ -->
<complexType name="PerceptualGenreDistributionElementType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <attribute name="id" type="mpeg7:termReferenceType" use="required"/>
      <attribute name="score" use="required">
        <simpleType>
          <restriction base="integer">
            <minInclusive value="1"/>
            <maxInclusive value="255"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
```

6.13.1.2 Semantics

Name	*Definition*
`PerceptualGenreDistributionElementType`	This type represents a perceptual genre value within a distribution of perceptual genres.
`id`	ID of the specific perceptual genre represented by this distribution element. A reference to a term from a predefined classification scheme. For an example perceptual genre classification scheme, see Genre Perceptual Attribute CS in Annex D.3.
`score`	Value (1 to 255) for this perceptual genre distribution element denoting the degree of membership in the specified perceptual genre. A value of 1 indicates a minimal degree of membership, while a value of 255 indicates a maximal degree of membership.

6.13.1.3 Usage

The following table includes some of the perceptual genre terms. The classification scheme for perceptual genre can be found in Annex D.3.

Perceptual Genres	
ID	**Name**
urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.1	Alternative
urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.1.1	Alt Dance
urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.1.2	Alt Power Pop
urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.2	Blues
urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.2.1	Classic
urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.2.2	Early Acoustic

The ids used for the perceptual genre classification scheme are prefixed with urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:. This is followed by a hierarchical dot-separated identification scheme used to specify a particular perceptual genre. The first element in this id is always 100 for perceptual genre. This is followed by one or more dot-separated elements indicating the specific perceptual genre and possibly sub-genre. For example, the id suffix 100.1 indicates the perceptual genre Alternative, as seen in the table above. The id suffix 100.1.1 indicates the Alt Dance sub-genre underneath the parent genre Alternative.

6.13.2 PerceptualGenreDistributionType D

PerceptualGenreDistributionType is a container for PerceptualGenreDistributionElementType instances. An instance of PerceptualGenreDistributionType represents the distribution of Perceptual Genres for a given song.

6.13.2.1 Syntax

```
<!-- ############################################################ -->
<!-- Definition of PerceptualGenreDistributionType D              -->
<!-- ############################################################ -->
<complexType name="PerceptualGenreDistributionType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="PerceptualGenre"
                 type="mpeg7:PerceptualGenreDistributionElementType"
                 maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.13.2.2 Semantics

Name	*Definition*
`PerceptualGenreDistributionType`	This type represents a distribution of Perceptual Genres.
`PerceptualGenre`	A specific Perceptual Genre classification and degree of membership within the distribution.

6.13.2.3 Usage and Examples

PerceptualGenreDistributionType instances represent a distribution of Perceptual Genres. Perceptual Genre is an example of a distribution perceptual attribute because any particular song may have a degree of "membership" in one or more Perceptual Genres. PerceptualGenreDistributionType contains a sequence of one or more PerceptualGenreDistributionElementType instances, each one representing a particular Perceptual Genre within the distribution. The following xml scheme shows an example of the usage of the PerceptualGenreDistribution type.

```
<PerceptualGenreDistributionType>
  <PerceptualGenre
    id="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.15.11"
    score="43"/>
  <PerceptualGenre
    id="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.15.15"
    score="194"/>
  <PerceptualGenre
    id="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.15.16"
    score="18"/>
</PerceptualGenreDistributionType>
```

This example shows a perceptual genre distribution for a song that belongs primarily to the Rock/Pop Metal perceptual genre/sub-genre, but that also has some lesser degree of membership in the Rock/Hard Rock and Rock/Pop Rock genre/sub-genres.

6.13.3 PerceptualMoodDistributionElementType D

Perceptual Mood describes the basic emotional content of a piece of music. Like Perceptual Genre, each song is construed as having degrees of membership or expression in one or more basic moods. Examples of Perceptual Moods include Excited, Romantic, and Bitter.

Perceptual Mood is useful for characterizing the emotional content of music, and for finding songs that express similar emotional content.

An instance of PerceptualMoodDistributionElementType represents a single Perceptual Mood value within a distribution of Perceptual Moods. The class of a particular PerceptualMoodDistributionElementType instance is represented by its id, which is a reference to a term in a classification scheme.

The degree to which a particular mood is displayed is represented by its score. The score value ranges from 1 to 255, with a value of 1 indicating a minimal degree of display and a value of 255 representing a maximal degree of display.

6.13.3.1 Syntax

```
<!-- ########################################################## -->
<!-- Definition of PerceptualMoodDistributionElementType D        -->
<!-- ########################################################## -->
<complexType name="PerceptualMoodDistributionElementType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <attribute name="id" type="mpeg7:termReferenceType" use="required"/>
      <attribute name="score" use="required">
        <simpleType>
          <restriction base="integer">
            <minInclusive value="1"/>
            <maxInclusive value="255"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
```

6.13.3.2 Semantics

Name	*Definition*
`PerceptualMoodDistributionElementType`	This type represents a perceptual mood value within a distribution of perceptual moods.
`id`	ID of the specific perceptual mood represented by this distribution element. A reference to a term from a predefined classification scheme. For an example perceptual mood classification scheme, see Mood Perceptual Attribute CS in Annex D.4.
`score`	Value (1 to 255) for this perceptual mood distribution element denoting the degree to which the specified mood is displayed. A value of 1 indicates a minimal degree, while a value of 255 indicates a maximal degree.

6.13.3.3 Usage

The following table includes some of the perceptual mood terms. The classification scheme for perceptual mood can be found in Annex D.4.

Perceptual Moods	
ID	**Name**
urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.1	Fun / Cheerful
urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.1.1	Celebratory
urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.1.2	Enthusiastic
urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.3	Down / Negative
urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.3.1	Bitter
urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.3.2	Brooding

The ids used for the perceptual mood classification scheme are prefixed with urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:. This is followed by a hierarchical dot-separated identification scheme used to specify a particular perceptual mood. The first element in this id is always 200 for perceptual mood. This is followed by one or more dot-separated elements indicating the specific perceptual mood. For example, the id suffix 200.1 indicates the perceptual mood group Fun / Cheerful, as seen in the table above. The id suffix 200.1.1 indicates the more specific perceptual mood Celebratory, which is a child of the Fun / Cheerful mood group.

6.13.4 PerceptualMoodDistributionType D

PerceptualMoodDistributionType is a container for PerceptualMoodDistributionElementType instances. An instance of PerceptualMoodDistributionType represents the distribution of Perceptual Moods for a given song.

6.13.4.1 Syntax

```
<!-- ############################################################### -->
<!-- Definition of PerceptualMoodDistributionType D                  -->
<!-- ############################################################### -->
<complexType name="PerceptualMoodDistributionType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="PerceptualMood"
                 type="mpeg7:PerceptualMoodDistributionElementType"
                 maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.13.4.2 Semantics

Name	*Definition*
`PerceptualMoodDistributionType`	This type represents a distribution of Perceptual Moods.
`PerceptualMood`	A specific Perceptual Mood classification and degree of membership within the distribution.

6.13.4.3 Usage and Examples

PerceptualMoodDistributionType instances represent a distribution of Perceptual Moods. Perceptual Mood is an example of a distribution perceptual attribute because any particular song may display varying degrees of different Perceptual Moods. PerceptualMoodDistributionType contains a sequence of one or more PerceptualMoodDistributionElementType instances, each one representing a particular Perceptual Mood.

The following example shows a perceptual mood distribution for a song that displays almost equal amounts of the moods Humorous and Playful. The song also displays the mood Upbeat to a lesser degree

```
<PerceptualMoodDistributionType>
  <PerceptualMood
    id="urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.1.7"
    score="116"/>
  <PerceptualMood
    id="urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.1.8"
    score="114"/>
  <PerceptualMood
    id="urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.1.13"
    score="25"/>
</PerceptualMoodDistributionType>
```

6.13.5 PerceptualLyricsDistributionElementTypeD

The Perceptual Lyrics perceptual attribute describes the content of the lyrics of a piece of music. Like Perceptual Genre and Perceptual Mood, each song is construed to have degrees of expression in one or more topics or themes of lyrical content. Examples of Perceptual Lyrics include Love / Romance, Happiness / Celebration, and Sadness / Loss.

Perceptual Lyrics is useful for characterizing the semantic or thematic content of the words of a song. An instance of PerceptualLyricsDistributionElementType represents a single Perceptual Lyrics value within a distribution of Perceptual Lyrics. The class of a particular PerceptualLyricsDistributionElementType instance is represented by its id, which is a reference to a term in a classification scheme.

The degree of prominence of a particular lyrical topic is represented by its score. The score value ranges from 1 to 255, with a value of 1 indicating a minimal degree of prominence and a value of 255 representing a maximal degree of prominence.

6.13.5.1 Syntax

```
<!-- ############################################################ -->
<!-- Definition of PerceptualLyricsDistributionElementType D      -->
<!-- ############################################################ -->
<complexType name="PerceptualLyricsDistributionElementType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <attribute name="id" type="mpeg7:termReferenceType" use="required"/>
      <attribute name="score" use="required">
        <simpleType>
          <restriction base="integer">
            <minInclusive value="1"/>
            <maxInclusive value="255"/>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
```

6.13.5.2 Semantics

Name	*Definition*
PerceptualLyricsDistributionElementType	This type represents a perceptual lyrics value within a distribution of perceptual lyrics.
id	ID of the specific perceptual lyrics type represented by this distribution element. A reference to a term from a predefined classification scheme. For an example perceptual lyrics classification scheme, see Lyrics Perceptual Attribute CS in Annex D.6.

Name	*Definition*
`score`	Value (1 to 255) for this perceptual lyrics distribution element denoting the degree of prominence of the specified perceptual lyrics type. A value of 1 indicates a minimal degree, while a value of 255 indicates a maximal degree.

6.13.5.3 Usage

The following table includes some of the perceptual lyrics terms. The classification scheme for perceptual lyrics can be found in Annex D.6.

Perceptual Lyrics	
ID	**Name**
urn:mpeg:mpeg7:cs:LyricsPerceptualAttributeCS:2006:400.1	Love / Romance
urn:mpeg:mpeg7:cs:LyricsPerceptualAttributeCS:2006:400.2	Happiness / Celebration
urn:mpeg:mpeg7:cs:LyricsPerceptualAttributeCS:2006:400.3	Sadness / Loss

The ids used for the perceptual lyrics classification scheme are prefixed with urn:mpeg:mpeg7:cs:LyricsPerceptualAttributeCS:2006:. This is followed by a hierarchical dot-separated identification scheme used to specify a particular perceptual lyrics type. The first element in this id is always 400 for perceptual lyrics. This is followed by one or more dot-separated elements indicating the specific perceptual lyrics type. For example, the id suffix 400.1 indicates the perceptual lyrics type Love / Romance, as seen in the table above, while the id suffix 400.2 indicates the perceptual lyrics type Happiness / Celebration.

6.13.6 PerceptualLyricsDistributionType D

PerceptualLyricsDistributionType is a container for PerceptualLyricsDistributionElementType instances. An instance of PerceptualLyricsDistributionType represents the distribution of Perceptual Lyrics for a given song.

6.13.6.1 Syntax

```
<!-- ############################################################ -->
<!-- Definition of PerceptualLyricsDistributionType D             -->
<!-- ############################################################ -->
<complexType name="PerceptualLyricsDistributionType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="PerceptualLyrics"
                 type="mpeg7:PerceptualLyricsDistributionElementType"
                 maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.13.6.2 Semantics

Name	*Definition*
`PerceptualLyricsDistributionType`	This type represents a distribution of Perceptual Lyrics.
`PerceptualLyrics`	A specific Perceptual Lyrics classification and degree of membership within the distribution.

6.13.6.3 Usage and Examples

PerceptualLyricsDistributionType instances represent a distribution of Perceptual Lyrics. Perceptual Lyrics is an example of a distribution perceptual attribute because the lyrics of a particular song may cover a range of topics to varying degrees. PerceptualLyricsDistributionType contains a sequence of one or more PerceptualLyricsDistributionElementType instances, each one representing a particular Perceptual Lyrics type within the distribution.

For example:

```
<PerceptualLyricsDistributionType>
  <PerceptualLyrics
    id="urn:mpeg:mpeg7:cs:LyricsPerceptualAttributeCS:2006:400.1"
    score="171"/>
  <PerceptualLyrics
    id="urn:mpeg:mpeg7:cs:LyricsPerceptualAttributeCS:2006:400.2"
    score="84"/>
</PerceptualLyricsDistributionType>
```

This example shows a perceptual lyrics distribution for a song whose lyrics are primarily about Love / Romance, but are also to a lesser degree about Happiness / Celebration.

6.13.7 PerceptualInstrumentationDistributionElementType

The Perceptual Instrumentation perceptual attribute describes the perceptual prominence of various musical instruments in a piece of music. Like Perceptual Genre and Perceptual Mood, each song is construed to have degrees of prominence of one or more musical instruments.

Perceptual Instrumentation is useful for characterizing the perceptually prominent instruments in a piece of music and identifying songs with similar instrumentation.

An instance of PerceptualInstrumentationDistributionElementType represents a single Perceptual Instrumentation value within a distribution of Perceptual Instrumentations. The class of a particular PerceptualInstrumentationDistributionElementType instance is represented by its id, which is a reference to a term in a classification scheme.

The degree of prominence of a particular perceptual instrumentation is represented by its score. The score value ranges from 1 to 255, with a value of 1 indicating a minimal degree of prominence and a value of 255 representing a maximal degree of prominence.

6.13.7.1 Syntax

```
<!-- ################################################################ -->
<!-- Definition of PerceptualInstrumentationDistributionElementType   -->
<!-- ################################################################ -->
<complexType name="PerceptualInstrumentationDistributionElementType">
   <complexContent>
      <extension base="mpeg7:AudioDType">
         <attribute name="id" type="mpeg7:termReferenceType" use="required"/>
         <attribute name="score" use="required">
            <simpleType>
               <restriction base="integer">
                  <minInclusive value="1"/>
                  <maxInclusive value="255"/>
               </restriction>
            </simpleType>
         </attribute>
      </extension>
   </complexContent>
</complexType>
```

6.13.7.2 Semantics

Name	*Definition*
`PerceptualInstrumentationDistributionElementType`	This type represents a perceptual instrumentation value within a distribution of perceptual instrumentations.
`id`	ID of the specific perceptual instrumentation represented by this distribution element. A reference to a term from a predefined classification scheme. For an example perceptual instrumentation classification scheme, see Instrumentation Perceptual Attribute CS in Annex D.5.
`score`	Value (1 to 255) for this perceptual instrumentation distribution element denoting the degree of prominence of the specified perceptual instrumentation. A value of 1 indicates a minimal degree, while a value of 255 indicates a maximal degree.

6.13.7.3 Usage

The following table includes some of the perceptual instrumentation terms. The classification scheme for perceptual instrumentation can be found in Annex D.5.

Perceptual Instrumentation	
ID	**Name**
urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.1	Guitars
urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.1.1	Acoustic Guitar
urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.1.2	Electric Guitar
urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.2	Strings
urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.2.1	Violin
urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.2.2	Cello

The ids used for the perceptual instrumentation classification scheme are prefixed with urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:. This is followed by a hierarchical dot-separated identification scheme used to specify a particular perceptual instrumentation type. The first element in this id is always 300 for perceptual instrumentation. This is followed by one or more dot-separated elements indicating the specific perceptual instrumentation type. For example, the id suffix 300.1 indicates the perceptual instrumentation type Guitars, as seen in the table above. The id suffix 300.1.1 indicates the more specific perceptual instrumentation Acoustic Guitar, which is a child of the Guitars perceptual instrumentation.

6.13.8 PerceptualInstrumentationDistributionType D

PerceptualInstrumentationDistributionType is a container for PerceptualInstrumentation Distribution ElementType instances. An instance of PerceptualInstrumentationDistributionType represents the distribution of Perceptual Instrumentation for a given song.

6.13.8.1 Syntax

```
<!-- ############################################################ -->
<!-- Definition of PerceptualInstrumentationDistributionType D    -->
<!-- ############################################################ -->
<complexType name="PerceptualInstrumentationDistributionType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="PerceptualInstrumentation"
                 type="mpeg7:PerceptualInstrumentationDistributionElementType"
                 maxOccurs="unbounded"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.13.8.2 Semantics

Name	*Definition*
`PerceptualInstrumentationDistributionType`	This type represents a distribution of Perceptual Instrumentations.
`PerceptualInstrumentation`	A specific Perceptual Instrumentation classification and degree of membership within the distribution.

6.13.8.3 Example

PerceptualInstrumentationDistributionType instances represent a distribution of Perceptual Instrumentations. Perceptual Instruementation is an example of a distribution perceptual attribute because there may be multiple instruments that stand out in a particular song. PerceptualInstrumentationDistributionType contains a sequence of one or more PerceptualInstrumentationDistributionElementType instances, each one representing a particular Perceptual Instrumentation type within the distribution.

For example:

```
<PerceptualInstrumentationDistributionType>
  <PerceptualInstrumentation
    id="urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.1.2"
    score="97"/>
  <PerceptualInstrumentation
    id="urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.1.3"
    score="87"/>
  <PerceptualInstrumentation
    id="urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.7.1"
    score="71"/>
</PerceptualInstrumentationDistributionType>
```

This example shows a perceptual instrumentation distribution for a song whose primary instruments are Electric Guitar, Bass, and Drum Set.

6.13.9 PerceptualLanguageType D

A classification perceptual attribute takes a single value out of several possible values. For example, the perceptual attribute Language takes on one of several possible languages, while the perceptual attribute Vocals taken one of the possible vocals types such as Male, Female, Mixed, etc.

The four classification perceptual attributes are Perceptual Language, Perceptual Sound, Perceptual Recording, and Perceptual Vocals.

Perceptual Language refers to the principal language of the lyrics of a song. Each song is understood to have at most one principal language.

An instance of PerceptualLanguageType represents the principal language of the lyrics of a song. The class of a particular PerceptualLanguageType instance is represented by its id, which is a reference to a term in a classification scheme.

6.13.9.1 Syntax

```
<!-- ############################################################ -->
<!-- Definition of PerceptualLanguageType D                       -->
<!-- ############################################################ -->
<complexType name="PerceptualLanguageType">
   <complexContent>
      <extension base="mpeg7:AudioDType">
         <attribute name="id" type="mpeg7:termReferenceType" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

6.13.9.2 Semantics

Name	*Definition*
`PerceptualLanguageType`	This type represents the perceptual language of a song's lyrics.
`id`	ID of the specific perceptual language represented by this instance. A reference to a term from a predefined classification scheme. For an example perceptual language classification scheme, see Language Perceptual Attribute CS in Annex D.7.

6.13.9.3 Usage and Examples

The PerceptualLanguageType attribute takes a single value out of several possible values. The possible values are identified by terms in a classification scheme. For example, English language lyrics are represented as:

```
<PerceptualLanguageType
  id="urn:mpeg:mpeg7:cs:LanguagePerceptualAttributeCS:2006:500.1"/>
```

6.13.10 PerceptualSoundType D

Perceptual Sound refers to the perceived overall instrumentation style of a song: whether a song sounds Acoustic (no electronic instrumentation), Electronic (predominantly electronic instrumentation) or Synthesized (predominantly synthesized).

An instance of PerceptualSoundType represents the overall instrumentation style of a song. The class of a particular PerceptualSoundType instance is represented by its id, which is a reference to a term in a classification scheme.

6.13.10.1 Syntax

```
<!-- ############################################################ -->
<!-- Definition of PerceptualSoundType D                          -->
<!-- ############################################################ -->
<complexType name="PerceptualSoundType">
   <complexContent>
      <extension base="mpeg7:AudioDType">
```

```
         <attribute name="id" type="mpeg7:termReferenceType" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

6.13.10.2 Semantics

Name	*Definition*
`PerceptualSoundType`	This type represents the overall perceptual sound type of a song.
`id`	ID of the specific perceptual sound type represented by this instance. A reference to a term from a predefined classification scheme. For an example perceptual sound type classification scheme, see Sound Perceptual Attribute CS in Annex D.8.

6.13.10.3 Usage and Examples

The PerceptualSoundType attribute takes a single value out of several possible values. The possible values are identified by terms in a classification scheme. For example, acoustic sound is represented as:

```
<PerceptualSoundType
  id="urn:mpeg:mpeg7:cs:SoundPerceptualAttributeCS:2006:600.1"/>
```

6.13.11 PerceptualRecordingType D

Perceptual Recording refers to the perceived overall sound quality of a song which reflects whether it is perceived as a live performance or a studio recording.

An instance of PerceptualRecordingType represents the overall sound quality / recording style of a song. The class of a particular PerceptualRecordingType instance is represented by its id, which is a reference to a term in a classification scheme.

6.13.11.1 Syntax

```
<!-- ############################################################# -->
<!-- Definition of PerceptualRecordingType D                       -->
<!-- ############################################################# -->
<complexType name="PerceptualRecordingType">
   <complexContent>
      <extension base="mpeg7:AudioDType">
         <attribute name="id" type="mpeg7:termReferenceType" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

6.13.11.2 Semantics

Name	*Definition*
`PerceptualRecordingType`	This type represents the overall perceptual recording type of a song.
`id`	ID of the specific perceptual recording type represented by this instance. A reference to a term from a predefined classification scheme. For an example perceptual recording type classification scheme, see Recording Perceptual Attribute CS in Annex D.9.

6.13.11.3 Usage and Examples

The PerceptualRecordingType attribute takes a single value out of several possible values. The possible values are identified by terms in a classification scheme.

For example, a live / concert recording is represented as:

```
<PerceptualRecordingType
  id="urn:mpeg:mpeg7:cs:RecordingPerceptualAttributeCS:2006:601.2"/>
```

And a studio recording is represented as:

```
<PerceptualRecordingType
  id="urn:mpeg:mpeg7:cs:RecordingPerceptualAttributeCS:2006:601.1"/>
```

6.13.12 PerceptualVocalsType D

Perceptual Vocals refers to the perceived gender quality of the voices in a song. Each song is understood to be exactly one of either predominantly male voices, predominantly female voices, both male and female voices, or no voices (i.e. an instrumental song).

An instance of PerceptualVocalsType represents the perceived gender quality of the voices in a song. The class of a particular PerceptualVocalsType instance is represented by its id, which is a reference to a term in a classification scheme.

6.13.12.1 Syntax

```
<!-- ########################################################### -->
<!-- Definition of PerceptualVocalsType D                         -->
<!-- ########################################################### -->
<complexType name="PerceptualVocalsType">
   <complexContent>
      <extension base="mpeg7:AudioDType">
         <attribute name="id" type="mpeg7:termReferenceType" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

6.13.12.2 Semantics

Name	*Definition*
`PerceptualVocalsType`	This type represents the perceived gender quality of the voices in a song.
`id`	ID of the specific perceptual vocals type represented by this instance. A reference to a term from a predefined classification scheme. For an example perceptual vocals type classification scheme, see Vocals Perceptual Attribute CS in Annex D.10.

6.13.12.3 Usage and Examples

The PerceptualVocalsType attribute takes a single value out of several possible values. The possible values are identified by terms in a classification scheme.

For example, a song with predominantly male vocals would be represented as:

```
<PerceptualVocalsType
  id="urn:mpeg:mpeg7:cs:VocalsPerceptualAttributeCS:2006:602.1"/>
```

And a song with predominantly female vocals is represented as:

```
<PerceptualVocals
  id="urn:mpeg:mpeg7:cs:VocalsPerceptualAttributeCS:2006:602.2"/>
```

6.13.13 PerceptualEnergyType D

Linear perceptual attributes are those attributes that only take a scalar value, such that values are understood to have an inherent ordering. An example of such a value would be the attribute PerceptualTempo. PerceptualTempo is understood to be represented on a single axis, so that any two songs can be assigned a single value representing their perceived tempo. In addition, any two songs may be ordered with respect to their PercepualTempo. In this case, the value can be represented as an integer between 1 and 255, such that the slowest perceived tempo is indicated by a value of 1, and the fastest perceived tempo is indicated by 255.

The four linear perceptual attribute are Perceptual Energy, Perceptual Valence, Perceptual Beat, and Perceptual Tempo.

Perceptual Energy refers to the perceived energy or intensity of a song. Songs with low Perceptual Energy values are generally perceived as being quiet, slow, mellow, and lacking in sharp changes. Songs with high Perceptual Energy values are generally perceived as being loud, fast, upbeat, and featuring aggressive emotional qualities.

Perceptual Energy is useful for grouping songs broadly alike in tone or suitability for particular activities (e.g. dancing, studying, dining etc.)

An instance of PerceptualEnergyType represents the perceived energy level of a song. The score attribute, which ranges from 1 to 255, represents the degree of intensity of this attribute. Thus, a value of 1 indicates a low degree of Perceptual Energy, while a value of 255 indicates a high degree of Perceptual Energy.

6.13.13.1 Syntax

```
<!-- ############################################################ -->
<!-- Definition of PerceptualEnergyType D                         -->
<!-- ############################################################ -->
<complexType name="PerceptualEnergyType">
   <complexContent>
      <extension base="mpeg7:AudioDType">
         <attribute name="score" use="required">
            <simpleType>
               <restriction base="integer">
                  <minInclusive value="1"/>
                  <maxInclusive value="255"/>
               </restriction>
            </simpleType>
         </attribute>
      </extension>
   </complexContent>
</complexType>
```

6.13.13.2 Semantics

Name	*Definition*
`PerceptualEnergyType`	This type represents the perceptual energy of a song.
`score`	Value (1 to 255) representing the degree of perceptual energy displayed by a song. A value of 1 indicates a minimal degree, while a value of 255 indicates a maximal degree.

6.13.13.3 Usage and Examples

The PerceptualEnergyType attribute contains a single score attribute that indicates the level of Perceptual Energy displayed by a song. For example:

```
<PerceptualEnergyType score="221"/>
```

This represents a song with a high level of Perceptual Energy. Songs with high Perceptual Energy values are generally perceived as being loud, fast, upbeat, and featuring aggressive emotional qualities. In contrast, songs with low Perceptual Energy values are generally perceived as being quiet, slow, mellow, and lacking in sharp changes.

6.13.14 PerceptualValenceType D

Perceptual Valence refers to the overall perceived positiveness / pleasantness (high Perceptual Valence) or negativeness / unpleasantness (low Perceptual Valence) of a piece of music. Songs with low Perceptual Valence are generally perceived to be harsh, aggressive or sad. Songs high in Perceptual Valence are generally perceived to be happy, romantic or upbeat.

Perceptual Valence is useful for grouping songs with broadly similar mood pleasantness or emotional content.

An instance of PerceptualValenceType represents the perceived valence level of a song. The score attribute, which ranges from 1 to 255, represents the degree of intensity of this attribute. Thus, a value of 1 indicates a low degree of Perceptual Valence, while a value of 255 indicates a high degree of Perceptual Valence.

6.13.14.1 Syntax

```
<!-- ############################################################ -->
<!-- Definition of PerceptualValenceType D                        -->
<!-- ############################################################ -->
<complexType name="PerceptualValenceType">
   <complexContent>
      <extension base="mpeg7:AudioDType">
         <attribute name="score" use="required">
            <simpleType>
               <restriction base="integer">
                  <minInclusive value="1"/>
                  <maxInclusive value="255"/>
               </restriction>
            </simpleType>
         </attribute>
      </extension>
   </complexContent>
</complexType>
```

6.13.14.2 Semantics

Name	*Definition*
`PerceptualValenceType`	This type represents the perceptual valence of a song.
`score`	Value (1 to 255) representing the degree of perceptual valence displayed by a song. A value of 1 indicates a minimal degree, while a value of 255 indicates a maximal degree.

6.13.14.3 Usage and Examples

The PerceptualValenceType attribute contains a single score attribute that indicates the level of Perceptual Valence displayed by a song. For example:

```
<PerceptualValenceType score="173"/>
```

This represents a song with a high level of Perceptual Valence. Songs high in Perceptual Valence are generally perceived to be happy, romantic or upbeat. Songs with low Perceptual Valence are generally perceived to be harsh, aggressive or sad.

6.13.15 PerceptualBeatType D

Perceptual Beat refers to the perceived heaviness of the basic beat of a song. Songs low in Perceptual Beat are perceived as having a light, unobtrusive beat whereas songs high in Perceptual Beat are perceived as having heavy, prominent beats.

Perceptual Beat is useful for grouping songs with broadly similar beat prominence.

An instance of PerceptualBeatType represents the perceived heaviness of the beat of a song. The score attribute, which ranges from 1 to 255, represents the degree of intensity of this attribute. Thus, a value of 1 indicates Perceptual Beat of low intensity, while a value of 255 indicates a Perceptual Beat of high intensity.

6.13.15.1 Syntax

```
<!-- ############################################################# -->
<!-- Definition of PerceptualBeatType D                            -->
<!-- ############################################################# -->
<complexType name="PerceptualBeatType">
   <complexContent>
      <extension base="mpeg7:AudioDType">
         <attribute name="score" use="required">
            <simpleType>
               <restriction base="integer">
                  <minInclusive value="1"/>
                  <maxInclusive value="255"/>
               </restriction>
            </simpleType>
         </attribute>
      </extension>
   </complexContent>
</complexType>
```

6.13.15.2 Semantics

Name	*Definition*
`PerceptualBeatType`	This type represents the perceptual beat of a song.
`score`	Value (1 to 255) representing the heaviness of the perceptual beat displayed by a song. A value of 1 indicates a light perceptual beat, while a value of 255 indicates a heavy perceptual beat.

6.13.15.3 Usage and Examples

The PerceptualBeatType attribute contains a single score attribute that indicates the heaviness of the Perceptual Beat displayed by a song. For example:

```
<PerceptualBeatType score="221"/>
```

This represents a song with a high Percetual Beat. Songs high in Perceptual Beat are perceived as having heavy, prominent beats, whereas songs low in Perceptual Beat are perceived as having a light, unobtrusive beat.

6.13.16 PerceptualTempoType D

Perceptual Tempo refers to the perceived subjective tempo of a song. Perceptual Tempo does not reflect the exact tempo, as for example might be reflected by the beats-per-minute (BPM) or similar measures but rather the perceived subjective speed of a piece of music. Songs low in Perceptual Tempo are perceived as being slow, whereas songs high in Perceptual Tempo are perceived as being fast.

Perceptual Tempo is useful for grouping songs with broadly similar perceived speed. An instance of PerceptualTempoType represents the perceived tempo of a song. The score attribute, which ranges from 1 to 255, represents the degree of intensity of this attribute. Thus, a value of 1 indicates a low Perceptual Tempo, while a value of 255 indicates a high Perceptual Tempo.

6.13.16.1 Syntax

```
<!-- ############################################################ -->
<!-- Definition of PerceptualTempoType D                          -->
<!-- ############################################################ -->
<complexType name="PerceptualTempoType">
   <complexContent>
      <extension base="mpeg7:AudioDType">
         <attribute name="score" use="required">
            <simpleType>
               <restriction base="integer">
                  <minInclusive value="1"/>
                  <maxInclusive value="255"/>
               </restriction>
            </simpleType>
         </attribute>
      </extension>
   </complexContent>
</complexType>
```

6.13.16.2 Semantics

Name	*Definition*
`PerceptualTempoType`	This type represents the perceptual tempo of a song.
`score`	Value (1 to 255) representing the speed of the perceptual tempo displayed by a song. A value of 1 indicates a slow perceptual tempo, while a value of 255 indicates a fast perceptual tempo.

6.13.16.3 Usage and Examples

The PerceptualTempoType attribute contains a single score attribute that indicates the speed of the Perceptual Tempo displayed by a song. For example:

```
<PerceptualTempoType score="187"/>
```

This represents a song with a high Percetual Tempo. Songs high in Perceptual Tempo are perceived as being fast, whereas songs low in Perceptual Tempo are perceived as being slow.

6.13.17 DistributionPerceptualAttributeType

This is a description scheme for the set of perceptual attributes of a given song that are represented by distributions. Not all distribution perceptual attributes may be available for a particular song. DistributionPerceptualAttributeType provides a convenience grouping for multiple perceptual attributes of the same type.

6.13.17.1 Syntax

```
<!-- ################################################################## -->
<!-- Definition of DistributionPerceptualAttributeType                  -->
<!-- ################################################################## -->
<complexType name="DistributionPerceptualAttributeType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="PerceptualGenreDistribution"
                 type="mpeg7:PerceptualGenreDistributionType" minOccurs="0"/>
        <element name="PerceptualMoodDistribution"
                 type="mpeg7:PerceptualMoodDistributionType" minOccurs="0"/>
        <element name="PerceptualLyricsDistribution"
                 type="mpeg7:PerceptualLyricsDistributionType" minOccurs="0"/>
        <element name="PerceptualInstrumentationDistribution"
                 type="mpeg7:PerceptualInstrumentationDistributionType"
                 minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.13.17.2 Semantics

Name	*Definition*
`DistributionPerceptualAttributeDSType`	Container for all distribution perceptual attributes in a song.
`PerceptualGenreDistribution`	Represents the Perceptual Genre distribution of a song.
`PerceptualMoodDistribution`	Represents the Perceptual Mood distribution of a song.
`PerceptualLyricsDistribution`	Represents the Perceptual Lyrics distribution of a song.
`PerceptualInstrumentationDistribution`	Represents the Perceptual Instrumentation distribution of a song.

6.13.17.3 Instantiation Requirements

In order to guarantee a proper instantiation of this description scheme, the following requirements have to be fulfilled:

- At least one *DistributionPerceptualAttributeDSType* sub-type element must be instantiated.
- All *id* attributes used within the *DistributionPerceptualAttributeDSType* sub-types must refer to terms previously defined in a classification scheme.

6.13.17.4 Usage and Examples

DistributionPerceptualAttributeDSType contains sub-elements representing the Perceptual Genre, Perceptual Mood, Perceptual Lyrics, and Perceptual Instrumentation of a song. While none of these sub-elements are required, at least one should be present in order for a DistributionPerceptualAttributeDSType instance to be useful.

The example below shows a DistributionPerceptualAttributeDSType instance for a song with the following characteristics:

- Primarily in the Rock/Pop Metal perceptual genre/sub-genre, with some Rock/Hard Rock and Rock/Pop Rock elements
- Displays a prominent Happy mood, and to a lesser degree, an Upbeat mood
- The lyrics of the song are about Love / Romance and Happiness / Celebration
- No instrumentation information is provided for this song

```
<DistributionPerceptualAttributeType>
  <PerceptualGenreDistribution>
    <PerceptualGenre
      id="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.15.11"
      score="43"/>
    <PerceptualGenre
      id="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.15.15"
      score="194"/>
    <PerceptualGenre
      id="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.15.16"
      score="18"/>
  </PerceptualGenreDistribution>
  <PerceptualMoodDistribution>
    <PerceptualMood
      id="urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.1.6"
      score="163"/>
    <PerceptualMood
      id="urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.1.13"
      score="92"/>
  </PerceptualMoodDistribution>
  <PerceptualLyricsDistribution>
    <PerceptualLyrics
      id="urn:mpeg:mpeg7:cs:LyricsPerceptualAttributeCS:2006:400.1"
      score="143"/>
    <PerceptualLyrics
      id="urn:mpeg:mpeg7:cs:LyricsPerceptualAttributeCS:2006:400.2"
      score="112"/>
  </PerceptualLyricsDistribution>
</DistributionPerceptualAttributeType>
```

6.13.18 ClassificationPerceptualAttributeType

This is a description scheme for the set of perceptual attributes of a given song that are represented solely by classifications. Not all classification perceptual attributes may be available for a particular song. ClassificationPerceptualAttributeDSType provides a convenience grouping for multiple perceptual attributes of the same type.

6.13.18.1 Syntax

```
<!-- ################################################################# -->
<!-- Definition of ClassificationPerceptualAttributeType             -->
<!-- ################################################################# -->
<complexType name="ClassificationPerceptualAttributeType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="PerceptualLanguage" type="mpeg7:PerceptualLanguageType"
                 minOccurs="0"/>
        <element name="PerceptualSound" type="mpeg7:PerceptualSoundType"
                 minOccurs="0"/>
        <element name="PerceptualRecording" type="mpeg7:PerceptualRecordingType"
                 minOccurs="0"/>
        <element name="PerceptualVocals" type="mpeg7:PerceptualVocalsType"
                 minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.13.18.2 Semantics

Name	*Definition*
`ClassificationPerceptualAttributeType`	Container for all classification perceptual attributes for a song.
`PerceptualLanguage`	Represents the Perceptual Language of the lyrics of a song.
`PerceptualSound`	Represents the Perceptual Sound of a song.
`PerceptualRecording`	Represents the Perceptual Recording type of a song.
`PerceptualVocals`	Represents the Perceptual Vocals type of a song.

6.13.18.3 Instantiation Requirements

In order to guarantee a proper instantiation of this description scheme, the following requirements have to be fulfilled:

- At least one *ClassificationPerceptualAttributeDSType* sub-type element must be instantiated.
- All *id* attributes used within the *ClassificationPerceptualAttributeDSType* sub-types must refer to terms previously defined in a classification scheme.

6.13.18.4 Usage and Examples

ClassificationPerceptualAttributeDSType contains sub-elements representing the Perceptual Language, Perceptual Sound type, Perceptual Recording type, and Perceptual Vocals type of a song. While none of these sub-elements are required, at least one should be present in order for a ClassificationPerceptualAttributeDSType instance to be useful.

The example below shows a ClassificationPerceptualAttributeDSType instance for a song with the following characteristics:

- English language lyrics
- An electronic sound
- A studio recording
- Male vocals

```
<ClassificationPerceptualAttributeType>
  <PerceptualLanguage
    id="urn:mpeg:mpeg7:cs:LanguagePerceptualAttributeCS:2006:500.1"/>
  <PerceptualSound
    id="urn:mpeg:mpeg7:cs:SoundPerceptualAttributeCS:2006:600.2"/>
  <PerceptualRecording
    id="urn:mpeg:mpeg7:cs:LanguagePerceptualAttributeCS:2006:601.1"/>
  <PerceptualVocals
    id="urn:mpeg:mpeg7:cs:VocalsPerceptualAttributeCS:2006:602.1"/>
</ClassificationPerceptualAttributeType>
```

6.13.19 LinearPerceptualAttributeDSType

This is a description scheme for the set of perceptual attributes of a given song that are represented solely by a score value. Not all linear perceptual attributes may be available for a particular song. LinearPerceptualAttributeDSType provides a convenience grouping for multiple perceptual attributes of the same type.

6.13.19.1 Syntax

```
<!-- ################################################################## -->
<!-- Definition of LinearPerceptualAttributeType                        -->
<!-- ################################################################## -->
<complexType name="LinearPerceptualAttributeType">
  <complexContent>
    <extension base="mpeg7:AudioDType">
      <sequence>
        <element name="PerceptualEnergy" type="mpeg7:PerceptualEnergyType"
                 minOccurs="0"/>
        <element name="PerceptualValence" type="mpeg7:PerceptualValenceType"
                 minOccurs="0"/>
        <element name="PerceptualBeat" type="mpeg7:PerceptualBeatType"
                 minOccurs="0"/>
        <element name="PerceptualTempo" type="mpeg7:PerceptualTempoType"
                 minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.13.19.2 Semantics

Name	*Definition*
`LinearPerceptualAttributeDSType`	Container for all linear perceptual attributes for a song.
`PerceptualEnergy`	Represents the Perceptual Energy of a song.
`PerceptualValence`	Represents the Perceptual Valence of a song.
`PerceptualBeat`	Represents the Perceptual Beat of a song.
`PerceptualTempo`	Represents the Perceptual Tempo of a song.

6.13.19.3 Instantiation Requirements

In order to guarantee a proper instantiation of this description scheme, the following requirements have to be fulfilled:

- At least one *LinearPerceptualAttributeDSType* sub-type element must be instantiated.

6.13.19.4 Usage and Examples

LinearPerceptualAttributeDSType contains sub-elements representing the Perceptual Energy, Perceptual Valence, Perceptual Beat, and Perceptual Tempo of a song. While none of these sub-elements are required, at least one should be present in order for a LinearPerceptualAttributeDSType instance to be useful.

The example below shows a LinearPerceptualAttributeDSType instance for a song with the following characteristics:

- High Perceptual Energy
- Very high Perceptual Valence (positive feelings)
- A relatively heavy Perceptual Beat
- A relatively fast Perceptual Tempo

```
<LinearPerceptionType>
  <PerceptualEnergy score="181"/>
  <PerceptualValence score="244"/>
  <PerceptualBeat score="160"/>
  <PerceptualTempo score="170"/>
</LinearPerceptionType>
```

6.13.20 PerceptualAttributeDSType

PerceptualAttributeDSType is a super-container for the three type-specific perceptual attribute containers. A PerceptualAttributeDSType instance represents the full set of perceptual attributes for a given song.

6.13.20.1 Syntax

```
<!-- ################################################################### -->
<!-- Definition of PerceptualAttributeDSType                               -->
<!-- ################################################################### -->
<complexType name="PerceptualAttributeDSType">
  <complexContent>
    <extension base="mpeg7:AudioDSType">
      <sequence>
        <element name="Distribution"
                 type="mpeg7:DistributionPerceptualAttributeType" minOccurs="0"/>
        <element name="Classification"
                 type="mpeg7:ClassificationPerceptualAttributeType"
                 minOccurs="0"/>
        <element name="LinearPerception"
                 type="mpeg7:LinearPerceptualAttributeType" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

6.13.20.2 Semantics

Name	*Definition*
`PerceptualAttributeDSType`	Container for all perceptual attributes of a song.
`Distribution`	Sub-container for the distribution-specific perceptual attributes of a song.
`Classification`	Sub-container for the classification-specific perceptual attributes of a song.
`LinearPerception`	Sub-container for the linear-specific perceptual attributes of a song.

6.13.20.3 Instantiation Requirements

In order to guarantee a proper instantiation of this description scheme, the following requirements have to be fulfilled:

- At least one *PerceptualAttributeDSType* sub-type element must be instantiated.

6.13.20.4 Usage and Examples

The following example describes the perceptual attributes of a song:

```
<AudioDescriptionScheme xsi:type="PerceptualAttributeDSType">
  <Distribution>
    <PerceptualGenreDistribution>
      <PerceptualGenre
        id="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.15.11"
        score="13"/>
      <PerceptualGenre
        id="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.15.16"
        score="6"/>
      <PerceptualGenre
        id="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.15.18"
        score="228"/>
      <PerceptualGenre
        id="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.15.23"
        score="3"/>
      <PerceptualGenre
        id="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006:100.15.27"
        score="5"/>
    </PerceptualGenreDistribution>
    <PerceptualMoodDistribution>
      <PerceptualMood
        id="urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.1.6"
        score="66"/>
      <PerceptualMood
        id="urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.1.13"
        score="92"/>
      <PerceptualMood
        id="urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.2.5"
        score="49"/>
      <PerceptualMood
        id="urn:mpeg:mpeg7:cs:MoodPerceptualAttributeCS:2006:200.2.8"
        score="48"/>
    </PerceptualMoodDistribution>
    <PerceptualLyricsDistribution>
      <PerceptualLyrics
        id="urn:mpeg:mpeg7:cs:LyricsPerceptualAttributeCS:2006:400.1"
        score="143"/>
      <PerceptualLyrics
        id="urn:mpeg:mpeg7:cs:LyricsPerceptualAttributeCS:2006:400.2"
        score="40"/>
      <PerceptualLyrics
        id="urn:mpeg:mpeg7:cs:LyricsPerceptualAttributeCS:2006:400.5"
        score="72"/>
    </PerceptualLyricsDistribution>
    <PerceptualInstrumentationDistribution>
      <PerceptualInstrumentation
        id="urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.1.2"
        score="104"/>
      <PerceptualInstrumentation
        id="urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.1.3"
        score="61"/>
      <PerceptualInstrumentation
        id="urn:mpeg:mpeg7:cs:InstrumentationPerceptualAttributeCS:2006:300.7.1"
        score="90"/>
    </PerceptualInstrumentationDistribution>
  </Distribution>
  <Classification>
    <PerceptualLanguage
      id="urn:mpeg:mpeg7:cs:LanguagePerceptualAttributeCS:2006:500.1"/>
    <PerceptualSound
      id="urn:mpeg:mpeg7:cs:SoundPerceptualAttributeCS:2006:600.2"/>
    <PerceptualRecording
      id="urn:mpeg:mpeg7:cs:LanguagePerceptualAttributeCS:2006:601.1"/>
    <PerceptualVocals
      id="urn:mpeg:mpeg7:cs:VocalsPerceptualAttributeCS:2006:602.1"/>
  </Classification>
  <LinearPerception>
    <PerceptualEnergy score="181"/>
    <PerceptualValence score="244"/>
    <PerceptualBeat score="160"/>
    <PerceptualTempo score="170"/>
  </LinearPerception>
</AudioDescriptorScheme>
```

Add the following table to Annex B:

Patent statements for ISO/IEC 15938-4:2002/Amd.2

ISO/IEC 15938-4:2002/Amd.2 **Patent Statements**
Fraunhofer
Moodlogic

Add a new Annex after Annex C:

Annex D
(normative)

Chord tables and additional Classification Schemes used in ISO/IEC 15938-4:Audio

D.1 Numbering of tones belonging to a certain type of triad

Table D.1 – This table shows the design of chords at different chord modes.

number of half tones from the base tone	0	1	2	3	4	5	6	7	8	9	10	11
sample notes	c	c#	d	d#	e	f	f#	g	g#	a	a#	b
major	1				1			1				
minor	1			1				1				
augmented	1				1				1			
diminished	1			1			1					
suspended 2	1		1					1				
suspended 4	1					1		1				

D.2 Numbering of tones belonging to a certain type of 7th chord.

Table D.2 – This table shows additional notes (marked with 1) on 7th chords.

Major7												1
Dominant 7											1	

D.3 Example Classification Scheme for Genre Perceptual Attribute CS

```
<ClassificationScheme
   uri="urn:mpeg:mpeg7:cs:GenrePerceptualAttributeCS:2006"
   domain="//PerceptualAttributeDS/Distribution/PerceptualGenreDistribution/Perc
eptualGenre">

   <Header xsi:type="DescriptionMetadataType">
      <Comment>
         <FreeTextAnnotation xml:lang="en">
            The terms in this classification scheme (perceptual genres)
            describe the general style or styles of a piece of music.
            A particular piece of music is understood to have potential
            degrees of membership in more than one genre or style
            of music. For example, a particular song may be perceived
            to be predominantly a Rock song with elements of Country.
            Perceptual genres are useful for finding songs which embody
            the same or similar styles of music.
         </FreeTextAnnotation>
      </Comment>
   </Header>

   <Term termId="100">
      <Name xml:lang="en">PerceptualGenre</Name>
      <Term termId="100.1">
         <Name xml:lang="en">Alternative</Name>
         <Term termId="100.1.1">
            <Name xml:lang="en">Alt Dance</Name>
         </Term>
         <Term termId="100.1.2">
            <Name xml:lang="en">Alt Power Pop</Name>
         </Term>
         <Term termId="100.1.3">
            <Name xml:lang="en">AlternaPop</Name>
         </Term>
         <Term termId="100.1.4">
            <Name xml:lang="en">BritPop</Name>
         </Term>
         <Term termId="100.1.5">
            <Name xml:lang="en">Chamber Pop</Name>
         </Term>
         <Term termId="100.1.6">
            <Name xml:lang="en">DreamPop</Name>
         </Term>
         <Term termId="100.1.7">
            <Name xml:lang="en">Experimental</Name>
         </Term>
         <Term termId="100.1.8">
```

```
            <Name xml:lang="en">Goth</Name>
        </Term>
        <Term termId="100.1.9">
            <Name xml:lang="en">Grunge/Post Grunge</Name>
        </Term>
        <Term termId="100.1.10">
            <Name xml:lang="en">Indie</Name>
        </Term>
        <Term termId="100.1.11">
            <Name xml:lang="en">Industrial</Name>
        </Term>
        <Term termId="100.1.12">
            <Name xml:lang="en">Industrial Dance</Name>
        </Term>
        <Term termId="100.1.13">
            <Name xml:lang="en">Industrial Metal</Name>
        </Term>
        <Term termId="100.1.14">
            <Name xml:lang="en">Latin Alternative</Name>
        </Term>
        <Term termId="100.1.15">
            <Name xml:lang="en">Neo Psychedelic</Name>
        </Term>
        <Term termId="100.1.16">
            <Name xml:lang="en">Neo-Swing</Name>
        </Term>
        <Term termId="100.1.17">
            <Name xml:lang="en">New Romantic</Name>
        </Term>
        <Term termId="100.1.18">
            <Name xml:lang="en">New Wave</Name>
        </Term>
        <Term termId="100.1.19">
            <Name xml:lang="en">New Wave Ska</Name>
        </Term>
        <Term termId="100.1.20">
            <Name xml:lang="en">Punk Revival</Name>
        </Term>
        <Term termId="100.1.21">
            <Name xml:lang="en">Punk/Pre-Punk</Name>
        </Term>
        <Term termId="100.1.22">
            <Name xml:lang="en">Rap/Funk/Metal</Name>
        </Term>
        <Term termId="100.1.23">
            <Name xml:lang="en">Riot Grrrl</Name>
        </Term>
        <Term termId="100.1.24">
            <Name xml:lang="en">Ska Revival</Name>
        </Term>
        <Term termId="100.1.25">
            <Name xml:lang="en">Synth Pop</Name>
        </Term>
        <Term termId="100.1.26">
            <Name xml:lang="en">Urban Folk</Name>
        </Term>
    </Term>
    <Term termId="100.2">
        <Name xml:lang="en">Blues</Name>
        <Term termId="100.2.1">
```

```
            <Name xml:lang="en">Classic</Name>
         </Term>
         <Term termId="100.2.2">
            <Name xml:lang="en">Early Acoustic</Name>
         </Term>
         <Term termId="100.2.3">
            <Name xml:lang="en">Jump Blues</Name>
         </Term>
         <Term termId="100.2.4">
            <Name xml:lang="en">Modern Acoustic</Name>
         </Term>
         <Term termId="100.2.5">
            <Name xml:lang="en">Modern Electric</Name>
         </Term>
      </Term>
      <Term termId="100.3">
         <Name xml:lang="en">Country</Name>
         <Term termId="100.3.1">
            <Name xml:lang="en">Alt Country Rock</Name>
         </Term>
         <Term termId="100.3.2">
            <Name xml:lang="en">Americana</Name>
         </Term>
         <Term termId="100.3.3">
            <Name xml:lang="en">Bluegrass</Name>
         </Term>
         <Term termId="100.3.4">
            <Name xml:lang="en">Country Pop</Name>
         </Term>
         <Term termId="100.3.5">
            <Name xml:lang="en">Cowboy</Name>
         </Term>
         <Term termId="100.3.6">
            <Name xml:lang="en">Easy Country</Name>
         </Term>
         <Term termId="100.3.7">
            <Name xml:lang="en">Honky Tonk</Name>
         </Term>
         <Term termId="100.3.8">
            <Name xml:lang="en">Nashville</Name>
         </Term>
         <Term termId="100.3.9">
            <Name xml:lang="en">New Traditional</Name>
         </Term>
         <Term termId="100.3.10">
            <Name xml:lang="en">Outlaw</Name>
         </Term>
         <Term termId="100.3.11">
            <Name xml:lang="en">Traditional</Name>
         </Term>
         <Term termId="100.3.12">
            <Name xml:lang="en">Western Swing</Name>
         </Term>
      </Term>
      <Term termId="100.4">
         <Name xml:lang="en">Easy Listening</Name>
         <Term termId="100.4.1">
            <Name xml:lang="en">Cabaret</Name>
         </Term>
         <Term termId="100.4.2">
```

```
            <Name xml:lang="en">Classical Crossover</Name>
         </Term>
         <Term termId="100.4.3">
            <Name xml:lang="en">Orchestral</Name>
         </Term>
         <Term termId="100.4.4">
            <Name xml:lang="en">Standards</Name>
         </Term>
      </Term>
      <Term termId="100.5">
         <Name xml:lang="en">Electronica</Name>
         <Term termId="100.5.1">
            <Name xml:lang="en">Acid House</Name>
         </Term>
         <Term termId="100.5.2">
            <Name xml:lang="en">Ambient</Name>
         </Term>
         <Term termId="100.5.3">
            <Name xml:lang="en">Beats &amp; Breaks</Name>
         </Term>
         <Term termId="100.5.4">
            <Name xml:lang="en">Detroit</Name>
         </Term>
         <Term termId="100.5.5">
            <Name xml:lang="en">Digital Hardcore</Name>
         </Term>
         <Term termId="100.5.6">
            <Name xml:lang="en">Drum &amp; Bass</Name>
         </Term>
         <Term termId="100.5.7">
            <Name xml:lang="en">ElectroFunk</Name>
         </Term>
         <Term termId="100.5.8">
            <Name xml:lang="en">ElectroTechno</Name>
         </Term>
         <Term termId="100.5.9">
            <Name xml:lang="en">Experimental Techno</Name>
         </Term>
         <Term termId="100.5.10">
            <Name xml:lang="en">House</Name>
         </Term>
         <Term termId="100.5.11">
            <Name xml:lang="en">Jazz-House</Name>
         </Term>
         <Term termId="100.5.12">
            <Name xml:lang="en">Krautrock</Name>
         </Term>
         <Term termId="100.5.13">
            <Name xml:lang="en">Techno</Name>
         </Term>
         <Term termId="100.5.14">
            <Name xml:lang="en">Trance</Name>
         </Term>
         <Term termId="100.5.15">
            <Name xml:lang="en">Triphop</Name>
         </Term>
         <Term termId="100.5.16">
            <Name xml:lang="en">UK Garage / 2-Step</Name>
         </Term>
      </Term>
      <Term termId="100.6">
         <Name xml:lang="en">Folk</Name>
         <Term termId="100.6.1">
            <Name xml:lang="en">British/Celtic</Name>
         </Term>
         <Term termId="100.6.2">
            <Name xml:lang="en">Cajun/Zydeco</Name>
         </Term>
         <Term termId="100.6.3">
            <Name xml:lang="en">Newgrass</Name>
         </Term>
         <Term termId="100.6.4">
            <Name xml:lang="en">Singer/Songwriter</Name>
         </Term>
         <Term termId="100.6.5">
            <Name xml:lang="en">Traditional</Name>
         </Term>
      </Term>
      <Term termId="100.7">
         <Name xml:lang="en">Gospel/Christian</Name>
         <Term termId="100.7.1">
            <Name xml:lang="en">Contemporary</Name>
         </Term>
         <Term termId="100.7.2">
            <Name xml:lang="en">Traditional</Name>
         </Term>
      </Term>
      <Term termId="100.8">
         <Name xml:lang="en">Jazz</Name>
         <Term termId="100.8.1">
            <Name xml:lang="en">BeBop</Name>
         </Term>
         <Term termId="100.8.2">
            <Name xml:lang="en">Classic</Name>
         </Term>
         <Term termId="100.8.3">
            <Name xml:lang="en">Cool</Name>
         </Term>
         <Term termId="100.8.4">
            <Name xml:lang="en">Free/Avant Garde</Name>
         </Term>
         <Term termId="100.8.5">
            <Name xml:lang="en">Fusion</Name>
         </Term>
         <Term termId="100.8.6">
            <Name xml:lang="en">Hard Bop</Name>
         </Term>
         <Term termId="100.8.7">
            <Name xml:lang="en">Jazz Vocal</Name>
         </Term>
         <Term termId="100.8.8">
            <Name xml:lang="en">Latin</Name>
         </Term>
         <Term termId="100.8.9">
            <Name xml:lang="en">Post Bop</Name>
         </Term>
         <Term termId="100.8.10">
            <Name xml:lang="en">Small Group Swing</Name>
         </Term>
         <Term termId="100.8.11">
```

```
            <Name xml:lang="en">Smooth</Name>
        </Term>
        <Term termId="100.8.12">
            <Name xml:lang="en">Soul/Groove</Name>
        </Term>
        <Term termId="100.8.13">
            <Name xml:lang="en">Swing/Big Band</Name>
        </Term>
    </Term>
    <Term termId="100.9">
        <Name xml:lang="en">Latin</Name>
        <Term termId="100.9.1">
            <Name xml:lang="en">Afro Cuban Heritage</Name>
        </Term>
        <Term termId="100.9.2">
            <Name xml:lang="en">Argentina-Tango</Name>
        </Term>
        <Term termId="100.9.3">
            <Name xml:lang="en">Brazilian</Name>
        </Term>
        <Term termId="100.9.4">
            <Name xml:lang="en">Conjunto</Name>
        </Term>
        <Term termId="100.9.5">
            <Name xml:lang="en">Latin Folk</Name>
        </Term>
        <Term termId="100.9.6">
            <Name xml:lang="en">Latin Pop</Name>
        </Term>
        <Term termId="100.9.7">
            <Name xml:lang="en">Norteno</Name>
        </Term>
        <Term termId="100.9.8">
            <Name xml:lang="en">Ranchera</Name>
        </Term>
        <Term termId="100.9.9">
            <Name xml:lang="en">Salsa</Name>
        </Term>
        <Term termId="100.9.10">
            <Name xml:lang="en">Tejano</Name>
        </Term>
    </Term>
    <Term termId="100.10">
        <Name xml:lang="en">New Age</Name>
        <Term termId="100.10.1">
            <Name xml:lang="en">Avant Garde/Minimal</Name>
        </Term>
        <Term termId="100.10.2">
            <Name xml:lang="en">Ethnic Fusion</Name>
        </Term>
        <Term termId="100.10.3">
            <Name xml:lang="en">Nature/Ethnic</Name>
        </Term>
        <Term termId="100.10.4">
            <Name xml:lang="en">Neo-Classical</Name>
        </Term>
        <Term termId="100.10.5">
            <Name xml:lang="en">Worldbeat Fusion</Name>
        </Term>
    </Term>
```

```
       <Term termId="100.11">
          <Name xml:lang="en">Pop/Dance</Name>
          <Term termId="100.11.1">
             <Name xml:lang="en">Acid-Jazz</Name>
          </Term>
          <Term termId="100.11.2">
             <Name xml:lang="en">Disco</Name>
          </Term>
          <Term termId="100.11.3">
             <Name xml:lang="en">Eurodance</Name>
          </Term>
          <Term termId="100.11.4">
             <Name xml:lang="en">Mainstream</Name>
          </Term>
          <Term termId="100.11.5">
             <Name xml:lang="en">Teen Pop</Name>
          </Term>
       </Term>
       <Term termId="100.12">
          <Name xml:lang="en">R&amp;B/Soul</Name>
          <Term termId="100.12.1">
             <Name xml:lang="en">Classic Soul</Name>
          </Term>
          <Term termId="100.12.2">
             <Name xml:lang="en">Contemporary</Name>
          </Term>
          <Term termId="100.12.3">
             <Name xml:lang="en">Doo-Wop</Name>
          </Term>
          <Term termId="100.12.4">
             <Name xml:lang="en">Funk</Name>
          </Term>
          <Term termId="100.12.5">
             <Name xml:lang="en">Motown</Name>
          </Term>
          <Term termId="100.12.6">
             <Name xml:lang="en">Neo-Soul</Name>
          </Term>
          <Term termId="100.12.7">
             <Name xml:lang="en">New Jack Swing</Name>
          </Term>
          <Term termId="100.12.8">
             <Name xml:lang="en">Philly Soul</Name>
          </Term>
          <Term termId="100.12.9">
             <Name xml:lang="en">Quiet Storm</Name>
          </Term>
       </Term>
       <Term termId="100.13">
          <Name xml:lang="en">Rap/Hip-Hop</Name>
          <Term termId="100.13.1">
             <Name xml:lang="en">Alternative</Name>
          </Term>
          <Term termId="100.13.2">
             <Name xml:lang="en">Gangsta/G-Funk</Name>
          </Term>
          <Term termId="100.13.3">
             <Name xml:lang="en">Hardcore</Name>
          </Term>
          <Term termId="100.13.4">
```

```
            <Name xml:lang="en">Old School</Name>
        </Term>
        <Term termId="100.13.5">
            <Name xml:lang="en">Pop Rap</Name>
        </Term>
        <Term termId="100.13.6">
            <Name xml:lang="en">Turntablist</Name>
        </Term>
    </Term>
    <Term termId="100.14">
        <Name xml:lang="en">Reggae</Name>
        <Term termId="100.14.1">
            <Name xml:lang="en">Dancehall &amp; Ragga</Name>
        </Term>
        <Term termId="100.14.2">
            <Name xml:lang="en">Dub &amp; DJ</Name>
        </Term>
        <Term termId="100.14.3">
            <Name xml:lang="en">Lovers Rock</Name>
        </Term>
        <Term termId="100.14.4">
            <Name xml:lang="en">Roots Reggae</Name>
        </Term>
        <Term termId="100.14.5">
            <Name xml:lang="en">Ska &amp; Rocksteady</Name>
        </Term>
    </Term>
    <Term termId="100.15">
        <Name xml:lang="en">Rock</Name>
        <Term termId="100.15.1">
            <Name xml:lang="en">Adult Alternative</Name>
        </Term>
        <Term termId="100.15.2">
            <Name xml:lang="en">Adult Contemporary</Name>
        </Term>
        <Term termId="100.15.3">
            <Name xml:lang="en">AM Radio Rock</Name>
        </Term>
        <Term termId="100.15.4">
            <Name xml:lang="en">Blues Rock</Name>
        </Term>
        <Term termId="100.15.5">
            <Name xml:lang="en">British Pop Invasion</Name>
        </Term>
        <Term termId="100.15.6">
            <Name xml:lang="en">British Rock Invasion</Name>
        </Term>
        <Term termId="100.15.7">
            <Name xml:lang="en">Classic Teen Idols</Name>
        </Term>
        <Term termId="100.15.8">
            <Name xml:lang="en">Country Rock</Name>
        </Term>
        <Term termId="100.15.9">
            <Name xml:lang="en">Folk Rock</Name>
        </Term>
        <Term termId="100.15.10">
            <Name xml:lang="en">Girl Groups</Name>
        </Term>
        <Term termId="100.15.11">
```