
wavinfo Documentation

Release 3.0.0

Jamie Hardt

Nov 20, 2023

CONTENTS

1	wavinfo Quickstart	3
2	Using <i>wavinfo</i> from the Command Line	5
3	ADM (Audio Definition Model) Metadata	7
4	Broadcast WAV Extension Metadata	9
5	Cue Marker and Range Metadata	13
6	Dolby Metadata	15
7	INFO Metadata	21
8	iXML Production Recorder Metadata	23
9	Other wavinfo Classes	27
10	References	29
11	Indices and tables	31
	Python Module Index	33
	Index	35

The *wavinfo* package allows you to probe WAVE and RF64/WAVE files and extract extended metadata, with an emphasis on film, video and professional music production metadata.

WAVINFO QUICKSTART

All metadata is read by an instance of *WaveInfoReader*. Each type of metadata, iXML, Broadcast-WAV etc. is accessible through *scopes*, properties on an instance of *WaveInfoReader*.

Listing 1: Using wavinfo

```
import wavinfo

path = 'path/to/your/wave/audio.wav'

info = wavinfo.WavInfoReader(path)

adm_metadata = info.adm
ixml_metadata = info.ixml
```

1.1 WavInfoReader Class Documentation

class `wavinfo.wave_reader.WavInfoReader`(*path*, *info_encoding*='latin_1', *bext_encoding*='ascii')

Parse a WAV audio file for metadata.

__init__(*path*, *info_encoding*='latin_1', *bext_encoding*='ascii')

Create a new reader object.

Parameters

- **path** – A pathlike object or IO to the wav file you wish to probe or a file handle to an open file.
- **info_encoding** – The text encoding of the INFO, LABL and other RIFF-defined metadata fields.
- **bext_encoding** – The text encoding to use when decoding the string fields of the Broadcast-WAV extension. Per EBU 3285 this is ASCII but this parameter is available to you if you encounter a weirdo.

adm: *WavADMReader* | None

ADM Audio Definiton Model metadata.

bext: *WavBextReader* | None

Broadcast-Wave metadata.

cues: *WavCuesReader* | None

RIFF cues markers, labels, and notes.

data: [WavDataDescriptor](#) | None

Statistics of the *data* section.

dolby: [WavDolbyMetadataReader](#) | None

Dolby bitstream metadata.

fmt: [WavAudioFormat](#) | None

Wave audio data format.

info: [WavInfoChunkReader](#) | None

RIFF INFO metadata.

ixml: [WavIXMLFormat](#) | None

iXML metadata.

url: **str**

file:// url for the file.

walk() → Generator[str, str, Any]

Walk all of the available metadata fields.

Yields

tuples of the *scope*, *key*, and *value* of each metadatum. The *scope* value will be one of “fmt”, “data”, “ixml”, “bext”, “info”, “dolby”, “cues” or “adm”.

USING WAVINFO FROM THE COMMAND LINE

wavinfo installs a command-line entry point that will read wav files from the command line and output metadata to stdout.

```
$ wavinfo [--ixml | --adm] INFILE +
```

By default, *wavinfo* will output a JSON dictionary for each file argument.

2.1 Options

Two option flags will change the behavior of the command:

--ixml

The *--ixml* flag will cause *wavinfo* to output the iXML metadata payload of each input wave file, or will emit an error message to stderr if iXML metadata is not present.

--adm

The *--adm* flag will cause *wavinfo* to output the ADM XML metadata payload of each input wave file, or will emit an error message to stderr if ADM XML metadata is not present.

These options are mutually-exclusive, with *--adm* taking precedence.

2.2 Example Output

```
{
  "filename": "tests/test_files/sounddevices/A101_1.WAV",
  "run_date": "2022-11-26T17:56:38.342935",
  "application": "wavinfo 2.1.0",
  "scopes": {
    "fmt": {
      "audio_format": 1,
      "channel_count": 2,
      "sample_rate": 48000,
      "byte_rate": 288000,
      "block_align": 6,
      "bits_per_sample": 24
    },
    "data": {
      "byte_count": 1441434,
```

(continues on next page)

(continued from previous page)

```

"frame_count": 240239
},
"ixml": {
  "track_list": [
    {
      "channel_index": "1",
      "interleave_index": "1",
      "name": "MKH516 A",
      "function": ""
    },
    {
      "channel_index": "2",
      "interleave_index": "2",
      "name": "Boom",
      "function": ""
    }
  ],
  "project": "BMH",
  "scene": "A101",
  "take": "1",
  "tape": "18Y12M31",
  "family_uid": "USSDVGR1112089007124001008206300",
  "family_name": null
},
  "bext": {
    "description": "sSPEED=023.976-ND\r\nsTAKE=1\r\nsUBITS=$12311801\r\nsSWVER=2.67\r\n
    ↪nsPROJECT=BMH\r\nsSCENE=A101\r\nsFILENAME=A101_1.WAV\r\nsTAPE=18Y12M31\r\nsTRK1=MKH516
    ↪A\r\nsTRK2=Boom\r\nsNOTE=\r\n",
    "originator": "Sound Dev: 702T S#GR1112089007",
    "originator_ref": "USSDVGR1112089007124001008206301",
    "originator_date": "2018-12-31",
    "originator_time": "12:40:00",
    "time_reference": 2190940753,
    "version": 1,
    "umid": "0000000000000000000000000000000000000000000000000000000000000000",
    "coding_history": "A=PCM,F=48000,W=24,M=stereo,R=48000,T=2 Ch\r\n",
    "loudness_value": null,
    "loudness_range": null,
    "max_true_peak": null,
    "max_momentary_loudness": null,
    "max_shortterm_loudness": null
  }
}
}

```

ADM (AUDIO DEFINITION MODEL) METADATA

3.1 Notes

[ADM metadata](#) is used in master recordings to describe the format and content of the tracks. In practice on wave files, ADM tells a client which tracks are members of multichannel stems or “beds” and their speaker assignment, and which tracks are freely-positioned 3D objects. ADM also records the panning moves on object tracks and their content group (“Dialogue”, “Music”, “Effects” etc.)

ADM wave files created with a Dolby Rendering and Mastering Unit are a common deliverable in feature film and television production. The [Dolby Atmos ADM Profile](#) describes how the RMU translates its native Master format into ADM.

3.2 Class Reference

class `wavinfo.wave_adm_reader.WavADMReader`(*axml_data*: bytes, *chna_data*: bytes)

Reads XML data from an EBU ADM (Audio Definiton Model) WAV File.

axml

An `lxml.etree` of the ADM XML document

programme() → dict

Read the ADM *audioProgramme* data structure and some of its reference properties.

to_dict() → dict

Get ADM metadata as a dictionary.

track_info(*index*) → dict | None

Information about a track in the WAV file.

Parameters

index – index of audio track (indexed from zero)

Returns

a dictionary with *content_name*, *content_id*, *object_name*, *object_id*, *pack_format_name*, *pack_type*, *channel_format_name*

xml_str() → str

ADM XML as a string

class `wavinfo.wave_adm_reader.ChannelEntry`(*track_index*, *uid*, *track_ref*, *pack_ref*)

pack_ref

Alias for field number 3

track_index

Alias for field number 0

track_ref

Alias for field number 2

uid

Alias for field number 1

BROADCAST WAV EXTENSION METADATA

4.1 Notes

A WAV file produced to Broadcast-WAV specifications will have the broadcast metadata extension, which includes a 256-character free text description, creating entity identifier (usually the recording application or equipment), the date and time of recording and a time reference for timecode synchronization.

The *coding_history* is designed to contain a record of every conversion performed on the audio file.

In this example (from a Sound Devices 702T) the bext metadata contains scene/take slating information in the *description*. Here also the *originator_ref* is a serial number conforming to EBU Rec 99.

If the bext metadata conforms to EBU 3285 v1, it will contain the WAV's 32 or 64 byte SMPTE ST 330 UMID. The 32-byte version of the UMID is usually just a random number, while the 64-byte UMID will also have information on the recording date and time, recording equipment and entity, and geolocation data.

If the bext metadata conforms to EBU 3285 v2, it will hold precomputed program loudness values as described by EBU Rec 128.

Note: All text fields in the Broadcast-WAV metadata structure are decoded by default as flat ASCII. To override this and use a different encoding, pass an string encoding name to the *bext_encoding* parameter of *WavInfoReader()*

4.2 Example

```
print(info.bext.description)
print("-----")
print("Originator:", info.bext.originator)
print("Originator Ref:", info.bext.originator_ref)
print("Originator Date:", info.bext.originator_date)
print("Originator Time:", info.bext.originator_time)
print("Time Reference:", info.bext.time_reference)
print(info.bext.coding_history)
```

Result:

```
sSPEED=023.976-ND
sTAKE=1
sSUBITS=$12311801
sSWVER=2.67
```

(continues on next page)

(continued from previous page)

```
sPROJECT=BMH
sSCENE=A101
sFILENAME=A101_1.WAV
sTAPE=18Y12M31
sTRK1=MKH516 A
sTRK2=Boom
sNOTE=

-----
Originator: Sound Dev: 702T S#GR1112089007
Originator Ref: USSDVGR1112089007124001008206301
Originator Date: 2018-12-31
Originator Time: 12:40:00
Time Reference: 2190940753
A=PCM,F=48000,W=24,M=stereo,R=48000,T=2 Ch
```

4.3 Class Reference

class wavinfo.wave_bext_reader.**WavBextReader**(*bext_data*, *encoding*)

coding_history: **str**

A variable-length text field containing a list of processes and and conversions performed on the file.

description: **str**

Description. A free-text field up to 256 characters long.

loudness_range: **float** | **None**

EBU R128 Loudness range, in LUFS.

loudness_value: **float** | **None**

EBU R128 Integrated loudness, in LUFS.

max_momentary_loudness: **float** | **None**

EBU R128 Maximum momentary loudness, in LUFS

max_shortterm_loudness: **float** | **None**

EBU R128 Maximum short-term loudness, in LUFS.

max_true_peak: **float** | **None**

True peak level, in dBFS TP

originator: **str**

Originator. Usually the name of the encoding application, sometimes an artist name.

originator_date: **str**

Date of the recording, in the format YYYY-MM-DD.

originator_ref: **str**

A unique identifier for the file, a serial number.

originator_time: **str**

Time of the recording, in the format HH:MM:SS.

time_reference: `int`

The sample offset of the start, usually relative to midnight.

umid: `bytes` | `None`

SMPTE 330M UMID of this audio file, 64 bytes are allocated though the UMID may only be 32 bytes long.

version: `int`

BEXT version.

CUE MARKER AND RANGE METADATA

5.1 Notes

Cue metadata stores timed markers that clients use to mark times of interest in a wave file, and optionally give them a name and longer comment. Markers can also have an associated length, allowing ranges of times in a file to be marked.

5.1.1 String Encoding of Cue Metadata

Cue labels and notes will be decoded using the string encoding passed to `WavInfoReader`'s `info_encoding=` parameter, which by default is `latin_1` (ISO 8859-1).

Text associated with `ltxx` time ranges may specify their own encoding in the form of a Windows codepage number. `wavinfo` will attempt to use the encoding specified.

Note: `cset` character set/locale metadata is not supported. If it is present in the file it will be ignored by `wavinfo`.

5.2 Class Reference

```
class wavinfo.wave_cues_reader.WavCuesReader(cues: List[wavinfo.wave_cues_reader.CueEntry], labels:
    List[wavinfo.wave_cues_reader.LabelEntry], ranges:
    List[wavinfo.wave_cues_reader.RangeLabel], notes:
    List[wavinfo.wave_cues_reader.LabelEntry])
```

cues: `List[CueEntry]`

Every cue entry in the file

each_cue() \rightarrow `Generator[Tuple[int, int], None, None]`

Iterate through each cue.

Yields

the cue's name and `sample_offset`

label_and_note(`cue_ident: int`) \rightarrow `Tuple[str | None, str | None]`

Get the label and note (extended comment) for a cue.

Parameters

cue_ident – the cue's name, its unique identifying number

Returns

a tuple of the the cue's label (if present) and note (if present)

labels: `List[LabelEntry]`

Every labl in the file

notes: `List[LabelEntry]`

Every note in the file

range(*cue_ident: int*) \rightarrow `int | None`

Get the length of the time range for a cue, if it has one.

Parameters

cue_ident – the cue’s name, its unique identifying number

Returns

the length of the marker’s range, or *None*

ranges: `List[RangeLabel]`

Every ltxt in the file

class `wavinfo.wave_cues_reader.CueEntry`(*name: int, position: int, chunk_id: bytes, chunk_start: int, block_start: int, sample_offset: int*)

A cue element structure.

block_start: `int`

Alias for field number 4

chunk_id: `bytes`

Alias for field number 2

chunk_start: `int`

Alias for field number 3

name: `int`

Cue “name” or id number

position: `int`

Cue position, as a frame count in the play order of the WAVE file. In principle this can be affected by playlists and wavl chunk placement.

sample_offset: `int`

Alias for field number 5

class `wavinfo.wave_cues_reader.LabelEntry`(*name: int, text: str*)

A labl structure.

name: `int`

Alias for field number 0

text: `str`

Alias for field number 1

`wavinfo.wave_cues_reader.NoteEntry`

alias of `LabelEntry`

DOLBY METADATA

6.1 Notes

Dolby software and equipment creates detailed hinting metadata that can help receiving applications decide how to present the audio content, particularly how it should be downmixed, and dialogue normalization settings.

6.2 Class Reference

Reading Dolby Bitstream Metadata

Unless otherwise stated, all § references here are to [EBU Tech 3285 Supplement 6](#).

class `wavinfo.wave_dbmd_reader.WavDolbyMetadataReader(dbmd_data)`

Reads Dolby bitstream metadata.

dolby_atmos() → `List[DolbyAtmosMetadata]`

Every valid Dolby Atmos metadata segment in the file.

dolby_atmos_supplemental() → `List[DolbyAtmosSupplementalMetadata]`

Every valid Dolby Atmos Supplemental metadata segment in the file.

dolby_digital_plus() → `List[DolbyDigitalPlusMetadata]`

Every valid Dolby Digital Plus metadata segment in the file.

segment_list: `List[Tuple[SegmentType | int, bool, Any]]`

List of the Dolby Metadata Segments.

Each list entry is a tuple of *SegmentType*, a *bool* indicating if the segment's checksum was valid, and the segment's parsed dataclass (or a *bytes* array if it was not recognized).

```
class wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata(program_id: int, lfe_on: bool,
                                                         bitstream_mode: BitStreamMode,
                                                         audio_coding_mode: AudioCodingMode,
                                                         center_downmix_level:
                                                         CenterDownMixLevel,
                                                         surround_downmix_level:
                                                         SurroundDownMixLevel,
                                                         dolby_surround_encoded:
                                                         DolbySurroundEncodingMode,
                                                         langcode_present: bool,
                                                         copyright_bitstream: bool,
                                                         original_bitstream: bool, dialnorm:
                                                         DialnormLevel, langcode: int,
                                                         prod_info_exists: bool, mixlevel:
                                                         MixLevel, roomtype: RoomType,
                                                         loro_center_downmix_level:
                                                         DownMixLevelToken,
                                                         loro_surround_downmix_level:
                                                         DownMixLevelToken, downmix_mode:
                                                         PreferredDownMixMode,
                                                         ltrt_center_downmix_level:
                                                         DownMixLevelToken,
                                                         ltrt_surround_downmix_level:
                                                         DownMixLevelToken, surround_ex_mode:
                                                         SurroundEXMode,
                                                         dolby_headphone_encoded:
                                                         HeadphoneMode, ad_converter_type:
                                                         ADConverterType, compression_profile:
                                                         RFCompressionProfile, dynamic_range:
                                                         RFCompressionProfile,
                                                         stream_dependency: StreamDependency,
                                                         datarate_kbps: int)
```

Dolby Digital Plus is Dolby's brand for multichannel surround on discrete formats that aren't AC-3 (Dolby Digital) or Dolby E. This metadata segment is present in ADM wave files created with a Dolby Atmos Production Suite.

Where an AC-3 bitstream can contain multiple programs, a Dolby Digital Plus bitstream will only contain one program.

class ADConverterType(value)

An enumeration.

class AudioCodingMode(value)

Dolby Digital Plus *acmod* field § 4.3.2.3

CH_ORD_1_0 = 1

Mono

CH_ORD_2_0 = 2

L/R stereo

CH_ORD_2_1 = 4

LR + mono surround

CH_ORD_2_2 = 6

LR + LR surround

```

CH_ORD_3_0 = 3
    LCR stereo

CH_ORD_3_1 = 5
    LCR + mono surround

CH_ORD_3_2 = 7
    LCR + LR surround

class BitStreamMode(value)
    Dolby Digital Plus bsmod field § 4.3.2.2

    COMMENTARY = 5
        associated service: commentary

    COMPLETE_MAIN = 0
        main audio service: complete main

    DIALOGUE_ONLY = 4
        associated service: dialogue

    EMERGENCY = 6
        associated service: emergency

    HEARING_IMPAIRED = 3
        associated service: hearing impaired

    MUSIC_AND_EFFECTS = 1
        main audio service: music and effects

    VISUALLY_IMPAIRED = 2
        associated service: visually impaired

    VOICEOVER_KARAOKE = 7
        associated service: voice over OR main audio service: karaoke. If acmod is 0b001 (mono 1/0), this is
        voice-over, otherwise it should be interpreted as karaoke.

class CenterDownMixLevel(value)
    § 4.3.3.1

    DOWN_3DB = 0
        Attenuate 3 dB

    DOWN_45DB = 1
        Attenuate 4.5 dB

    DOWN_6DB = 2
        Attenuate 6 dB

class DialnormLevel
    § 4.3.4.4

class DolbySurroundEncodingMode(value)
    Dolby surround encoding mode.

class DownMixLevelToken(value)
    A gain coefficient used in several metadata fields for downmix scenarios.

```

MINUS_1_5DB = 3

-1.5 dB

MINUS_3DB = 4

-3 dB

MINUS_4_5DB = 5

-4.5 dB

MINUS_6DB = 6

-6 dB

MUTE = 7

$-\infty$ dB

PLUS_1_5DB = 1

+1.5 dB

PLUS_3DB = 0

+3 dB

UNITY = 2

0dB

class HeadphoneMode(*value*)

dheadphonmod § 4.3.9.2

class LanguageCode

§ 4.3.4.1

Per ATSC/A52 § 5.4.2.12, this is not in use and always 0xFF.

class MixLevel

§ 4.3.6.2

class PreferredDownMixMode(*value*)

Indicates the creating engineer's preference of what the receiver should downmix. § 4.3.8.1

class RFCompressionProfile(*value*)

compr1 RF compression profile § 4.3.10 (fig 42)

class RoomType(*value*)

roomtyp 4.3.6.3

class StreamDependency(*value*)

Encodes *ddplus_info1.stream_type* field § 4.3.12.1

class SurroundDownMixLevel(*value*)

Dolby Digital Plus *surmixlev* field § 4.3.3.2

class SurroundEXMode(*value*)

Dolby Surround-EX mode. *dsurexmod* § 4.3.9.1

audio_coding_mode: [AudioCodingMode](#)

Indicates which channels are in use. *acmod* § 4.3.2.3

bitstream_mode: [BitStreamMode](#)

The kind of service of this stream. *bsmod* § 4.3.2.2

center_downmix_level: [CenterDownMixLevel](#)

When the front three channels are in use, gives the center downmix level. ``

copyright_bitstream: **bool**

True if this bitstream is copyrighted.

datarate_kbps: **int**

Data rate of this bitstream in kilobits per second

dolby_headphone_encoded: [HeadphoneMode](#)

Dolby Headphone mode

dolby_surround_encoded: [DolbySurroundEncodingMode](#)

If the *acmod* is LR, this indicates if the channels are encoded in Dolby Surround.

downmix_mode: [PreferredDownMixMode](#)

Preferred downmix mode

langcode: **int**

Language code

langcode_present: **bool**

True if there is a langcode present in the metadata.

lfe_on: **bool**

True if LFE is enabled. § 4.3.2.1

loro_center_downmix_level: [DownMixLevelToken](#)

LoRo preferred center downmix level

loro_surround_downmix_level: [DownMixLevelToken](#)

LoRo preferred surround downmix level

ltrt_center_downmix_level: [DownMixLevelToken](#)

LtRt preferred center downmix level

ltrt_surround_downmix_level: [DownMixLevelToken](#)

LtRt preferred surround downmix level

mixlevel: [MixLevel](#)

Mix level

original_bitstream: **bool**

True if this bitstream is original.

prod_info_exists: **bool**

True if *mixlevel* and *roomtype* are valid

program_id: **int**

Program ID number, this identifies the program in a multi-program element. § 4.3.1

roomtype: [RoomType](#)

Room Type

stream_dependency: [StreamDependency](#)

Indicates if this stream can be decoded independently or not

surround_downmix_level: *SurroundDownMixLevel*

When the surround channels are in use, gives the surround downmix level.

surround_ex_mode: *SurroundEXMode*

Surround-EX mode

INFO METADATA

7.1 Notes

INFO Metadata is a standard method for saving tagged text data in a WAV or AVI file. INFO fields are often read by the file explorer and host OS, and used in music library software.

```
bullet_path = '../tests/test_files/BULLET Impact Plastic LCD TV Screen Shatter Debris 2x.  
↳wav'  
  
bullet = WavInfoReader(bullet_path)  
  
print("INFO Artist:",    bullet.info.artist)  
print("INFO Copyright:", bullet.info.copyright)  
print("INFO Comment:",   bullet.info.comment)
```

7.1.1 String Encoding of INFO Metadata

Info metadata fields will be decoded using the string encoding passed to `WavInfoReader`'s `info_encoding=` parameter, which by default is `latin_1` (ISO 8859-1).

Note: cset character set/locale metadata is not supported. If it is present in the file it will be ignored by *wavinfo*.

7.2 Class Reference

`class wavinfo.wave_info_reader.WavInfoChunkReader(f, encoding)`

archival_location: str | None

‘IARL’ Archival Location

artist: str | None

‘IART’ Artist, composer, author

comment: str | None

‘ICMT’ Comment

commissioned: str | None

‘ICSM’ Commissioned

copyright: str | None

‘ICOP’ Copyright

created_date: str | None

‘ICRD’ Created date

engineer: str | None

‘IENG’ Engineer

genre: str | None

‘IGNR’ Genre

keywords: str | None

‘IKEY’ Keywords, keyword list

product: str | None

‘IPRD’ Product

software: str | None

‘ISFT’ Software, encoding application

source: str | None

‘ISRC’ Source

subject: str | None

‘ISBJ’ Subject

tape: str | None

‘TAPE’ Tape

technician: str | None

‘ITCH’ Technician

title: str | None

‘INAM’ Name, title

to_dict() → dict

A dictionary with all of the key/values read from the INFO scope.

IXML PRODUCTION RECORDER METADATA

8.1 Notes

iXML allows an XML document to be embedded in a WAV file.

The iXML website recommends a schema for recorder information but there is no official DTD and vendors mostly do their own thing, apart from hitting a few key xpaths. iXML is used by most location/production recorders to save slating information, timecode and sync points in a reliable way.

iXML is also used to link “families” of WAV files together, so WAV files recorded simultaneously or contiguously can be related by a receiving client.

```
print("iXML Project:", info.ixml.project)
print("iXML Scene:", info.ixml.scene)
print("iXML Take:", info.ixml.take)
print("iXML Tape:", info.ixml.tape)
print("iXML File Family Name:", info.ixml.family_name)
print("iXML File Family UID:", info.ixml.family_uid)
```

Result:

```
iXML Project: BMH
iXML Scene: A101
iXML Take: 1
iXML Tape: 18Y12M31
iXML File Family Name: None
iXML File Family UID: USSDVGR1112089007124001008206300
```

8.2 Class Reference

class wavinfo.wave_ixml_reader.WavIXMLFormat(*xml*)

iXML recorder metadata.

property family_name: str | None

The name of this file’s file family.

property family_uid: str | None

The globally-unique ID for this file family. This may be in the format of a GUID, or an EBU Rec 9 source identifier, or some other dumb number.

property project: `str` | `None`
The project/film name entered for the recording.

property raw_xml: `ElementTree`
The root entity of the iXML document.

property scene: `str` | `None`
Scene/slate.

property steinberg: [*SteinbergMetadata*](#) | `None`
Steinberg vendor iXML metadata if present.

property take: `str` | `None`
Take number.

property tape: `str` | `None`
Tape name.

property track_list
A description of each track.

Yields
IXMLTrack for each track.

8.3 Steinberg-Specific iXML Metadata

```
class wavinfo.wave_ixml_reader.SteinbergMetadata(xml: ElementTree)
    Vendor-specific Steinberg metadata.

    class AudioSpeakerArrangement(value)
        Steinberg speaker format enumeration.

    property audio_speaker_arrangement: AudioSpeakerArrangement | None
        AudioSpeakerArrangement property

    property media_company: str | None
        MediaCompany

    property media_drop_frames: bool | None
        MediaDropFrames

    property media_duration: float | None
        MediaDuration

    property media_start_time: float | None
        MediaStartTime

    property media_track_title: str | None
        MediaTrackTitle

    classmethod present(xml: ElementTree) → bool
        Test if xml has Steinberg metadata. :param xml: an iXML ElementTree

    property program_name: str | None
        ProgramName
```

property program_version: str | None

ProgramVersion

property sample_format_size: int | None

AudioSampleFormatSize

OTHER WAVINFO CLASSES

```
class wavinfo.wave_reader.WavAudioFormat(audio_format, channel_count, sample_rate, byte_rate,  
                                         block_align, bits_per_sample)
```

```
    audio_format: int
```

```
        Alias for field number 0
```

```
    bits_per_sample: int
```

```
        Alias for field number 5
```

```
    block_align: int
```

```
        Alias for field number 4
```

```
    byte_rate: int
```

```
        Alias for field number 3
```

```
    channel_count: int
```

```
        Alias for field number 1
```

```
    sample_rate: int
```

```
        Alias for field number 2
```

```
class wavinfo.wave_reader.WavDataDescriptor(byte_count, frame_count)
```

```
    byte_count: int
```

```
        Alias for field number 0
```

```
    frame_count: int
```

```
        Alias for field number 1
```


REFERENCES

A complete list of technical references and commentary is available as a man page and is installed as `wavinfo(7)` when you install *wavinfo* via pip.

10.1 Wave File Format

- ITU Recommendation BS.2088-1-2019 — Long-form file format for the international exchange of audio programme materials with metadata
- IETF Network Working Group RFC2361 — WAVE and AVI Codec Registries

10.2 Broadcast Wave Format

- EBU Tech 3285 — Specification of the Broadcast Wave Format (BWF)
- EBU Tech 3306 — MBWF / RF64: An extended File Format for Audio
- SMPTE ST 330-2011 — Unique Material Identifier

10.3 Audio Definition Model

- ITU Recommendation BS.2076-2-2019 — Audio definition model
- EBU Tech 3285 Supplement 5 — <axml> Chunk
- EBU ADM Guidelines

10.4 Dolby

- EBU Tech 3285 Supplement 6 — Dolby Metadata
- Dolby Laboratories Atmos ADM Profile

10.5 iXML

- [Gallery Software iXML Specification](#)

10.6 RIFF Metadata

- [1991. Multimedia Programming Interface and Data Specifications 1.0](#)
- [Exiftool Documentation](#)

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

W

wavinfo, [7](#)

wavinfo.wave_dbmd_reader, [15](#)

Symbols

`__init__()` (wavinfo.wave_reader.WavInfoReader method), 3

A

`adm` (wavinfo.wave_reader.WavInfoReader attribute), 3

`archival_location` (wavinfo.wave_info_reader.WavInfoChunkReader attribute), 21

`artist` (wavinfo.wave_info_reader.WavInfoChunkReader attribute), 21

`audio_coding_mode` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 18

`audio_format` (wavinfo.wave_reader.WavAudioFormat attribute), 27

`audio_speaker_arrangement` (wavinfo.wave_ixml_reader.SteinbergMetadata property), 24

`axml` (wavinfo.wave_adm_reader.WavADMReader attribute), 7

B

`bext` (wavinfo.wave_reader.WavInfoReader attribute), 3

`bits_per_sample` (wavinfo.wave_reader.WavAudioFormat attribute), 27

`bitstream_mode` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 18

`block_align` (wavinfo.wave_reader.WavAudioFormat attribute), 27

`block_start` (wavinfo.wave_cues_reader.CueEntry attribute), 14

`byte_count` (wavinfo.wave_reader.WavDataDescriptor attribute), 27

`byte_rate` (wavinfo.wave_reader.WavAudioFormat attribute), 27

C

`center_downmix_level` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata

attribute), 18

`CH_ORD_1_0` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.Audi

attribute), 16

`CH_ORD_2_0` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.Audi

attribute), 16

`CH_ORD_2_1` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.Audi

attribute), 16

`CH_ORD_2_2` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.Audi

attribute), 16

`CH_ORD_3_0` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.Audi

attribute), 16

`CH_ORD_3_1` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.Audi

attribute), 17

`CH_ORD_3_2` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.Audi

attribute), 17

`channel_count` (wavinfo.wave_reader.WavAudioFormat attribute), 27

`ChannelEntry` (class in wavinfo.wave_adm_reader), 7

`chunk_id` (wavinfo.wave_cues_reader.CueEntry attribute), 14

`chunk_start` (wavinfo.wave_cues_reader.CueEntry attribute), 14

`coding_history` (wavinfo.wave_bext_reader.WavBextReader attribute), 10

`comment` (wavinfo.wave_info_reader.WavInfoChunkReader attribute), 21

`COMMENTARY` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.BitSt

attribute), 17

`commissioned` (wavinfo.wave_info_reader.WavInfoChunkReader attribute), 21

`COMPLETE_MAIN` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.L

attribute), 17

`copyright` (wavinfo.wave_info_reader.WavInfoChunkReader attribute), 21

`copyright_bitstream` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 19

`created_date` (wavinfo.wave_info_reader.WavInfoChunkReader attribute), 22

`CueEntry` (class in wavinfo.wave_cues_reader), 14

`cues` (wavinfo.wave_cues_reader.WavCuesReader

attribute), 13
 cues (*wavinfo.wave_reader.WavInfoReader attribute*), 3

D

data (*wavinfo.wave_reader.WavInfoReader attribute*), 3
 datarate_kbps (*wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute*), 19
 description (*wavinfo.wave_bext_reader.WavBextReader attribute*), 10
 DIALOGUE_ONLY (*wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute*), 17
 dolby (*wavinfo.wave_reader.WavInfoReader attribute*), 4
 dolby_atmos() (*wavinfo.wave_dbmd_reader.WavDolbyMetadataReader method*), 15
 dolby_atmos_supplemental() (*wavinfo.wave_dbmd_reader.WavDolbyMetadataReader method*), 15
 dolby_digital_plus() (*wavinfo.wave_dbmd_reader.WavDolbyMetadataReader method*), 15
 dolby_headphone_encoded (*wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute*), 19
 dolby_surround_encoded (*wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute*), 19
 DolbyDigitalPlusMetadata (*class in wavinfo.wave_dbmd_reader*), 15
 DolbyDigitalPlusMetadata.ADConverterType (*class in wavinfo.wave_dbmd_reader*), 16
 DolbyDigitalPlusMetadata.AudioCodingMode (*class in wavinfo.wave_dbmd_reader*), 16
 DolbyDigitalPlusMetadata.BitStreamMode (*class in wavinfo.wave_dbmd_reader*), 17
 DolbyDigitalPlusMetadata.CenterDownMixLevel (*class in wavinfo.wave_dbmd_reader*), 17
 DolbyDigitalPlusMetadata.DialnormLevel (*class in wavinfo.wave_dbmd_reader*), 17
 DolbyDigitalPlusMetadata.DolbySurroundEncodingMode (*class in wavinfo.wave_dbmd_reader*), 17
 DolbyDigitalPlusMetadata.DownMixLevelToken (*class in wavinfo.wave_dbmd_reader*), 17
 DolbyDigitalPlusMetadata.HeadphoneMode (*class in wavinfo.wave_dbmd_reader*), 18
 DolbyDigitalPlusMetadata.LanguageCode (*class in wavinfo.wave_dbmd_reader*), 18
 DolbyDigitalPlusMetadata.MixLevel (*class in wavinfo.wave_dbmd_reader*), 18
 DolbyDigitalPlusMetadata.PreferredDownMixMode (*class in wavinfo.wave_dbmd_reader*), 18
 DolbyDigitalPlusMetadata.RFCompressionProfile (*class in wavinfo.wave_dbmd_reader*), 18
 DolbyDigitalPlusMetadata.RoomType (*class in wavinfo.wave_dbmd_reader*), 18
 DolbyDigitalPlusMetadata.StreamDependency (*class in wavinfo.wave_dbmd_reader*), 18
 DolbyDigitalPlusMetadata.SurroundDownMixLevel (*class in wavinfo.wave_dbmd_reader*), 18
 DolbyDigitalPlusMetadata.SurroundEXMode (*class in wavinfo.wave_dbmd_reader*), 18
 DOWN_3DB (*wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.CenterDownMixLevel attribute*), 17
 DOWN_45DB (*wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.CenterDownMixLevel attribute*), 17
 DOWN_6DB (*wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.CenterDownMixLevel attribute*), 17
 downmix_mode (*wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute*), 19
 E
 each_cue() (*wavinfo.wave_cues_reader.WavCuesReader method*), 13
 EMERGENCY (*wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.BitStreamMode attribute*), 17
 engineer (*wavinfo.wave_info_reader.WavInfoChunkReader attribute*), 22
 F
 family_name (*wavinfo.wave_ixml_reader.WavIXMLFormat property*), 23
 family_uid (*wavinfo.wave_ixml_reader.WavIXMLFormat property*), 23
 fmt (*wavinfo.wave_reader.WavInfoReader attribute*), 4
 frame_count (*wavinfo.wave_reader.WavDataDescriptor attribute*), 27
 G
 genre (*wavinfo.wave_info_reader.WavInfoChunkReader attribute*), 22
 H
 HEARING_IMPAIRED (*wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.BitStreamMode attribute*), 17
 I
 info (*wavinfo.wave_reader.WavInfoReader attribute*), 4
 ixml (*wavinfo.wave_reader.WavInfoReader attribute*), 4
 K
 keywords (*wavinfo.wave_info_reader.WavInfoChunkReader attribute*), 22
 L
 label_and_note() (*wavinfo.wave_cues_reader.WavCuesReader method*), 13

LabelEntry (class in wavinfo.wave_cues_reader), 14
 labels (wavinfo.wave_cues_reader.WavCuesReader attribute), 14
 langcode (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 19
 langcode_present (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 19
 life_on (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 19
 loro_center_downmix_level (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 19
 loro_surround_downmix_level (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 19
 loudness_range (wavinfo.wave_bext_reader.WavBextReader attribute), 10
 loudness_value (wavinfo.wave_bext_reader.WavBextReader attribute), 10
 ltrt_center_downmix_level (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 19
 ltrt_surround_downmix_level (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 19

M

max_momentary_loudness (wavinfo.wave_bext_reader.WavBextReader attribute), 10
 max_shortterm_loudness (wavinfo.wave_bext_reader.WavBextReader attribute), 10
 max_true_peak (wavinfo.wave_bext_reader.WavBextReader attribute), 10
 media_company (wavinfo.wave_ixml_reader.SteinbergMetadata property), 24
 media_drop_frames (wavinfo.wave_ixml_reader.SteinbergMetadata property), 24
 media_duration (wavinfo.wave_ixml_reader.SteinbergMetadata property), 24
 media_start_time (wavinfo.wave_ixml_reader.SteinbergMetadata property), 24
 media_track_title (wavinfo.wave_ixml_reader.SteinbergMetadata property), 24
 MINUS_1_5DB (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 17
 MINUS_3DB (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 18
 MINUS_4_5DB (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 18
 MINUS_6DB (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 18
 mix_level (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 19
 module wavinfo, 7
 wavinfo.wave_dbmd_reader, 15
 MUSIC_AND_EFFECTS (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.BitStreamMode attribute), 17
 MUTE (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.DownMixLevel attribute), 18

N

name (wavinfo.wave_cues_reader.CueEntry attribute), 14
 name (wavinfo.wave_cues_reader.LabelEntry attribute), 14
 NoteEntry (in module wavinfo.wave_cues_reader), 14
 notes (wavinfo.wave_cues_reader.WavCuesReader attribute), 14

O

original_bitstream (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 19
 originator (wavinfo.wave_bext_reader.WavBextReader attribute), 10
 originator_date (wavinfo.wave_bext_reader.WavBextReader attribute), 10
 originator_ref (wavinfo.wave_bext_reader.WavBextReader attribute), 10
 originator_time (wavinfo.wave_bext_reader.WavBextReader attribute), 10

P

pack_ref (wavinfo.wave_adm_reader.ChannelEntry attribute), 7
 PLUS_1_5DB (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 18
 PLUS_3DB (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata attribute), 18
 position (wavinfo.wave_cues_reader.CueEntry attribute), 14
 present() (wavinfo.wave_ixml_reader.SteinbergMetadata class method), 24
 DownMixLevelToken

`prod_info_exists` (wav-
info.wave_dbmd_reader.DolbyDigitalPlusMetadata
attribute), 19

`product` (wavinfo.wave_info_reader.WavInfoChunkReader
attribute), 22

`program_id` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata
attribute), 19

`program_name` (wavinfo.wave_ixml_reader.SteinbergMetadata
property), 24

`program_version` (wav-
info.wave_ixml_reader.SteinbergMetadata
property), 24

`programme()` (wavinfo.wave_adm_reader.WavADMReader
method), 7

`project` (wavinfo.wave_ixml_reader.WavIXMLFormat
property), 23

R

`range()` (wavinfo.wave_cues_reader.WavCuesReader
method), 14

`ranges` (wavinfo.wave_cues_reader.WavCuesReader
attribute), 14

`raw_xml` (wavinfo.wave_ixml_reader.WavIXMLFormat
property), 24

`roomtype` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata
attribute), 19

S

`sample_format_size` (wav-
info.wave_ixml_reader.SteinbergMetadata
property), 25

`sample_offset` (wavinfo.wave_cues_reader.CueEntry
attribute), 14

`sample_rate` (wavinfo.wave_reader.WavAudioFormat
attribute), 27

`scene` (wavinfo.wave_ixml_reader.WavIXMLFormat
property), 24

`segment_list` (wavinfo.wave_dbmd_reader.WavDolbyMetadata
attribute), 15

`software` (wavinfo.wave_info_reader.WavInfoChunkReader
attribute), 22

`source` (wavinfo.wave_info_reader.WavInfoChunkReader
attribute), 22

`steinberg` (wavinfo.wave_ixml_reader.WavIXMLFormat
property), 24

`SteinbergMetadata` (class in wav-
info.wave_ixml_reader), 24

`SteinbergMetadata.AudioSpeakerArrangement`
(class in wavinfo.wave_ixml_reader), 24

`stream_dependency` (wav-
info.wave_dbmd_reader.DolbyDigitalPlusMetadata
attribute), 19

`subject` (wavinfo.wave_info_reader.WavInfoChunkReader
attribute), 22

T

`take` (wavinfo.wave_ixml_reader.WavIXMLFormat prop-
erty), 24

`tape` (wavinfo.wave_info_reader.WavInfoChunkReader
attribute), 22

`tape` (wavinfo.wave_ixml_reader.WavIXMLFormat prop-
erty), 24

`technician` (wavinfo.wave_info_reader.WavInfoChunkReader
attribute), 22

`text` (wavinfo.wave_cues_reader.LabelEntry attribute),
14

`time_reference` (wav-
info.wave_bext_reader.WavBextReader at-
tribute), 10

`title` (wavinfo.wave_info_reader.WavInfoChunkReader
attribute), 22

`to_dict()` (wavinfo.wave_adm_reader.WavADMReader
method), 7

`to_dict()` (wavinfo.wave_info_reader.WavInfoChunkReader
method), 22

`track_index` (wavinfo.wave_adm_reader.ChannelEntry
attribute), 8

`track_info()` (wavinfo.wave_adm_reader.WavADMReader
method), 7

`track_list` (wavinfo.wave_ixml_reader.WavIXMLFormat
property), 24

`track_ref` (wavinfo.wave_adm_reader.ChannelEntry
attribute), 8

U

`uid` (wavinfo.wave_adm_reader.ChannelEntry attribute),
8

`umid` (wavinfo.wave_bext_reader.WavBextReader at-
tribute), 11

`UNITY` (wavinfo.wave_dbmd_reader.DolbyDigitalPlusMetadata.DownMixL
attribute), 18

`url` (wavinfo.wave_reader.WavInfoReader attribute), 4

V

`version` (wavinfo.wave_bext_reader.WavBextReader at-
tribute), 11

`VISUALLY_IMPAIRED` (wav-
info.wave_dbmd_reader.DolbyDigitalPlusMetadata.BitStreamMo
attribute), 17

`VOICEOVER_KARAOKE` (wav-
info.wave_dbmd_reader.DolbyDigitalPlusMetadata.BitStreamMo
attribute), 17

W

`walk()` (*wavinfo.wave_reader.WavInfoReader* method), 4

`WavADMReader` (class in *wavinfo.wave_adm_reader*), 7

`WavAudioFormat` (class in *wavinfo.wave_reader*), 27

`WavBextReader` (class in *wavinfo.wave_bext_reader*), 10

`WavCuesReader` (class in *wavinfo.wave_cues_reader*), 13

`WavDataDescriptor` (class in *wavinfo.wave_reader*), 27

`WavDolbyMetadataReader` (class in *wavinfo.wave_dbmd_reader*), 15

`wavinfo` module, 7

`wavinfo.wave_dbmd_reader` module, 15

`WavInfoChunkReader` (class in *wavinfo.wave_info_reader*), 21

`WavInfoReader` (class in *wavinfo.wave_reader*), 3

`WavIXMLFormat` (class in *wavinfo.wave_xml_reader*), 23

X

`xml_str()` (*wavinfo.wave_adm_reader.WavADMReader* method), 7