

Recommendation ITU-R BS.1352-4 (05/2023)

BS Series: Broadcasting service (sound)

**File format for the exchange of audio
programme materials with metadata on
information technology media**

Foreword

The role of the Radiocommunication Sector is to ensure the rational, equitable, efficient and economical use of the radio-frequency spectrum by all radiocommunication services, including satellite services, and carry out studies without limit of frequency range on the basis of which Recommendations are adopted.

The regulatory and policy functions of the Radiocommunication Sector are performed by World and Regional Radiocommunication Conferences and Radiocommunication Assemblies supported by Study Groups.

Policy on Intellectual Property Right (IPR)

ITU-R policy on IPR is described in the Common Patent Policy for ITU-T/ITU-R/ISO/IEC referenced in Resolution ITU-R 1. Forms to be used for the submission of patent statements and licensing declarations by patent holders are available from <http://www.itu.int/ITU-R/go/patents/en> where the Guidelines for Implementation of the Common Patent Policy for ITU-T/ITU-R/ISO/IEC and the ITU-R patent information database can also be found.

Series of ITU-R Recommendations

(Also available online at <https://www.itu.int/publ/R-REC/en>)

Series	Title
BO	Satellite delivery
BR	Recording for production, archival and play-out; film for television
BS	Broadcasting service (sound)
BT	Broadcasting service (television)
F	Fixed service
M	Mobile, radiodetermination, amateur and related satellite services
P	Radiowave propagation
RA	Radio astronomy
RS	Remote sensing systems
S	Fixed-satellite service
SA	Space applications and meteorology
SF	Frequency sharing and coordination between fixed-satellite and fixed service systems
SM	Spectrum management
SNG	Satellite news gathering
TF	Time signals and frequency standards emissions
V	Vocabulary and related subjects

***Note:** This ITU-R Recommendation was approved in English under the procedure detailed in Resolution ITU-R 1.*

Electronic Publication
Geneva, 2023

© ITU 2023

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without written permission of ITU.

RECOMMENDATION ITU-R BS.1352-4

**File format for the exchange of audio programme materials
with metadata on information technology media**

(Question ITU-R 34-3/6)

(1998-2001-2002-2007-2023)

Scope

This Recommendation contains the specification of the broadcast audio extension chunk¹ and its use with PCM-coded, and MPEG-1 or MPEG-2 audio data that is smaller than 4 gigabytes in size. Basic information on the RIFF format and how it can be extended to other types of audio data is also included.

Keywords

File, file format, wave, WAV, RIFF, BWF, broadcast wave format file

The ITU Radiocommunication Assembly,

considering

- a) that storage media based on Information Technology, including data disks and tapes, have penetrated all areas of audio production for radio broadcasting, namely non-linear editing, on-air play-out and archives;
- b) that this technology offers significant advantages in terms of operating flexibility, production flow and station automation and it is therefore attractive for the up-grading of existing studios and the design of new studio installations;
- c) that the adoption of a single file format for signal interchange would greatly simplify the interoperability of individual equipment and remote studios, it would facilitate the desirable integration of editing, on-air play-out and archiving;
- d) that a minimum set of broadcast related information must be included in the file to document the metadata related to the audio signal;
- e) that, to ensure the compatibility between applications with different complexity, a minimum set of functions, common to all the applications able to handle the recommended file format must be agreed;
- f) that Recommendation ITU-R BS.646 defines the digital audio format used in audio production for radio and television broadcasting;
- g) that the need for exchanging audio materials also arises when ISO/IEC 11172-3 and ISO/IEC 13818-3 coding systems are used to compress the signal;
- h) that the compatibility with currently available commercial file formats could minimize the industry efforts required to implement this format in the equipment;
- j) that a standard format for the coding history information would simplify the use of the information after programme exchange;
- k) that the quality of an audio signal is influenced by signal processing experienced by the signal, particularly by the use of non-linear coding and decoding during bit-rate reduction processes,

¹ A chunk is the basic building block of a file in the Microsoft ® Resource Interchange File Format (RIFF).

recognizing

that the file-format specified in Recommendation ITU-R BS.2088 can convey data of up to 16 exabytes and provide support for Recommendation ITU-R BS.2076 metadata,

recommends

- 1 that, for the exchange of audio programmes on Information Technology media, the audio signal parameters, sampling frequency, coding resolution and pre-emphasis should be set in agreement with the relevant parts of Recommendation ITU-R BS.646;
- 2 that the file format specified in Annex 1 should be used for the interchange of audio programmes in linear pulse code modulation (PCM) format on Information Technology media;
- 3 that, when the audio signals are coded using ISO/IEC 11172-3 or ISO/IEC 13818-3 coding systems, the file format specified in Annex 1 and complemented with Annex 2 should be used for the interchange of audio programmes on Information Technology media²;
- 4 that, when the file format specified in Annexes 1 and/or 2 is used to carry information on the audio material gathered and computed by a capturing workstation (Digital Audio Workstation (DAW)), the metadata should conform to the specifications detailed in Annex 3.

Annex 1

Specification of the broadcast wave format

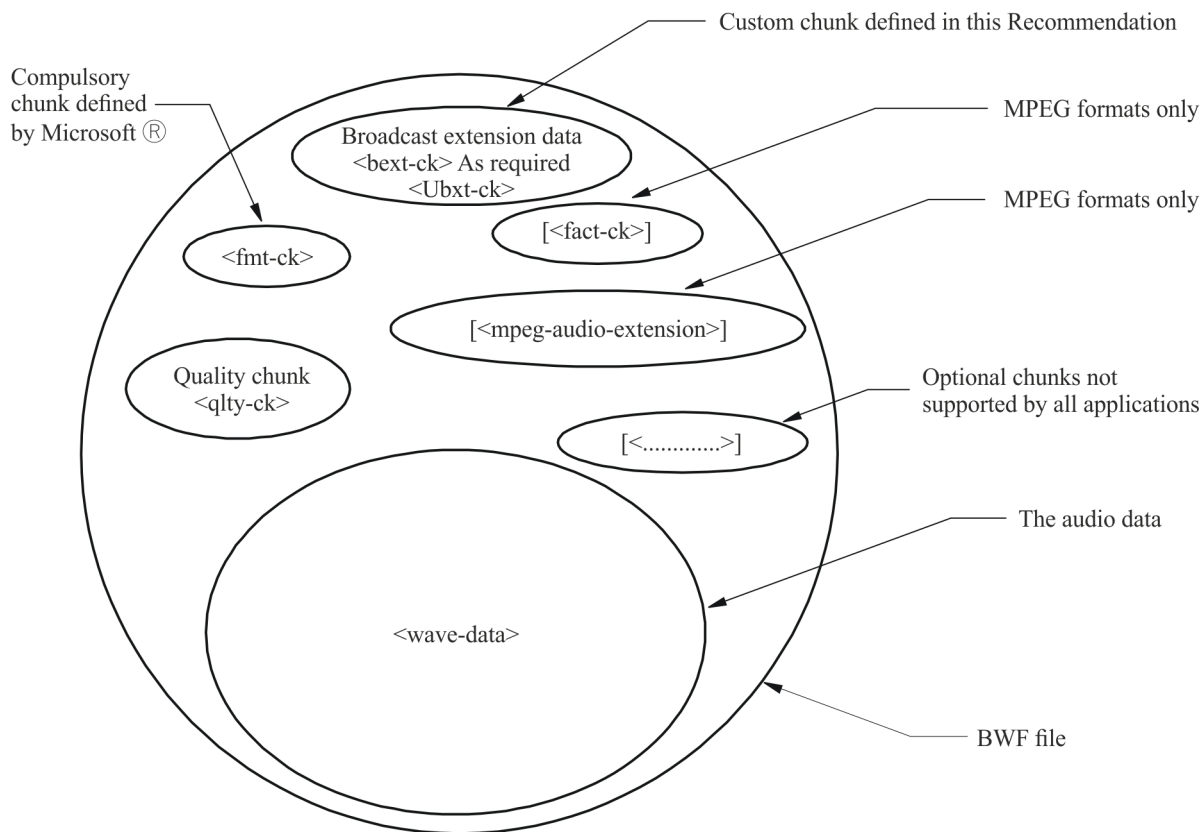
A format for audio data files in broadcasting

1 Introduction

The Broadcast Wave Format (BWF) is based on the Microsoft® WAVE audio file format which is a type of file specified in the Microsoft® “Resource Interchange File Format”, RIFF. WAVE files specifically contain audio data. The basic building block of the RIFF file format, called a chunk, contains a group of tightly related pieces of information. It consists of a chunk identifier, an integer value representing the length in bytes and the information carried. A RIFF file is made up of a collection of chunks.

For the BWF, some restrictions are applied to the original WAVE format. In addition the BWF file includes a <Broadcast Audio Extension> chunk. This is illustrated in Fig. 1.

² It is recognized that a recommendation in that sense could penalize developers using some computer platforms.

FIGURE 1
BWF file

BS.1352-01

This Annex contains the specification of the broadcast audio extension chunk that is used in all BWF files. In addition, information on the basic RIFF format and how it can be extended to other types of audio data is given in Attachment 1. Details of the PCM wave format are also given in Attachment 1. Detailed specifications of the extension to other types of audio data, and metadata are included in Annexes 2 and 3.

1.1 Normative provisions

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met.

The words “shall” or some other obligatory language such as “must” and the negative equivalents are used to express those mandatory provisions. The use of such words does not suggest that compliance with the Recommendation is required of any party.

2 Broadcast wave format (BWF) file

2.1 Contents of a broadcast wave format file

A broadcast wave format file shall start with the mandatory Microsoft® RIFF “WAVE” header and at least the following chunks:

<WAVE-form>	
RIFF('WAVE'	
<fmt-ck>	/*Format of the audio signal:PCM/MPEG */
<broadcast_audio_extension>	/*information on the audio sequence */
<universal broadcast audio extension>	/* ubxt is required for multi-byte language support only*/
<fact-ck>	/* Fact chunk is required for MPEG formats only*/
<mpeg_audio_extension>	/* MPEG Audio Extension chunk is required for MPEG formats only*/
<wave-data>)	/*sound data */
<quality-chunk>	/* only required when information concerning relevant events impacting quality is needed*/

NOTE – Additional chunks may be present in the file. Some of these may be outside the scope of this Recommendation. Applications may or may not interpret or make use of these chunks, so the integrity of the data contained in such unknown chunks cannot be guaranteed. However, compliant applications shall pass on unknown chunks transparently.

2.2 Existing chunks defined as part of the RIFF standard

The RIFF standard is defined in documents issued by the Microsoft^{®3} Corporation. This application uses a number of chunks that are already defined. These are:

fmt-ck

fact-ck

The current descriptions of these chunks are given for information in Attachment 1 to Annex 1.

2.3 Broadcast audio extension chunk⁴

Extra parameters needed for exchange of material between broadcasters are added in a specific “Broadcast Audio Extension” chunk defined as follows:

```

broadcast_audio_extension typedef struct {
    DWORD ckID,                /* (broadcastextension)ckID=bext. */
    DWORD ckSize,              /* size of extension chunk */
    BYTE  ckData[ckSize],      /* data of the chunk */
}

typedef struct broadcast_audio_extension {
    CHAR Description[256],      /* ASCII: "Description of the sound sequence"*/
    CHAR Originator[32],        /* ASCII: "Name of the originator"*/
    CHAR OriginatorReference[32], /* ASCII: "Reference of the originator"*/
    CHAR OriginationDate[10],    /* ASCII: "yyyy:mm:dd" */
    CHAR OriginationTime[8],     /* ASCII: "hh:mm:ss" */
    DWORD TimeReferenceLow,      /* First sample count since midnight, low word*/
    DWORD TimeReferenceHigh,     /* First sample count since midnight, high word */
}

```

³ Microsoft Resource Interchange File Format, RIFF, available (2005-12) at http://www.tactilemedia.com/info/MCI_Control_Info.html.

⁴ See § 2.4 for ubxt chunk definition, to express the human-readable information of the bext chunk in a multi-byte character set.

WORD Version,	/* Version of the BWF; unsigned binary number */
BYTE UMID_0,	/* Binary byte 0 of SMPTE UMID */....
BYTE UMID_63,	/* Binary byte 63 of SMPTE UMID */
CHAR Reserved[190],	/* 190 bytes, reserved for future use, set to .NULL. * /
CHAR CodingHistory[],	/* ASCII: "History coding" */
} BROADCAST_EXT,	

Field	Description
Description	ASCII string (maximum 256 characters) containing a free description of the sequence. To help applications which only display a short description it is recommended that a resume of the description is contained in the first 64 characters and the last 192 characters are used for details. If the length of the string is less than 256 characters the last one is followed by a null character. (0x00)
Originator	ASCII string (maximum 32 characters) containing the name of the originator/producer of the audio file. If the length of the string is less than 32 characters the field is ended by a null character. (0x00)
OriginatorReference	ASCII string (maximum 32 characters) containing a non ambiguous reference allocated by the originating organization. If the length of the string is less than 32 characters the field is ended a null character. (0x00) A standard format for the "Unique" Source Identifier (USID) information for use in the OriginatorReference field is given in Attachment 3 to Annex 1.
OriginationDate	10 ASCII characters containing the date of creation of the audio sequence. The format is «',year',-',month',-',day',» with 4 characters for the year and 2 characters per other item. Year is defined from 0000 to 9999 Month is defined from 1 to 12 Day is defined from 1 to 31 The separator between the items should be a hyphen in compliance with ISO 8601. Some legacy implementations may use '_' underscore, ':' colon, ' ' space, '.' Stop, reproducing equipment should recognize these separator characters
OriginationTime	8 ASCII characters containing the time of creation of the audio sequence. The format is «',hour',-',minute',-',second'» with 2 characters per item. Hour is defined from 0 to 23. Minute and second are defined from 0 to 59.

The separator between the items should be a hyphen in compliance with ISO 8601 . Some legacy implementations may use ‘_’ underscore, ‘:’ colon, ‘ ’ space, ‘.’ Stop, reproducing equipment should recognize these separator characters.

TimeReference	This field contains the time-code of the sequence. It is a 64-bit value which contains the first sample count since midnight. The number of samples per second depends on the sample frequency that is defined in the field <nSamplesPerSec> from the <fmt-ck>.
Version	An unsigned binary number giving the version of the BWF. For Version 1, this is set to 0x0001.
UMID	64 bytes containing an extended UMID defined by SMPTE 330M. If a 32-byte basic UMID is used, the last 32 bytes should be filled with zeros. If no UMID is available, the 64 bytes should be filled with zeros. NOTE – The length of the UMID is coded at the head of the UMID itself.
Reserved	190 bytes reserved for extension. These 190 bytes should be set to zero.
Coding History	A variable-size block of ASCII characters comprising 0 or more strings each terminated by <CR><LF>The first unused character should be a null character (0x00). Each string should contain a description of a coding process applied to the audio data. Each new coding application should add a new string with the appropriate info. A standard format for the coding history information is given in Attachment 2 to Annex 1. This information must contain the type of sound (PCM or MPEG) with its specific parameters: PCM: mode (mono, stereo), size of the sample (8, 16 bits) and sample frequency, MPEG: sampling frequency, bit rate, Layer (I or II) and the mode (mono, stereo, joint stereo or dual channel), It is recommended that the manufacturers of the coders provide an ASCII string for use in the coding history.

2.4 Universal broadcast audio extension chunk

The information contained in the Broadcast Audio Extension (bext) chunk defined in § 2.3 may additionally be carried by a dedicated chunk called “Universal Broadcast Audio Extension”, or “ubxt” chunk to express the human-readable information of the bext chunk in multi-byte languages. The basic structure of this metadata chunk is the same as that of the bext chunk. Four human-readable items, uDescription, uOriginator, uOriginatorReference and uCodingHistory, are described in UTF-8 (8-bit UCS Transformation Format) instead of ASCII. The first three items have 8 times the data size of the corresponding items in the bext chunk. The structure of the ubxt chunk is defined as follows:

```
typedef struct chunk_header {
    DWORD   ckID;           /* (universal broadcast extension)ckID=ubxt */
    DWORD   ckSize;         /* size of extension chunk */
    BYTE    ckData[ckSize]; /* data of the chunk */
}
```

```
} CHUNK_HEADER;
```

```
typedef struct universal_broadcast_audio_extension {
```

```
    BYTE    uDescription[256*8];        /* UTF-8 : "Description of the sound sequence" */
    BYTE    uOriginator[32*8];          /* UTF-8 : "Name of the originator" */
    BYTE    uOriginatorReference[32*8]; /* UTF-8 : "Reference of the originator" */
    CHAR    OriginationDate[10];        /* ASCII : "yyyy:mm:dd" */
    CHAR    OriginationTime[8];         /* ASCII : "hh:mm:ss" */
    DWORD    TimeReferenceLow;          /* First sample count since midnight, low word */
    DWORD    TimeReferenceHigh;         /* First sample count since midnight, high word */
    WORD     Version;                   /* Version of the BWF; unsigned binary number */
    BYTE     UMID_0;                    /* Binary byte 0 of SMPTE UMID */
    BYTE     UMID_63;                   /* Binary byte 63 of SMPTE UMID */
    CHAR     Reserved[190];             /* 190 bytes, reserved for future use, set to
                                         "NULL" */
    BYTE     uCodingHistory[];          /* UTF-8 : "Coding history" */
```

```
} UNIV_BROADCAST_EXT;
```

Field	Description
uDescription	UTF-8 string, 2 048 bytes or less, containing a description of the sequence. If data is not available or if the length of the string is less than 2 048 bytes, the first unused byte shall be a null character (0x00).
uOriginator	UTF-8 string, 256 bytes or less, containing the name of the originator of the audio file. If data is not available or if the length of the string is less than 256 bytes, the first unused byte shall be a null character (0x00).
uOriginatorReference	UTF-8 string, 256 bytes or less, containing a reference allocated by the originating organization. If data is not available or if the length of the string is less than 256 bytes, the first unused byte shall be a null character (0x00).
OriginationDate	10 ASCII characters containing the date of creation of the audio sequence. The format is « 'year', '-', 'month', '-', 'day', '» with 4 characters for the year and 2 characters per other item. Year is defined from 0000 to 9999 Month is defined from 1 to 12 Day is defined from 1 to 31 The separator between the items should be a hyphen in compliance with ISO 8601. Some legacy implementations may use ' _ ' underscore, ' : ' colon, ' ' space, ' . ' Stop; reproducing equipment should recognize these separator characters.
OriginationTime	8 ASCII characters containing the time of creation of the audio sequence. The format is « 'hour', '-', 'minute', '-', 'second' » with 2 characters per item. Hour is defined from 0 to 23. Minute and second are defined from 0 to 59.

The separator between the items should be a hyphen in compliance with ISO 8601 . Some legacy implementations may use ‘_’ underscore, ‘:’ colon; ‘ ’ space; ‘.’ Stop; reproducing equipment should recognize these separator characters.

TimeReference	This field contains the time-code of the sequence. It is a 64-bit value which contains the first sample count since midnight. The number of samples per second depends on the sample frequency that is defined in the field <nSamplesPerSec> from the <fmt-ck>.
Version	An unsigned binary number giving the version of the BWF. For Version 1, this is set to 0x0001.
UMID	64 bytes containing an extended UMID defined by SMPTE 330M. If a 32-byte basic UMID is used, the last 32 bytes should be filled with zeros. If no UMID is available, the 64 bytes should be filled with zeros. NOTE – The length of the UMID is coded at the head of the UMID itself.
Reserved	190 bytes reserved for extension. These 190 bytes should be set to zero.
uCoding History	A variable-size block of UTF-8 characters comprising 0 or more strings each terminated by <CR><LF>. The first unused byte shall be a null character (0x00). Each string shall contain a description of a coding process applied to the audio data. Each new coding application should add a new string with the appropriate information. A standard format for the coding history information is given in Attachment 2 to Annex 1. This information shall contain the type of sound (PCM or MPEG) with its specific parameters: PCM: mode (mono, stereo), size of the sample (8, 16 bits) and sample frequency, MPEG: sampling frequency, bit rate, Layer (I or II) and the mode (mono, stereo, joint stereo or dual channel),

NOTE 1 – All the items except uDescription, uOriginator, uOriginatorReference and uCodingHistory shall have the same content as that of each corresponding item of the bext chunk. § 2.3.

NOTE 2 – When a given code value in UTF-8 is out of the subset (as defined in Chapter 12 of ISO/IEC 10646:2003) supported by a piece of processing equipment, the value shall be unchanged and ignored for processing.

Attachment 1 to Annex 1 (Informative)

RIFF WAVE (.WAV) file format

The information in this Attachment is taken from the specification documents of Microsoft® RIFF file format. It is included for information only.

1 Waveform audio file format (WAVE)

The WAVE form is defined as follows. Programs must expect (and ignore) any unknown chunks encountered, as with all RIFF forms. However, <fmt-ck> must always occur before <wave-data>, and both of these chunks are mandatory in a WAVE file.

```
<WAVE-form> ->
    RIFF ('WAVE'
        <fmt-ck>           // Format chunk
        [<fact-ck>]        // Fact chunk
        [<other-ck>]       // Other optional chunks
        <wave-data>)       // Sound data
```

The WAVE chunks are described in the following sections:

1.1 WAVE format chunk

The WAVE format chunk <fmt-ck> specifies the format of the <wave-data>. The <fmt-ck> is defined as follows:

```
<fmt-ck> ->fmt( <common-fields>
                <format-specific-fields> )
<common-fields> ->
    struct{
        WORD wFormatTag,           /* Format category */
        WORD nChannels,           /* Number of channels */
        DWORD nSamplesPerSec,     /* Sampling rate */
        DWORD nAvgBytesPerSec,    /* For buffer estimation*/
        WORD nBlockAlign,         /* Data block size*/
    }
```

The fields in the <common-fields> portion of the chunk are as follows:

Field	Description
wFormatTag	A number indicating the WAVE format category of the file. The content of the <format-specific-fields> portion of the <fmt-ck> and the interpretation of the waveform data, depend on this value.
nchannels	The number of channels represented in the waveform data, such as 1 for mono or 2 for stereo.
nSamplesPerSec	The sampling rate (in samples per second) at which each channel should be reproduced.
nAvgBytesPerSec	The average number of bytes per second at which the waveform data should be transferred. Playback software can estimate the buffer size using this value.
nBlockAlign	The block alignment (in bytes) of the waveform data. Playback software needs to process a multiple of <nBlockAlign> bytes of data at a time, so the value of <nBlockAlign> can be used for buffer alignment.

The <format-specific-fields> consists of zero or more bytes of parameters. Which parameters occur depends on the WAVE format category – see the following sections for details. Playback software should be written to allow for (and ignore) any unknown <format-specific-fields> parameters that occur at the end of this field.

1.2 WAVE format categories

The format category of a WAVE file is specified by the value of the <wFormatTag> field of the 'fmt' chunk. The representation of data in <wave-data>, and the content of the <format-specific-fields> of the 'fmt' chunk, depend on the format category.

Among the currently defined open non-proprietary WAVE format categories are as follows:

wFormatTag	Value	Format Category
WAVE_FORMAT_PCM	(0x0001)	Microsoft® (PCM) format
WAVE_FORMAT_MPEG	(0x0050)	MPEG-1 Audio (audio only)

NOTE – Although other WAVE formats are registered with Microsoft®, only the above formats are used at present with the BWF. Details of the PCM WAVE format are given in the following Section 2. General information on other WAVE formats is given in § 3. Details of MPEG WAVE format are given in Annex 2. Other WAVE formats may be defined in future.

2 PCM format

If the <wFormatTag> field of the <fmt-ck> is set to WAVE_FORMAT_PCM, then the waveform data consists of samples represented in PCM format. For PCM waveform data, the <format-specific-fields> is defined as follows:

```
<PCM-format-specific> ->
    struct{
        WORD nBitsPerSample,    /* Sample size */
    }
```

The <nBitsPerSample> field specifies the number of bits of data used to represent each sample of each channel. If there are multiple channels, the sample size is the same for each channel.

The <nBlockAlign> field should be equal to the following formula, rounded to the next whole number

$$nchannels \times BytesPerSample$$

The value of BytesPerSample should be calculated by rounding up nBitsPerSample to the next whole byte. Where the audio sample word is less than an integer number of bytes, the most significant bits of the audio sample are placed in the most significant bits of the data word, the unused data bits adjacent to the least significant bit should be set to zero

For PCM data, the <nAvgBytesPerSec> field of the 'fmt' chunk should be equal to the following formula.

$$nSamplesPerSec \times nBblockAlign$$

NOTE – The original WAVE specification permits, for example 20-bit samples from two channels to be packed into 5 bytes-sharing a single byte for the least significant bits of the two channels. This Recommendation specifies a whole number of bytes per audio sample in order to reduce ambiguity in implementations and to achieve maximum interchange compatibility.

2.1 Data packing for PCM WAVE files

In a single-channel WAVE file, samples are stored consecutively. For stereo WAVE files, channel 0 represents the left-hand channel, and channel 1 represents the right-hand channel. In multiple-channel WAVE files, samples are interleaved.

The following diagrams show the data packing for 8-bit mono and stereo WAVE files:

Data packing for 8-bit mono PCM

Sample 1	Sample 2	Sample 3	Sample 4
Channel 0	Channel 0	Channel 0	Channel 0

Data packing for 8-bit stereo PCM

Sample 1		Sample 2	
Channel 0 (left)	Channel 1 (right)	Channel 0 (left)	Channel 1 (right)

The following diagrams show the data packing for 16-bit mono and stereo WAVE files:

Data packing for 16-bit mono PCM

Sample 1		Sample 2	
Channel 0 low-order byte	Channel 0 high-order byte	Channel 0 low-order byte	Channel 0 high-order byte

Data packing for 16-bit stereo PCM

Sample 1			
Channel 0 (left)	Channel 0 (left)	Channel 1 (right)	Channel 1 (right)
low-order byte	high-order byte	low-order byte	high-order byte

2.2 Data format of the samples

Each sample is contained in an integer i . The size of i is the smallest number of bytes required to contain the specified sample size. The least significant byte is stored first. The bits that represent the sample amplitude are stored in the most significant bits of i , and the remaining bits are set to zero.

For example, if the sample size (recorded in `<nBitsPerSample>`) is 12 bits, then each sample is stored in a two-byte integer. The least significant four bits of the first (least significant) byte are set to zero. The data format and maximum and minimum values for PCM waveform samples of various sizes are as follows:

Sample size	Data format	Maximum value	Minimum value
One to eight bits	Unsigned integer	255 (0xFF)	0
Nine or more bits	Signed integer i	Largest positive value of i	Most negative value of i

For example, the maximum, minimum, and midpoint values for 8-bit and 16-bit PCM waveform data are as follows:

Format	Maximum value	Minimum value	Midpoint value
8-bit PCM	255 (0xFF)	0	128 (0x80)
16-bit PCM	32767 (0x7FFF)	-32768 (-0x8000)	0

2.3 Examples of PCM WAVE files

Example of a PCM WAVE file with 11.025 kHz sampling rate, mono, 8 bits per sample:

```
RIFF('WAVE' fmt(1, 1, 11025, 11025, 1, 8)
      data( <wave-data> ) )
```

Example of a PCM WAVE file with 22.05 kHz sampling rate, stereo, 8 bits per sample:

```
RIFF('WAVE' fmt(1, 2, 22050, 44100, 2, 8)
      data( <wave-data> ) )
```

Example of a PCM WAVE file with 44.1 kHz sampling rate, mono, 20 bits per sample:

```
RIFF('WAVE' INFO(INAM("O Canada"Z) )
      fmt(1, 1, 44100, 132300, 3, 20)
      data( <wave-data> ) )
```

2.4 Storage of WAVE data

The <wave-data> contains the waveform data. It is defined as follows:

```
<wave-data> -> { <data-ck> }
<data-ck> -> data( <wave-data> )
```

2.5 Fact chunk

The <fact-ck> fact chunk stores important information about the contents of the WAVE file. This chunk is defined as follows:

```
<fact-ck> -> fact( <dwFileSize:DWORD> ) /*Number of samples */
```

The chunk is not required for PCM files.

The fact chunk will be expanded to include any other information required by future WAVE formats. Added fields will appear following the <dwFileSize> field. Applications can use the chunk size field to determine which fields are present.

2.6 Other optional chunks

A number of other chunks are specified for use in the WAVE format. Details of these chunks are given in the specification of the WAVE format and any updates issued later.

NOTE – The WAVE format can support other optional chunks that can be included in WAVE files to carry specific information. As stated in Note to § 2.1 of Annex 1, in the Broadcast Wave Format File these are considered to be private chunks and will be ignored by applications which cannot interpret them.

3 Other WAVE types

The following information has been extracted from the Microsoft® Data Standards. It outlines the necessary extensions of the basic WAVE files (used for PCM audio) to cover other types of WAVE format.

3.1 General information

All newly defined WAVE types must contain both a <fact-ck> and an extended wave format description within the <fmt-ck> format chunk. RIFF WAVE files of type WAVE_FORMAT_PCM need not have the extra chunk nor the extended wave format description.

3.2 Fact chunk

This chunk stores file dependent information about the contents of the WAVE file. It currently specifies the length of the file in samples.

WAVE format extension

The extended wave format structure added to the **<fmt-ck>** is used to define all non-PCM format wave data, and is described as follows. The general extended waveform format structure is used for all non PCM formats.

```
typedef struct waveformat_extended_tag {
    WORD  wFormatTag,           /* format type */
    WORD  nChannels,           /* number of channels (i.e. mono, stereo...) */
    DWORD nSamplesPerSec,      /* sample rate */
    DWORD nAvgBytesPerSec,     /* for buffer estimation */
    WORD  nBlockAlign,         /* block size of data */
    WORD  wBitsPerSample,      /* number of bits per sample of mono data */
    WORD  cbSize,              /* the count in bytes of the extra size */
} WAVEFORMATEX;
```

Field	Description
wFormatTag	Defines the type of WAVE file.
nChannels	Number of channels in the wave, 1 for mono, 2 for stereo.
nSamplesPerSec	Frequency of the sample rate of the wave file. This should be 48000 or 44100 etc. This rate is also used by the sample size entry in the fact chunk to determine the duration of the data.
nAvgBytesPerSec	Average data rate. Playback software can estimate the buffer size using the <nAvgBytesPerSec> value.
nBlockAlign	The block alignment (in bytes) of the data in <data-ck> . Playback software needs to process a multiple of <nBlockAlign> bytes of data at a time, so that the value of <nBlockAlign> can be used for buffer alignment.
wBitsPerSample	This is the number of bits per sample per channel. Each channel is assumed to have the same sample resolution. If this field is not needed, then it should be set to zero.
cbSize	The size in bytes of the extra information in the WAVE format header not including the size of the WAVEFORMATEX structure.

NOTE – The fields following the **<cbSize>** field contain specific information needed for the WAVE format defined in the field **<wFormatTag>**. Any WAVE formats which can be used in the BWF will be specified in individual Supplements to this Recommendation.