International Standard

ISO/IEC 23090-31

First edition
2025-01

Information technology — Coded representation of immersive media —

Part 31: Haptics coding

Technologies de l'information — Représentation codée de média immersifs —

Partie 31: Codage haptique

Reference number
ISO/IEC 23090-31:2025(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

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

A list of all parts in the ISO/IEC 23090 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

Haptics provide an additional layer of entertainment and sensory immersion to the user. Therefore, the user experience and enjoyment of media content, from a ISOBMFF files, broadcast channel, streaming games or mobile advertisements can be significantly enhanced by the judicious addition of haptics to the audio/video content. To that end, haptics has been proposed as a first-order media type, akin to audio and video, in ISOBMFF. Further, haptics has also been proposed as an addition to ISO/IEC 23009-1 (MPEG-DASH) to signal the presence of haptics in MP4 segments to DASH streaming clients. Lastly, the ISO/IEC 23090 (MPEG-I) use cases have been augmented with haptics

Haptics digital encoding is the storing of tactile data in a digital format. As with audio and video, digital encoding is of fundamental importance to allow digital haptic devices to function. Haptics encoding gained relevance with the increased market importance of wideband haptics in consumer peripherals such as smartphones with haptic engines and game consoles with haptic enabled controllers. The prior generation of haptics peripherals was based on less expressive haptic actuators usually based on state machine control processes.

In the field of haptics, the signal encoding usually takes one of two approaches:

— Quantized: This representation is generally made from measured data. The samples from the original phenomenon are stored at a specific acquisition frequency inside the file to represent this signal. One example of a quantized haptic signal is proposed through WAV files, originally developed for audio. WAV file formalism allows the capture of real-world data and the representation of complex wideband haptic feedback. This type of haptics encoding has the disadvantage of being difficult to modify once encoded due to the inability to access the primitives used to create the signal.

— Descriptive: This representation is used to encode haptic signals as a combination of functions to be synthesized. Examples of such vectorized formats include AHAP and IVS. These formats have the advantage of being created with a composition of primitives. They are easily modifiable at runtime by an application and by dedicated editing tools. Currently, these solutions support only vibrotactile perception, but can easily be extended for other forms of haptics such as kinaesthetic, temperature and textures. They also tend to be memory inefficient with increasing signal complexity and cannot encode non-periodic phenomena

This document describes the coded representation allowing to encode both descriptive and quantized data in a human readable JSON format (.hjif) used as an exchange format. This format can be compressed and packetized into a binary file format for distribution and streaming purposes (.hmpg).

Information technology — Coded representation of immersive media —

Part 31: Haptics coding

1 Scope

This document specifies technology that supports the efficient transmission and rendering of haptic signals for the playback of immersive experiences in a wide variety of scenarios. The document describes in detail a robust coded representation of haptic media covering the two most popular haptic perceptions leveraged by devices today: vibrotactile and kinaesthetic. Support for other haptic modalities has also been integrated.

The coded representation allows to encode both descriptive and quantized data in a human readable JSON format used for exchange purposes, and a compressed bitstream version, optimised for memory usage and distribution purposes. This approach also allows to meet the expectations for compatibility with both descriptive and quantized formats, as required by the market, as well as interoperability between devices for 3D immersive experiences, mobile applications and other distribution purposes.

Information provided in this document related to the decoder is normative, while information related to the encoder and renderer is informative.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 8601-1, *Date and time — Representations for information interchange — Part 1: Basic rules*

ISO 8601-2, *Date and time — Representations for information interchange — Part 2: Extensions*

ISO/IEC 21778:2017, *Information technology — The JSON data interchange syntax*

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, available at: https://www.rfc-editor.org/info/rfc3986

IETF RFC 8259, *The JavaScript Object Notation (JSON) Data Interchange Format*, available at: https://www.rfc-editor.org/info/rfc8259

IETF RFC 4648, *The Base16, Base32, and Base64 Data Encodings*, available at: https://www.rfc-editor.org/info/rfc4648

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1.1
actuator
component of a device for rendering haptic sensations

3.1.2
avatar
body (or part of body) representation

3.1.3
band
component in a channel for containing effects for a specific range of frequencies

3.1.4
channel
component in a perception containing one or more bands rendered on a device at a specific body location

3.1.5
device
physical system having one or more actuators configured to render a haptic sensation corresponding with a given signal

3.1.6
effect
component of a band for defining a signal, consisting of a haptic waveform or one or more haptic keyframes

3.1.7
experience
top level haptic component containing perceptions and metadata

3.1.8
haptics
tactile sensations

3.1.9
keyframe
component of an effect mapping a position in time or space to an effect parameter such as amplitude or frequency

3.1.10
metadata
global information about an experience, perception, channel or band

3.1.11
MIHS format
self-contained stream for transporting MPEG-I haptic data

3.1.12
MIHS initialization unit
MIHS unit containing metadata necessary to reset and initialize a haptic decoder

3.1.13
MIHS packet
MIHS data packet which includes metadata or binary effect data

3.1.14
MIHS unit
MIHS data unit covering a duration of time

3.1.15
modality
type of haptics, such as vibration, force, pressure, position, velocity or temperature

3.1.16
perception
haptic perception containing one or more channels of a specific modality

3.1.17
signal
representation of the haptics associated with a specific modality to be rendered on a device

3.1.18
transient
short momentary effect defined by an amplitude and, directly or indirectly, a frequency

3.1.19
vector
direction in space which can be used for haptic signals spatialization

3.2 Abbreviated terms

AC	Arithmetic coding/coder
AHAP	Apple Haptic and Audio Pattern - JSON-like file format that specifies a haptic pattern
ATSC	Advanced Television Systems Committee
CDF9/7	Cohen–Daubechies–Feauveau 9/7
CRC	Cyclic redundancy check
DASH	Dynamic Adaptive Streaming over HTTP (specified in ISO/IEC 23009)
FFT	Fast Fourier Transform
HJIF	Haptics JSON Interchange Format
ID	Identifier
ISOBMFF	ISO Base media file format (specified in ISO/IEC 14496-12)
IVS	XML-based file format for representing haptic effects
JSON	JavaScript Object Notation
LOD	Level of Detail
MIHS	MPEG-I Haptic Stream
MPEG	Moving Picture Experts Group
MPEG-I	MPEG Immersive media (specified in this and other parts of ISO/IEC 23090)
OHM	Object Haptic Metadata - Text file format for haptics metadata
PCM	Pulse-code modulation
SPIHT	Set Partitioning In Hierarchical Trees

3.3 Mnemonics

The following data types are used for data specifications:

bslbf — Bit string with left bit first, where “left” is the order in which bit strings are written. Bit strings are written as a string of 1s and 0s within single quote marks, for example ‘10000001’.

vlcs8 — Variable length character string. Contains string data stored as a character array encoded in UTF-8.

uimsbf — Unsigned integer with most significant bit first.

imsbf — Integer with most significant bit first.

vlclbf — Variable length code with left bit first, where “left” refers to the order in which the variable length codes are written.

duimsbf — Decimal stored as unsigned integer with most significant bit first in a given range. Default range is [-1,1]

islif — Integer stream with left integer first, where “left” is the order in which integer strings are written. Only for buffering.

vlislif — Variable length integer stream with left integer first, where “left” refers to the order in which the integers are written.

4 Overview and architecture

4.1 Overview

This document describes a coded representation of haptics based on a data model that enables the encoding of both descriptive and quantized haptic data. Two complementary formats based on this shared data model (Clause 5) are detailed in the specifications: an interchange format (.hjif) detailed in Clause 6, and a packetized compressed binary format for streaming that can also be stored in a binary file (.hmpg) detailed in Clause 7. In addition, a normative decoder and an informative encoder and synthesizer are described in detail.

The .hjif format is a human-readable format based on JSON and is not optimized for memory usage. It can easily be parsed and manually edited, which makes it an ideal interchange format, especially when designing or creating content. For distribution purposes, the .hjif data can be compressed and packetized into a more memory efficient binary .hmpg bitstream. This compression is lossy, with different parameters impacting the encoding depth of amplitudes and frequencies composing the bitstream. The compressed and packetized data can be directly distributed as a MPEG-I haptic stream (MIHS).

4.2 Codec architecture

The generic codec architecture can process both waveform PCM signals (WAV) and descriptive haptic files such as AHAP, IVS or HJIF, the proposed MPEG format. Metadata information is provided to the codec through OHM input files (described in Annex B). An overview of the codec architecture is depicted in Figure 1, and a more detailed description of the input files and encoder architecture is provided in Clause 8.

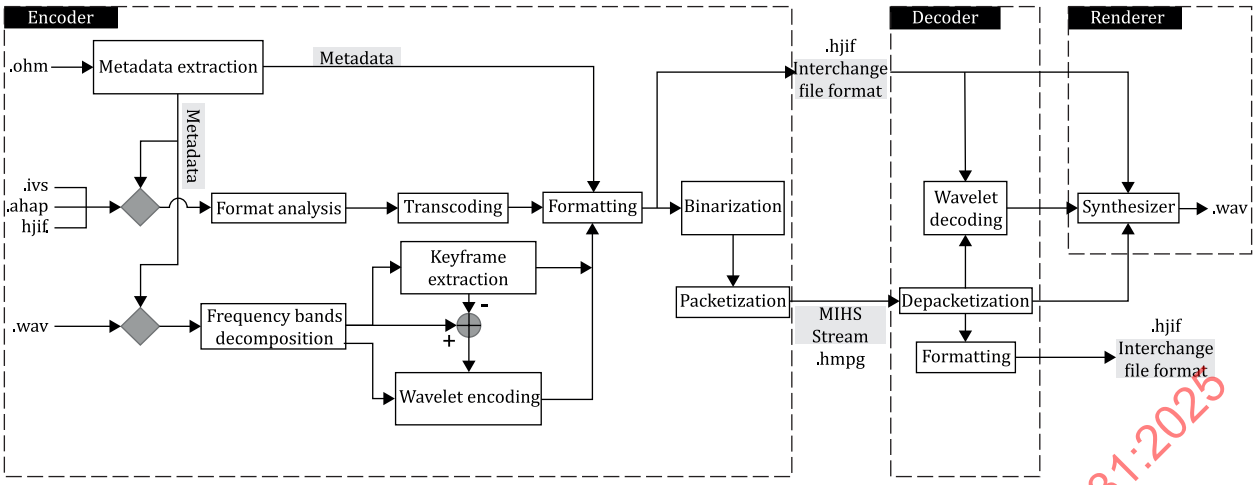


Figure 1 — Overview of the codec architecture

The encoder processes the two types of input files differently. For descriptive content, the input is analysed semantically to transcode (if necessary) the data into the proposed coded representation. For PCM content, the signal analysis process is split into two sub-processes. After performing a frequency band decomposition on the signal, the different bands (one or more) can be encoded as a set of keyframes. In Figure 1, an efficient example is given where the low frequencies are encoded using a keyframe extraction process. The low frequency band(s) is(are) then reconstructed, and the error between this signal and the original low frequency signal is computed. This residual signal is then added to the original high frequency band(s), before encoding using wavelet transforms. This multi-bands hybrid encoding scheme is detailed in Clause 8.

The encoder can output two types of formats: an interchange file format (.hjif) encoded in JSON, and a binary encoded streaming format (MIHS). that can also be stored as a binary file (.hmpg). The two formats have complementary purposes, and a lossy one-to-one conversion can be operated between them. The encoder is informative and is detailed in subclause 8.2.

The decoder takes as input a binary .hmpg file (the MIHS bitstream) or a .hjif file and either outputs a .hjif file or directly synthesizes the haptic data. Haptic data contained in a .hjif file can be rendered directly on haptic devices or using an intermediate synthesizer generating PCM data. The decoder is normative and is detailed in subclause 8.3.

The synthesizer allows to render haptic data from a .hjif input file or MIHS stream into a PCM output file. The synthesizer is informative and is detailed in subclause 8.3.4.

5 Data model

5.1 Data structure overview

This Clause focuses on the data model of the coded representation of haptics. It specifies the information required by a synthesizer to render the haptic data. The following subclauses provide detailed definitions for every property of each element of the data model depicted in Figure 2. The data structure introduced in this Clause is shared by the interchange format and the streaming format defined respectively in Clauses 6 and 7.

The data structure of the two formats follows the hierarchical organization illustrated in Figure 2.

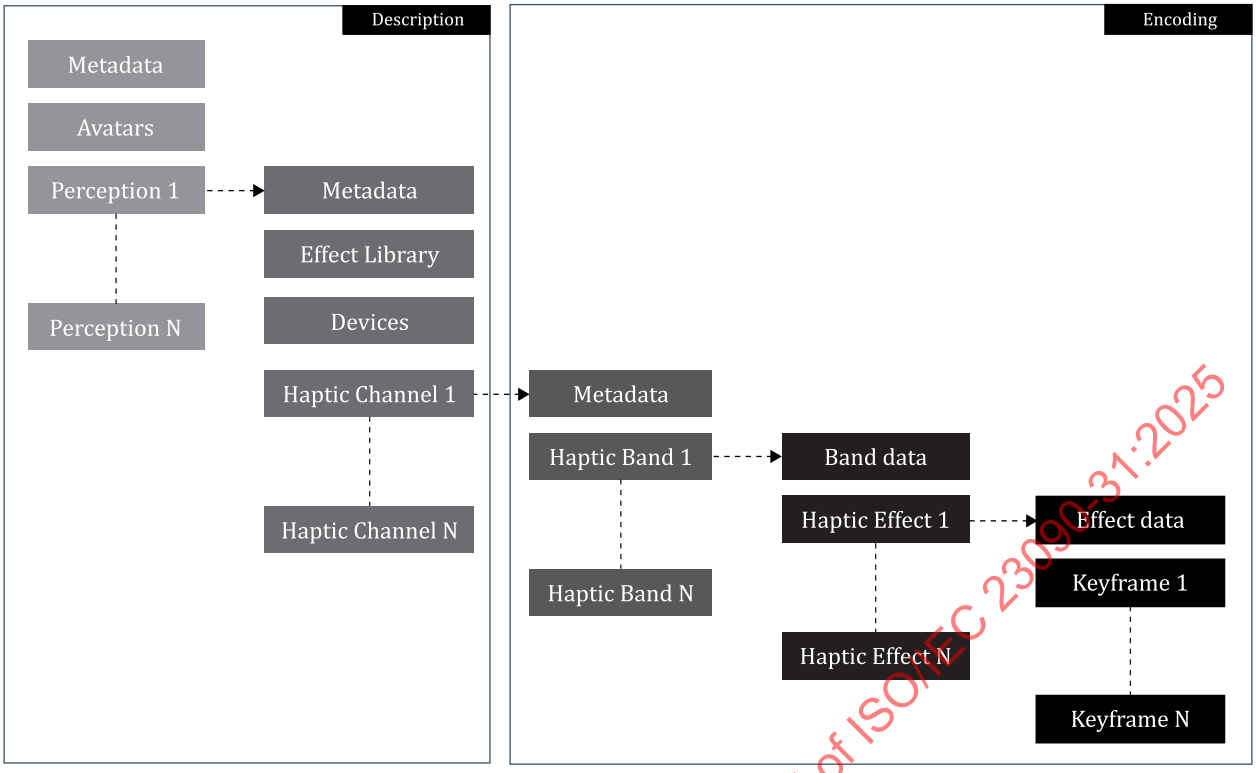
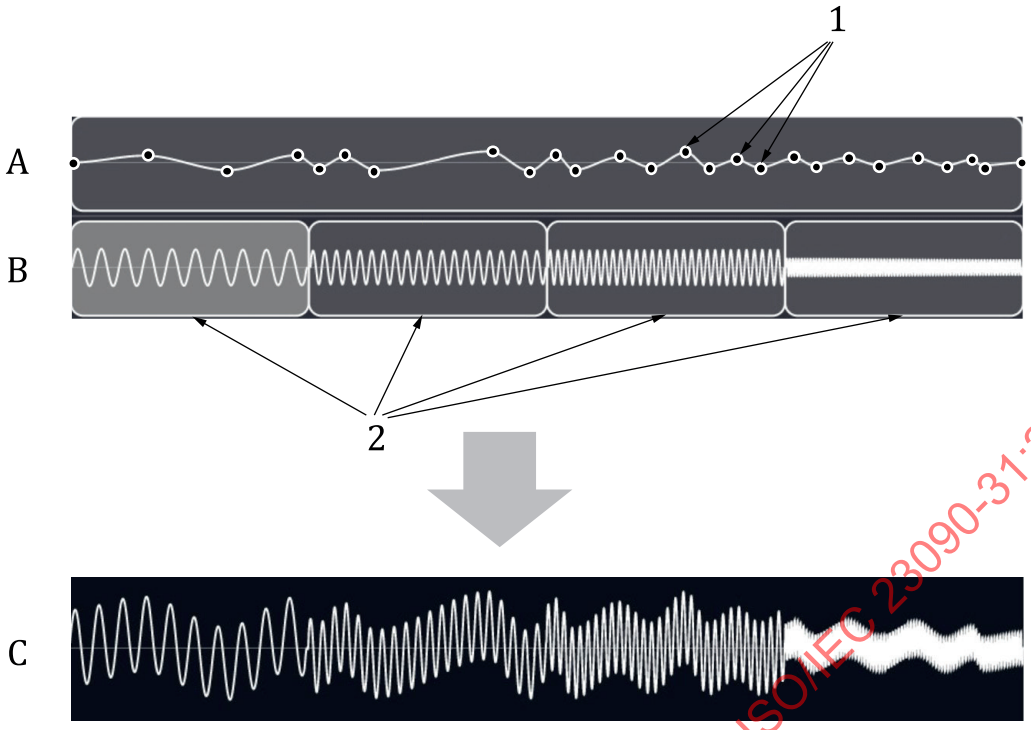


Figure 2 — General data model

The highest level of the structure describes the entire haptic experience defined in the file or stream. It contains some high-level metadata information. It also provides a list of avatars (i.e., body representation) referenced later in the file to specify the desired location of haptic stimuli on the body. Finally, the haptics are described through a list of perceptions. These perceptions correspond to haptic signals associated with specific perception modalities (e.g., vibration, force, position, velocity, temperature).

In addition to specific metadata, a perception contains a list of channels. The data in each channel are decomposed into one or more frequency bands. Each band defines part of the signal in a frequency range. A band is described by a list of haptic effects, each containing a list of keyframes. The haptic signal in a channel can then be reconstructed by combining the data in the different bands as illustrated in Figure 3. For instance, by adding the high and low frequency bands in this figure, the original signal can be reconstructed.

The perception and channel information describe the content (Description part of Figure 2), while the bands, effects and keyframes contain the data of the encoded signals (Encoding part of Figure 2).



Key

1 keyframes
2 effects
A band 1 (*curve* band)
B band 2 (vectorial *wave* band)
C channel 1

Figure 3 — Haptic channel (bottom) and its decomposition in two frequency bands (top)

The format proposes four types of bands: transient bands, curve bands, vectorial wave bands and wavelet wave bands. Each band is composed of a series of effects of the same type as the band, each defined by a list of keyframes. The data contained in the effects and keyframes are interpreted differently for different types of bands.

5.2 Haptic experience

The haptic experience defines the root of the hierarchical data model. It provides information on the date of the file, the version of the format and the encoding profile and level. It also describes the haptic experience, lists the different avatars (i.e., body representations) used throughout the experience, and defines all the haptic perceptions. Table 1 details the properties of a haptic experience.

Table 1 — Haptic experience properties

Property	Description
version	Year of the edition and amendment of ISO/IEC 23090-31 that this file conforms to, in the following format: XXXX or XXXX-Y, where XXXX is the year of publication and Y is the amendment number, if any.
profile	Name of the profile used to generate the encoded stream as defined in the normative Annex D.
level	Number of the level used to generate the encoded stream as defined in the normative Annex D.
date	Creation date of this haptic experience.
description	A user description of this haptic experience.

Table 1 *(continued)*

Property	Description
timescale	Number of ticks per second.
avatars	List of avatars defining body representations used in the haptic experience. See subclause 5.3.
perceptions	List of perceptions describing a haptic signal. See subclause 5.4.

5.3 Haptic avatar

Haptic avatars are used as body representations. Avatars can reference a custom 3D mesh from a companion file. Each perception may be associated with an avatar, which allows spatialization of effects at the channel level. The same avatar can be used by multiple perceptions.

A 3D mesh can provide resolution and accuracy with variable vertex density depending on the application. For instance, the density can be representative of the spatial acuity of a specific perception modality (Figure 4). The format of custom 3D meshes is out of the scope of this document.

Figure 4 — Example of haptic avatar body representations

Table 2 defines the list of properties of a haptic avatar:

Table 2 — Haptic avatar properties

Property	Description
id	Unique ID of the avatar in this experience.
lod	If the avatar uses a mesh with several levels of detail (LODs), this indicates which LOD should be used for the avatar.
type	Type of haptic perception represented by the avatar. This is related to the spatial acuity of the corresponding haptic modality. The avatar type may be: — Vibration: the avatar is for vibrotactile signals. — Pressure: the avatar is for pressure signals. — Temperature: the avatar is for temperature signals. — Custom: the avatar is a custom avatar. The mesh is provided as a companion file using the mesh URI. The definition of custom mesh is out of the scope of this document.
mesh	URI to access the associated 3D mesh file. The URI conforms to the syntax defined in RFC 3986.

5.4 Haptic perception

A haptic perception is a haptic signal associated with a specific perception modality. The format supports different types of modalities, encoded in function of time (pressure, acceleration, velocity, position, temperature, vibrotactile, water, wind, force, electrotactile) or space (vibrotactile texture, stiffness, friction). The list of supported modalities is provided in Table 4 with the corresponding units and type (aka temporal or spatial).

There is no accepted psychophysical representation set of tactile perception in function of a definite representation space and stimulus. Touch shows remarkable complexity in the relations between physical stimuli and the semantic assigned by humans.

In the case of haptics (touch stimulus mediated by technology) this is somewhat different as haptics is mediated by actuators which have a well-defined set of physical capabilities and representation spaces.

The concept of haptic perception in this specification proposes a simple construct which can be easily understood by implementers and designers alike tying haptic device capabilities to the most similar tactile perception, or at least the one that has most in common with the haptic representation.

For each haptic perception, metadata information is provided on the modality, the corresponding avatar representation and technical characteristics of targeted or compatible haptic devices.

The data associated with a perception may contain multiple channels. A channel is associated with a body location and usually corresponds to a haptic device; for instance, a vibrotactile suit with 16 channels corresponding to 16 vibrotactile actuators or a gamepad with one channel corresponding to a force feedback trigger.

A haptic perception may also contain an effect library.

Table 3 describes the full list of properties of a haptic perception.

Table 3 — Haptic perception properties

Property	Description
id	Unique ID of the perception in this experience.
perception modality	Type of perception modality of the haptic signal. The detailed list of perception modalities and the associated units is given in Table 4.
description	Description of the haptic data contained in the perception.
priority	Importance of the perception for scalability. A lower value indicates higher priority. Given a limited bandwidth, decoders may ignore perception with a higher priority value.
avatar id	Unique ID of the associated avatar.

Table 3 *(continued)*

Property	Description
effect library	List of haptic effects as defined in subclause 5.8. Each effect from the library has an ID and may be referenced directly in a band (see subclause 5.7).
semantic scheme	Scheme URN of the semantic description of effects. The default scheme is identified with `urn:mpeg:mpegi:haptics:effectsemantic:2023` and is defined in Semantic keyword is a two-layers hierarchical metadata structure illustrated in Table 13. It may be added as a supplementary information to an effect. At most only one layer 1 and one layer 2 semantic keywords shall be included with one effect. These keywords indicate the desired designer intention. When a presentation engine is incapable of rendering the exact specified effect it may decide to render a similar effect or not render at all. One example is a gunshot designed for a VR controller with a specific frequency range. A VR gun prop may be developed with its own haptics library. Table 13
reference devices	List of targeted haptic reference devices or actuators used for this haptic perception. More details on reference devices are given in subclause 5.5.
channels	List of haptic channels comprising this perception. More details on haptic channels are given in subclause 5.6.
unit exponent	Exponent of the powers of 10 for the SI unit of effect positions and keyframe positions of spatial modalities. This property specifies which measurement unit is used to encode the given perception. By default, the considered value is −3 (i.e. millimeters).
perception unit exponent	Exponent of the powers of 10 for the SI unit measure of the dependent variable. This property specifies which measurement unit is used to output the given perception. By default, the considered value is 0. For example, if the perception modality is set to stiffness and this exponent is set to 0, the perception experience is encoded in Newton.

Table 4 — Perception modalities and corresponding units

Modality	Perception unit	SI unit	Temporal (T)/Spatial(S)
Pressure	Pa	s	T
Acceleration	m/s^2	s	T
Velocity	m/s	s	T
Position	m	s	T
Temperature	°C	s	T
Vibrotactile	Normalized to [−1,1]	s	T
Water	m^3	s	T
Wind	m/s	s	T
Force	N	s	T
Electrotactile	Normalized to [−1,1]	s	T
Vibrotactile Texture	Normalized to [−1,1]	m	S
Stiffness	N/m	m	S
Friction	Normalized to [−1,1]	m	S
Humidity	Normalized to -1/1	s	T
User-defined Temporal	Normalized to [−1,1]	s	T
User-defined Spatial	Normalized to [−1,1]	m	S

The haptic library defined at the perception level allows to define a set of shared haptic effects at the perception level. It consists of a list of effects that can be referenced from a band. This allows to avoid unnecessary repetition of identical effects. The effect library is particularly useful for content creators and avoids the duplication of effects inside channels.

As illustrated in Figure 5, the effect library can be used to reduce the memory footprint of haptic data. The same experience can be stored in an optimized manner using a single haptic effect in the library. It is then referenced multiple times at the band level versus storing this effect multiple times directly at the band level.

As described in Figure 5, factoring and linearization can be used to convert from a library-based stream to a simple repetitive-based stream of haptic effects.

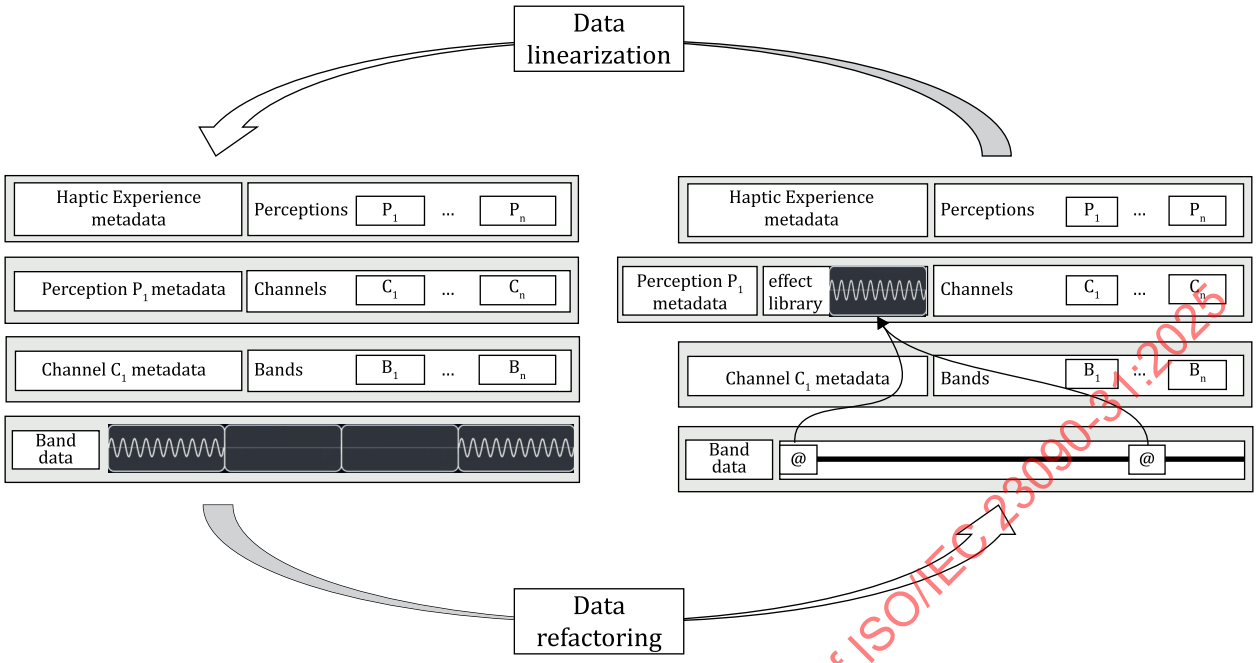


Figure 5 — Example of haptic files containing the same data with and without using the effect library

5.5 Haptic device

A haptic experience is usually defined with a reference setup to validate the experience, or with a number of specific targeted haptic devices in mind (compatible devices). If the experience is played back on different devices with different capabilities, the associated encoded signal may have to be rendered differently. To perform such adaptation, the capabilities of the original device(s) (reference or compatible devices) are needed. For this purpose, each haptic perception defines a list of reference devices (with their detailed characteristics) and each haptic channel may reference the corresponding device.

A haptic reference device is described through a list of characteristics, including the type of device, the frequency range of the device, the maximum voltage of the device. The detailed list of properties is specified in Table 5.

Table 5 — Haptic device properties

Property	Description
id	Unique ID of the device in this experience. This value is used for referencing the device in a channel.
name	Name of the device, for the purpose of human identification.
body part mask	Binary mask specifying the location of the device or actuator on the body based on the body segmentation detailed in Table 7.
maximum frequency	Maximum frequency of the actuator (Hz).
minimum frequency	Minimum frequency of the actuator (Hz).
resonance frequency	Resonance frequency of the actuator (Hz).
maximum amplitude	Maximum amplitude value of the targeted device according to the perception modality; for instance, the maximum acceleration speed if the perception modality is the acceleration. The corresponding unit is specified in Table 4 in subclause 5.4.
impedance	Impedance of the actuator (Ω).
maximum voltage	Maximum voltage of the actuator (V).

Table 5 *(continued)*

Property	Description
maximum current	Maximum current of the actuator (A).
maximum displacement	Maximum displacement of the actuator (mm).
weight	Mass of the device (kg).
size	Bounding sphere diameter of the device (mm).
custom	User-defined data. This parameter may be used to specify additional properties of the targeted device.
type	Type of actuator. Possible types are: — LRA — VCA — ERM — Piezo — Unknown (for other modalities)

5.6 Haptic channel

5.6.1 General

Haptic signals can be encoded on multiple channels. Typically, a haptic channel defines a signal to be rendered at a specific body location with a dedicated actuator or device. Metadata stored at the channel level includes information such as the gain associated with the channel, the mixing coefficient, the desired body location of the haptic feedback, optional reference device and optional direction. Additional information such as the desired sampling frequency or sample count can also be provided. Finally, the haptic data of a channel are contained in a set of haptic bands defined by their frequency range.

The list of properties of a haptic channel is detailed in Table 6.

Table 6 — Haptic channel properties

Property	Description
id	Unique ID of the channel. The identifier is unique among all channels in the perception containing this channel.
description	Description of the channel.
priority	Importance of the channel for scalability. A lower value indicates higher priority. Given a limited bandwidth, decoders may ignore perception with a higher priority value.
reference device id	Targeted reference device from the list defined in the perception.
gain	Gain associated with the channel to adapt the normalized encoded data values to a typical device, according to: $V = gain*x$, where x corresponds to the normalized encoded data.
mixing coefficient	Weight of the channel when mixing different channels together to produce a mixed signal. The resulting signal is given by: $V = \frac{\sum V_i * coefficient_i}{\sum coefficient_i}$, where V_i corresponds to the signal of channel i. A mixing coefficient of 0 indicates that the channel is not mixed.

Table 6 *(continued)*

Property	Description
body part mask	Binary mask specifying the location of the effect on the body as defined in Table 7. The binary mask 0x0 indicates that the body part is not specified. The application can render the effect anywhere. The mask 0xFFFFFFFF corresponds to the full body. It means that the effect is applied on the whole body; for instance, a background effect.
actuator resolution	Reference actuator resolution used to design the haptic experience. This value linked to body part target and actuator target can be used together as an experience spatialization model on the human body.
body part target	Identification of a unique body part or group of body parts on the human body. Table 9 describes all the possible values which can be stored here to construct the targeting command.
actuator target	List of different coordinates to target actuators on the previously identified human body parts.
frequency sampling	Sampling frequency of the original encoded signal (Hz). This may be used by the synthesizer to reconstruct the original signal. However, the synthesizer may sample the output signal at another sampling frequency.
sample count	Present if the frequency sampling value is greater than zero. It is the number of samples of the original encoded signal. This can be used along with the frequency sampling by the synthesizer to ensure that the output signal has the same size and duration as the original file.
vertices	List of the vertices from the avatar impacted by the effect. More precisely, it is the list of indices of the vertices from the mesh associated with the avatar of the perception. If the avatar does not specify a mesh, this field should be ignored. The vertices impacted by the effects of this channel are the body locations where the effects should be applied. The appropriate avatar representation is referenced by the avatar ID indicated at the perception level (see subclause 5.4).
bands	List of haptic bands comprising the channel. A channel can include one or several bands. A band corresponds to a frequency bandwidth as specified in subclause 5.7. If the bands array is empty, it corresponds to a channel without any haptic effect. As illustrated in Figure 3, the haptic signal of a channel is the sum of the signals in each band.
direction	Specifies a spatial direction for the channel as detailed in subclause 5.6.3. This direction metadata should only be used with haptic modalities dependent on space (i.e. Vibrotactile Texture, Stiffness and Friction). It indicates a preferred rendering direction of the haptic perception of the targeted body part. It can be composed with X, Y and Z following the formalism for unit vectors to indicate any direction in the 3D space. Each integer value stored in this vector will be transformed from its initial range [–127,127] to the [–1,1] range to interpret this vector as unitary.

The direction property identifies the preferred rendering direction of the spatialized haptic experience. The direction is related to the relative position of the body part in its own coordinate system. The information is required to specify the rendering of a haptic effect in a predetermined direction.

The spatialization of the haptic stimuli associated with a channel on the body can be specified either by referencing a set of vertices (with the vertices property) from a custom avatar mesh (Figure 4), or using specific localization maps. Two types of maps are defined, using a body part mask based on the body segmentation illustrated in Figure 6 with 32 defined parts, or using a semantic body parts identification combined with a map of actuators on these body parts (Figure 7). The body part mask provides a lightweight method to define a rough localization of the effects on the user's body. It does not specify how to map the effect to the device(s), which is left to the device manufacturer. The intent is to provide a simple and low data-rate mechanism to localize effects. Alternatively, the body part target is a more complex representation providing both localization and mapping to device(s). The hierarchical body representation is related to the human skeleton, while the mask only relates to the surface of the body. An additional actuator representation provides the localization and resolution of the device actuator(s) and allows interpolation, when necessary, to adapt the effect to the device configuration (as described in Annex C). Multiple methods can be combined since each addresses different goals and usages.

5.6.2 Custom mesh avatar

Consists of a set of 3D vertices, with their 3D coordinates such as Figure 4.

5.6.3 Body part mask

The mask has been defined based on Figure 6 segmentation of the human body.

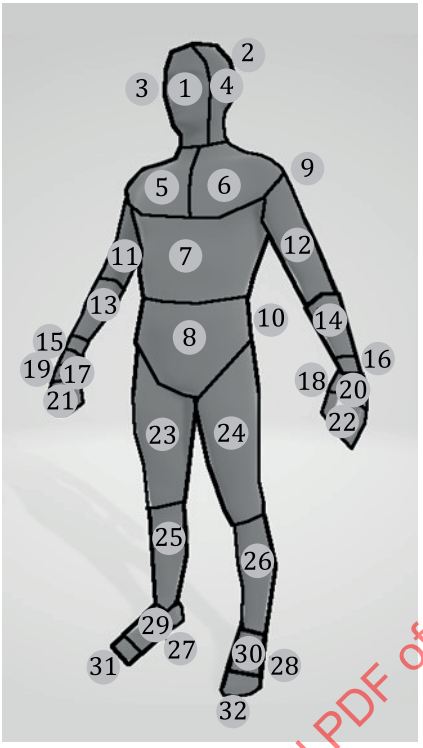


Figure 6 — Body part segmentation

The 32 body parts and their corresponding masks are detailed in Table 7. These binary masks can be logically combined to associate haptics stimuli to multiple body parts. Table 8 gives some common body parts combinations.

Table 7 — Body part masks

	Name	Body_part_mask (binary)	Hexadecimal	Decimal
0	Unspecified	00000000000000000000000000000000	0x00000000	0
1	Head front	00000000000000000000000000000001	0x00000001	1
2	Head back	00000000000000000000000000000010	0x00000002	2
3	Head right	00000000000000000000000000000100	0x00000004	4
4	Head left	00000000000000000000000000001000	0x00000008	8
5	Right upper chest	00000000000000000000000000010000	0x00000010	16
6	Left upper chest	00000000000000000000000000100000	0x00000020	32
7	Abdomen	00000000000000000000000001000000	0x00000040	64
8	Waist	00000000000000000000000010000000	0x00000080	128
9	Upper back	00000000000000000000000100000000	0x00000100	256
10	Lower back	00000000000000000000001000000000	0x00000200	512
11	Right upper arm	00000000000000000000010000000000	0x00000400	1 024
12	Left upper arm	00000000000000000000100000000000	0x00000800	2 048
13	Right forearm	00000000000000000001000000000000	0x00001000	4 096
14	Left forearm	00000000000000000010000000000000	0x00002000	8 192
15	Right wrist	00000000000000000100000000000000	0x00004000	16 384
16	Left wrist	00000000000000001000000000000000	0x00008000	32 768

Table 7 *(continued)*

	Name	Body_part_mask (binary)	Hexadecimal	Decimal
17	Right hand palm	00000000000000010000000000000000	0x00010000	65 536
18	Left hand palm	00000000000000100000000000000000	0x00020000	131 072
19	Right hand dorsum	00000000000001000000000000000000	0x00040000	262 144
20	Left hand dorsum	00000000000010000000000000000000	0x00080000	524 288
21	Right hand fingers	00000000000100000000000000000000	0x00100000	1 048 576
22	Left hand fingers	00000000001000000000000000000000	0x00200000	2 097 152
23	Right thigh	00000000010000000000000000000000	0x00400000	4 194 304
24	Left thigh	00000000100000000000000000000000	0x00800000	8 388 608
25	Right calf	00000001000000000000000000000000	0x01000000	16 777 216
26	Left calf	00000010000000000000000000000000	0x02000000	33 554 432
27	Right foot palm	00000100000000000000000000000000	0x04000000	67 108 864
28	Left foot palm	00001000000000000000000000000000	0x08000000	134 217 728
29	Right foot dorsum	00010000000000000000000000000000	0x10000000	268 435 456
30	Left foot dorsum	00100000000000000000000000000000	0x20000000	536 870 912
31	Right foot fingers	01000000000000000000000000000000	0x40000000	1 073 741 824
32	Left foot fingers	10000000000000000000000000000000	0x80000000	2 147 483 648

Table 8 — Examples of body part combinations

Name	Body_part_mask (binary)	Hexadecimal	Decimal
Right arm	00000000000101010101010000000000	0x00015540	87 360
Left arm	00000000001010101010100000000000	0x002AA800	2 795 520
Right leg	01010101010000000000000000000000	0x55400000	1 430 257 664
Left leg	10101010100000000000000000000000	0xAA800000	2 860 515 328
Upper body	00000000001111111111111111111111	0x003FFFFF	4 194 303
Lower body	11111111110000000000000000000000	0xFFC00000	4 290 772 992
Full body	11111111111111111111111111111111	0xFFFFFFFF	4 294 967 295

5.6.4 Semantic body part and actuator mapping

Alternatively, the haptic signal spatialization on the body representation can be done by a combination of body part targeting and actuator mapping on the body. This approach uses the body part target property, which is a set of successive values configured to represent a human body part or a group of body parts. The range of values is defined in Table 9:

— A Group Node is a shortcut command to target a group of body parts.

— A Locational value is an optional reference to some body part node to add precision.

— A Body part Node is a human body part which is defined hierarchically (see Figure 7).

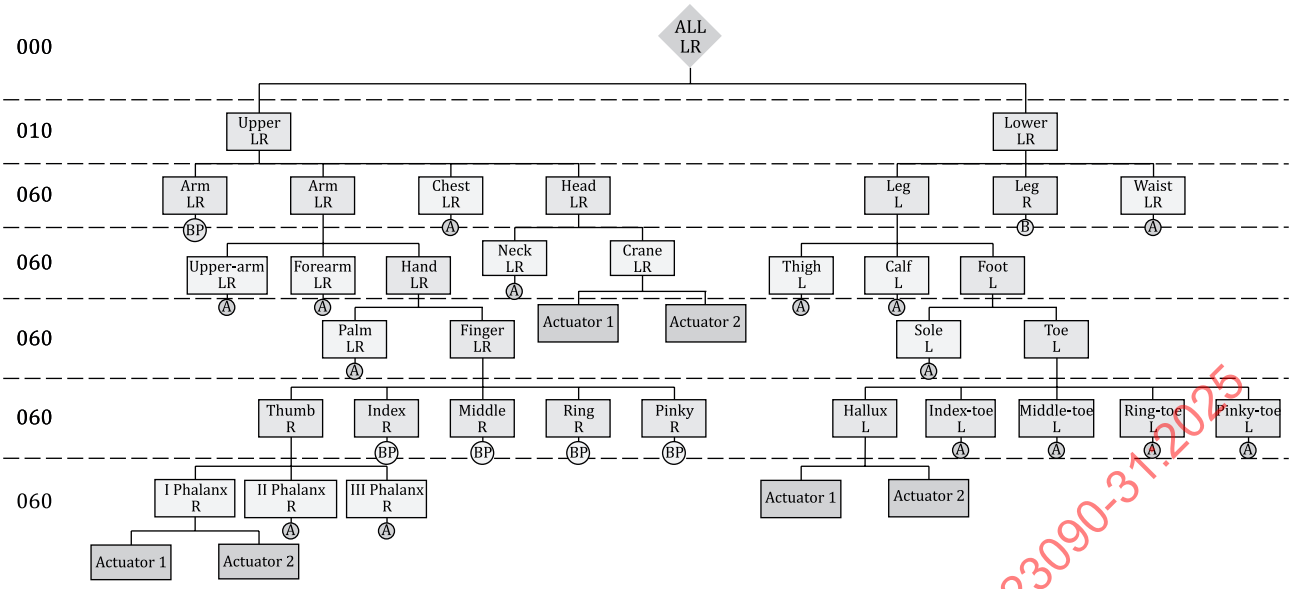


Figure 7 — Tree representation of body part target values

Table 9 — Body part target values

Name	Type	Value
Unknown	Group Node	0
All	Group Node	1
Upper	Group Node / Locational	10
Lower	Group Node / Locational	11
Right	Group Node / Locational	12
Left	Group Node / Locational	13
Front	Group Node / Locational	14
Back	Group Node / Locational	15
Arm	Body part Node	20
Head	Body part Node	21
Chest	Body part Node	22
Waist	Body part Node	23
Leg	Body part Node	24
Upper-arm	Body part Node	30
Forearm	Body part Node	31
Hand	Body part Node	32
Crane	Body part Node	33
Neck	Body part Node	34
Thigh	Body part Node	35
Calf	Body part Node	36
Foot	Body part Node	37
Palm	Body part Node	40
Finger	Body part Node	41
Sole	Body part Node	42
Toe	Body part Node	43
Thumb	Body part Node	50

Table 9 *(continued)*

Name	Type	Value
Index	Body part Node	51
Middle	Body part Node	52
Ring	Body part Node	53
Pinky	Body part Node	54
Hallux	Body part Node	55
Index-Toe	Body part Node	56
Middle-Toe	Body part Node	57
Ring-Toe	Body part Node	58
Pinky-Toe	Body part Node	59
First Phalanx	Body part Node	60
Second Phalanx	Body part Node	61
Third Phalanx	Body part Node	62
Minus	Operator	254
Plus	Operator	255

Each body part is then divided into a 3D right-handed actuator map which fulfils the following requirements:

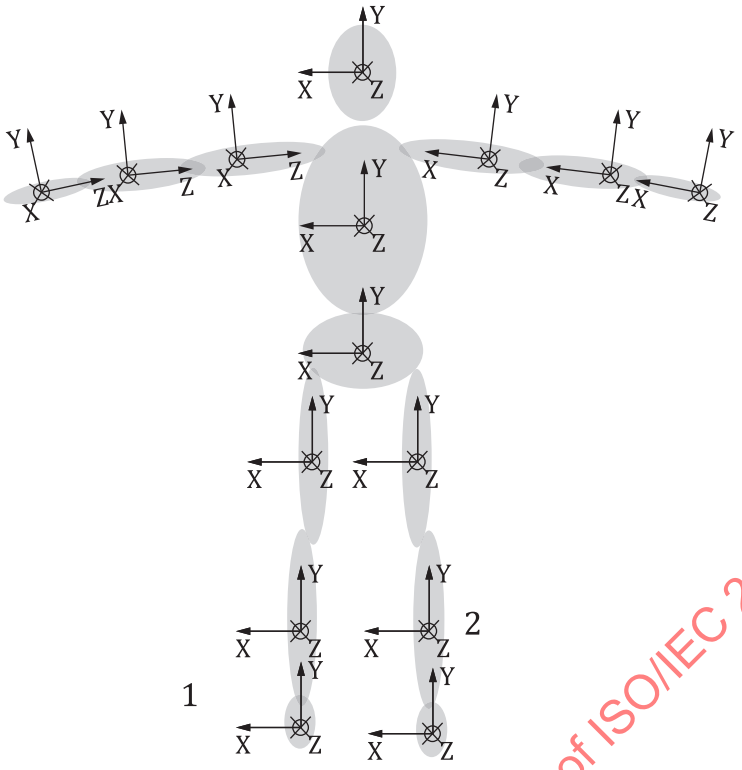
— The X axis points to the right.

— The Y axis points up.

— The Z axis points forward.

— The up direction on each body part is based on a standing human looking at his or her hand outstretched in front of himself or herself (see Figure 8 for a visual representation).

— The right direction on each body part is based on a standing human looking at his or her hand outstretched in front of himself or herself (see Figure 8 for a visual representation).



Key

1 Sole of the root downwards. Same as hand

2 Body part model Convention: Right-handed (x, y, z)

◎ towards viewer

⊠ away from viewer

Figure 8 — Representation of each body part basis for the actuator map

Each actuator can finally be targeted individually on the targeted body part to improve the precision of a haptic signal spatialization on the human body.

The actuators are mapped on the body part following a 3D mapping based on a vector [X, Y, Z]. Each value of the vector indicates the resolution in the respective 3D direction, according to the referential described in Figure 8.

For instance, for only one actuator the mapping is [1,1,1]. A resolution of [2,1,1] indicates two actuators on the X axis of the body part. A resolution of [2,2,1] indicates four actuators on the X-Y plane. A resolution of [2,2,2] indicates eight actuators arranged with two surfaces. For haptics devices which do not respect a cubic volumetric arrangement, some of the actuators may be virtual.

More details on the use of this representation are provided in Annex C.

The detailed list of properties defining a mapping vector is given in Table 10.

Table 10 — Mapping vector properties

Property	Description
X	Unit right vector. The range of possible values stored in the X axis is [–127,127].
Y	Unit up vector. The range of possible values stored in the Y axis is [–127,127].
Z	Unit forward vector. The range of possible values stored in the Z axis is [–127,127].

5.7 Haptic band

A haptic band describes the haptic signal of a channel in a given frequency range. Bands are defined by a type and a sequential list of haptic effects each containing a set of keyframes. Table 11 details the list of properties of a haptic band.

Table 11 — Haptic band properties

Property	Description
band type	Type of data contained in the band. There are four types of haptic bands: curve bands, transient bands, vectorial wave bands and wavelet wave bands. For each type of band, the information it contains has a different meaning: — Curve bands represent haptic signals with curves, described by a set of control points and a type of interpolation in-between. — Transient bands represent short momentary haptic effects, described with amplitude and frequency parameters. — Vectorial wave bands represent parametric haptic effects; described by a vector of parameters including temporal or spatial position, amplitude and frequency. The model allows both amplitude and frequency modulation of the signal. — Wavelet wave bands represent haptic effects encoded with wavelet transform decomposition, quantization, binary tree structure, and entropy coding. Subclause 5.9 details precisely how the data contained in keyframes is interpreted depending on the type of bands.
priority	Importance of the band for scalability. A lower value indicates higher priority. Given a limited bandwidth, decoders may ignore perception with a higher priority value.
Curve type	Present only for curve bands. This specifies the interpolation method to be used to synthetize the haptic signal of the band. Possible values are: — Linear — Cubic — Akima — Bézier — B-spline — Unknown (for application specific functions)
block length	This property is only present for wavelet wave bands. It is the duration of a wavelet block.
Lower frequency limit	Lower frequency limit of the band (Hz).
Upper frequency limit	Upper frequency limit of the band (Hz).
Effects	List of haptic effects as defined in 5.8. If the effect list is empty, the band does not contain haptic data.

Figure 9 illustrates two types of haptic bands: a curve band and a transient band). Points in the figure represent keyframes.

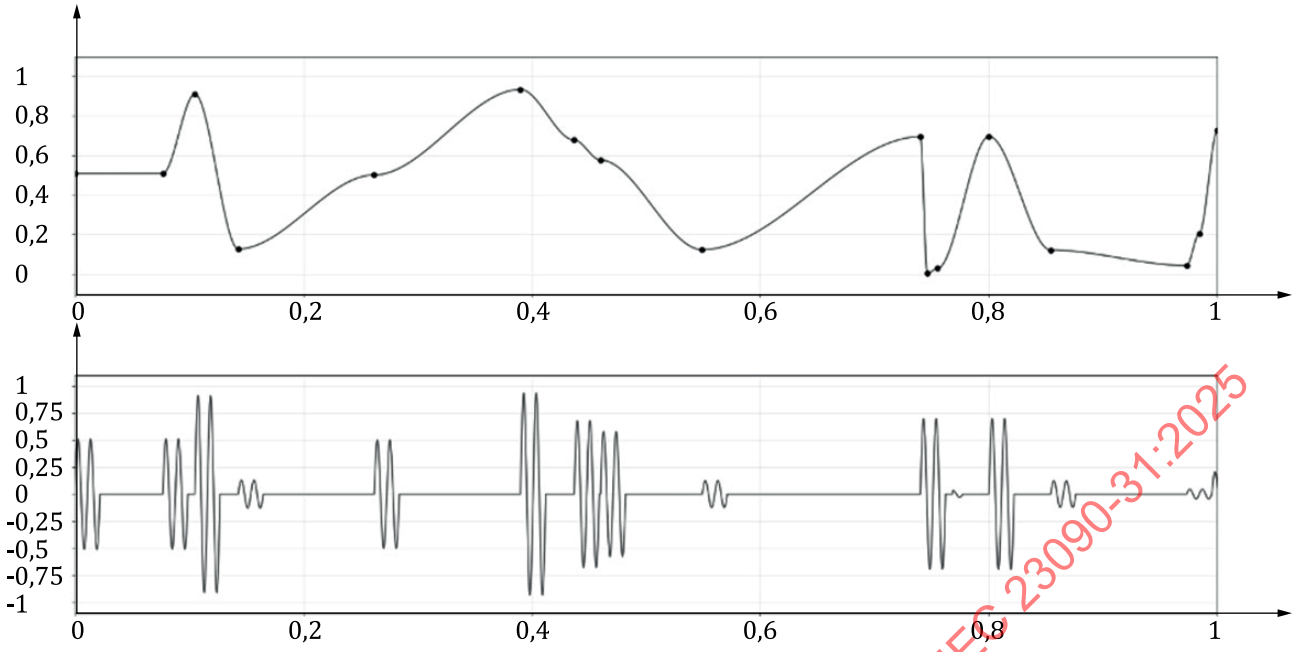


Figure 9 — Representation of a curve band (top) and a transient band (bottom)

For curve bands, different types of curve representations and interpolations can be used. The format currently supports five types of curves: linear, cubic, Akima, Bézier and B-spline.

— Linear

Each keyframe is a point in the curve and intermediate values are computed using linear interpolation.

— Cubic

Each keyframe is a point in the curve and intermediate values are computed using piecewise cubic polynomials.

— Akima

Each keyframe is a point in the curve, and intermediate values are computed using Akima interpolation. The Akima interpolation is a continuously differentiable sub-spline interpolation. It is built from piecewise third order polynomials. Only data from the next neighbour points are used to determine the coefficients of the interpolation polynomial.

For a set of data points:

$$s_i = s(x_i), 1 \le i \le k$$

where the interpolation function is defined as:

$$s(x) = a_0 + a_1.(x - x_i) + a_2.(x - x_i)^2 + a_3.(x - x_i)^3, x_i \le x \le x_{i+1}$$

To determine the coefficients a_0, a_1, a_2 and a_3 of the polynomial interpolation for each interval $[x_i, x_{i+1}]$, the function values s_i and s_{i+1}, and the first derivatives s'_i and s'_{i+1} at the end points of the interval are used.

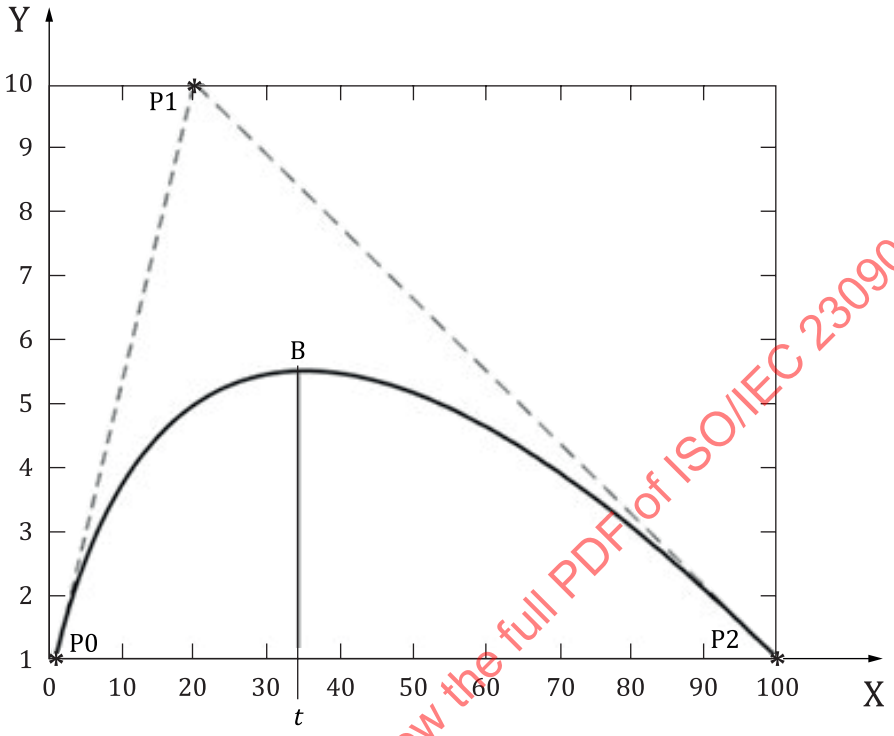
— Bézier

Bézier curves are parametric curves defined by a set of control points. Bézier curves are generally used to model smooth curves that can be scaled indefinitely. It is defined by a set of control points *P*0

through *Pn*, where *n* is called the order of the curve (*n* = 1 for linear, 2 for quadratic, 3 for cubic, etc.). The interpolation function of a quadratic Bézier curve is given by:

$$B(t)=(1-t)^2\,P0+2(1-t)tP1+t^2P2\ ,\ 0\le t\le 1$$

Where $B(t)$ is the interpolation for a parameter t (between 0 and 1), and ($P0$, $P1$, $P2$) are respectively the three control points used for the interpolation (see Figure 10).



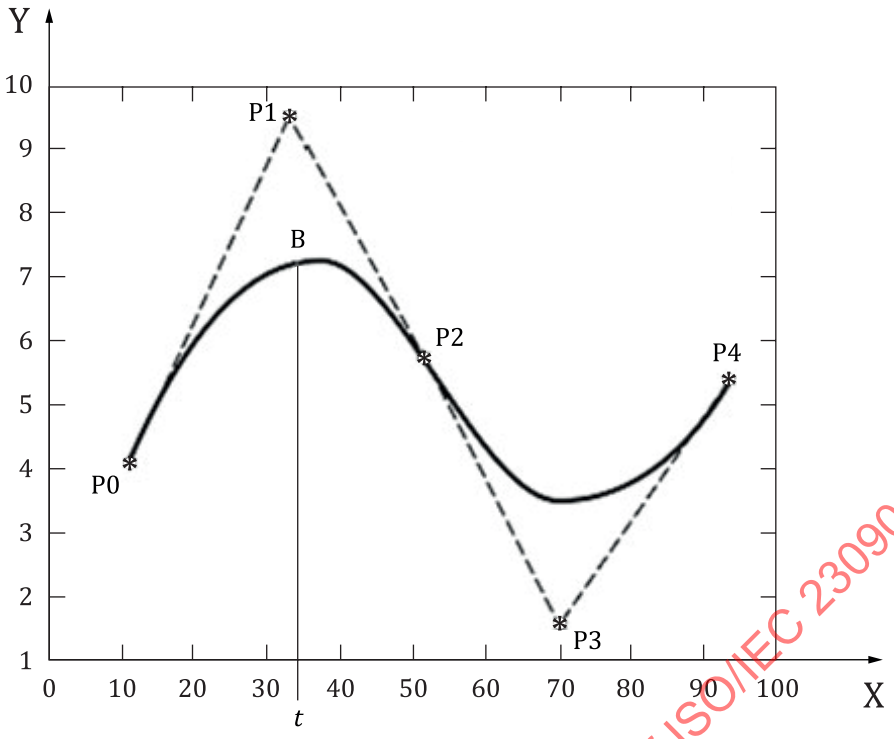
Key

X	samples
Y	amplitudes
———	bezier interpolation
*	control points

Figure 10 — Bézier curve example, with three control points

Here, the data in curve bands represent piecewise second order Bézier curves: the data are composed of consecutive quadratic Bézier curves, each defined by a set of three control points. The last control point of a Bézier curve is the first control point of the next one. This implies that the data contains at least three control points and an uneven number of control points. The control points contain 2D data defining their amplitude and timestamp. To ensure that the haptic data can be rendered properly, the control points are be ordered in time: the timestamp of each control point is higher than the timestamp of the previous point (if it exists) and lower than the timestamp of the next control point (if it exists).

Figure 11 illustrates the curve produced with a set of five control points. The signal is computed using two Bézier curves with *P0*, *P1* and *P2* as control points for the first one and *P2*, *P3* and *P4* for the second one.



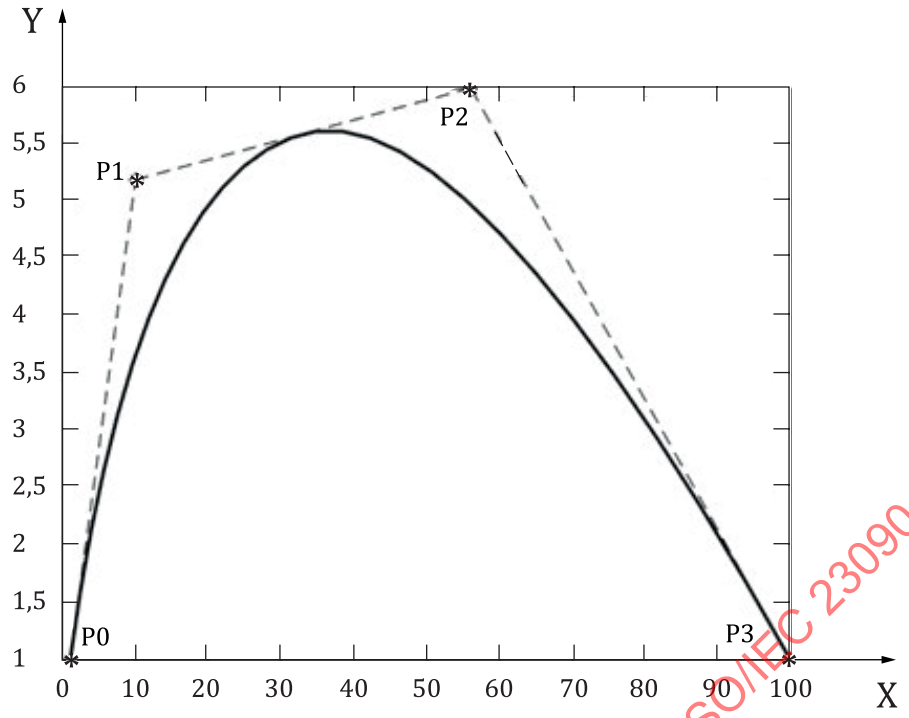
Key

X	samples
Y	amplitudes
———	bezier interpolation
*	control points

Figure 11 — Example of piecewise Bézier curve defined by five control points

— B-spline

A B-spline function is a combination of flexible segments controlled by a number of control points (*Pi*) creating smooth curves (see Figure 12). It is a generalization of Bézier curves, with higher interpolation accuracy. The shape of the curve only depends on the position of the control points. Modifying one control point will only change the curve locally. Here the data in the curve bands represent the control points of the B-spline.



Key

X	samples
Y	amplitudes
———	bezier interpolation
*	control points

Figure 12 — Example of B-spline defined with four control points

5.8 Haptic effect

For every type of haptic band, haptic effects are defined with at least a position and a type. Depending on the type of band and the type of effect, additional properties is specified, including the phase, the base signal, a composition and a number of consecutive haptic keyframes describing the effect.

The detailed list of properties of a haptic effect is given in Table 12.

Table 12 — Haptic effect properties

Property	Description
Id	Unique ID of the effect. This property is required for effects defined at the perception level in the library and for reference effects. For reference effects, this property corresponds to the ID of the library effect being referenced. The ID is unique among all effect IDs in this haptic experience.
Effect type	Type of haptic effect. Possible values are: — Basis: Effect containing signal data defined through a set of keyframes. — Composite: Effect composed of a set of other effects defined in the composition property. This type of effect does not directly contain keyframes. — Reference: Effect referencing an effect in the library. This type of effect is used at the band level to reference effects defined at the perception level in the effect library using a unique ID.
Semantic keywords	Semantic keywords of the haptic effect. It is an optional field that may be added to an effect.
Position	Indicates the temporal or spatial position of the effect. The value 0 corresponds to the starting position of the experience. The default unit for temporal haptic feedback will be milliseconds while it will be millimetres for spatial haptic feedback. Temporal or spatial is defined by the perceptionModality.
Phase	Indicates the phase of the effect in radian in the range [0,2π]. This value is used only for vectorial wave bands and is ignored in effects stored inside a band of a different type.
Base signal	This property is only used for vectorial wave bands. It defines the type of waveform signal to be used by the synthesizer. Possible values are: — Sine — Square — Triangle — Sawtooth up — Sawtooth down
composition	This property can only be used with composite effects. It contains a sub-list of effects.
keyframes	List of keyframes as defined in 5.9. The keyframes list is only used for basis effects.
Wavelet stream	Encoded wavelet stream.

The duration of a haptic effect is given, depending on the band type, as:

— Transient band: the position of the latest keyframe plus the duration of a transient which can differ among different implementations.

— Curve band: the position of the latest keyframe

— Vectorial wave band: the position of the latest keyframe

— Wavelet wave band: For this type of band, each effect corresponds to a single block. The duration of an effect then corresponds to the block length property specified at the band level.

Semantic keyword is a two-layers hierarchical metadata structure illustrated in Table 13. It may be added as a supplementary information to an effect. At most only one layer 1 and one layer 2 semantic keywords shall be included with one effect. These keywords indicate the desired designer intention. When a presentation engine is incapable of rendering the exact specified effect it may decide to render a similar effect or not render at all. One example is a gunshot designed for a VR controller with a specific frequency range. A VR gun prop may be developed with its own haptics library.

Table 13 — Two-layer semantic keywords

Layer 1	Layer 2
UX	Undefined (default)
	Click
	Double click
	Success
	Error
	Alarm
	Confirmation
	Wrong
	Ring
	Message
Avatar	Undefined (default)
	Jumping
	Fall
	Crawl
	Swim
	Collision
	Grab
	Touch
	Swip
	Footstep
Special effect	Undefined (default)
	Washout
	Noise
Weapons & Combat	Undefined (default)
	Blade
	Hit
	Hand-thrown
	Elastic propulsion
	Pneumatic
	Handguns
	Rifles
	Shotgun
	Gun
	Machinegun
	Taser
	Electric shock
	Mines
	Missile
	Grenade
	Blast
	Undefined (default)
	Wind low
	Heat
	Cold

Table 13 *(continued)*

Layer 1	Layer 2
Ambient	Rain
	Waterfall
	Water drop
	Electric buzz
	Ignition
	Cracks
	Earthquake
	Sparks
	Thunderbolt
Texture	Undefined (default)
	Rock
	Gravel
	Sand
	Wood
	Metal
	Plastic
Vehicles	Undefined (default)
	Engine
	Doors
	Brake
	Mechanical Contraption
	Drift
	Road friction
	Brake
	Road bump
	Tires
	Air friction
Music	Undefined (default)
	Hard material
	Bouncy material
	Pucking
	Bowing
	Dtriking
	Brass instruments
	Woodwind instruments

5.9 Haptic keyframe

A haptic keyframe encodes a point or sample at a given position in time or space.

Table 14 — Haptic keyframe properties

Property	Description
amplitude modulation	Amplitude of the keyframe. For vibrotactile signals, a negative vibration refers to an opposite polarity of the modulated signal.
Frequency modulation	Relative frequency of the keyframe.
Relative position	Relative position of the keyframe.

The type of band indicates how a keyframe should be interpreted. Depending on the type of band, a keyframe stores one or several of the properties defined in Table 14. The different interpretation of these values based on band type is detailed in table Table 15.

Table 15 — Keyframe interpretation for each band type

Band type	Keyframe interpretation
transient	Keyframes define with a position, an amplitude and a frequency.
Curve	Keyframes define a position and an amplitude. The keyframes represent the control points of the curve. The type of curve is specified at the band level and is used to compute the signal as detailed in subclause 5.7.
vectorial wave	Keyframes define a position, an optional amplitude modulation and an optional frequency modulation. A keyframe contains at least one of these two optional parameters. Any interpolation between keyframes can be used.

The combination of multiple of these primitive keyframes allows to describe a full haptic effect.

6 Interchange file format

6.1 Overview

The interchange format is a JSON implementation of the data model. It is not memory-optimized, but it is human-readable and can be manually edited. The data structure of this format is illustrated in Figure 13.

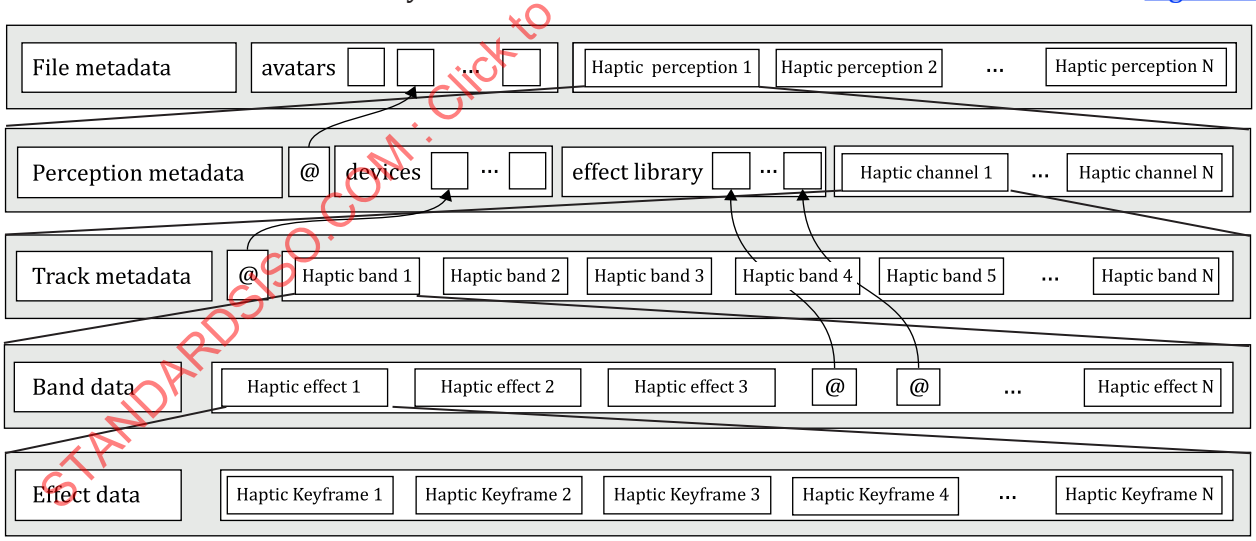


Figure 13 — Detailed data structure of the interchange format

As a JSON based format, it shall be compliant with ISO/IEC 21778 and RFC 8259. Each component of the data structure is defined as a JSON object and described below using the JSON Schema specification mechanism. The interchange format is described through the following list of JSON schemas. All the Schemas are provided in the normative Annex A

Table 16 — JSON schemas describing the interchange format

Name	Brief Description	Subclause
MPEG_haptics	Describes the haptic experience. This refers to subclause 5.2 of the data model.	6.2.1
MPEG_haptics_avatar	Describes a haptic avatar. This refers to subclause 5.3 of the data model. A haptic experience may include several avatars.	6.2.2
MPEG_haptics_perception	Describes a haptic signal associated with a specific perception modality (vibration, force, temperature, ...). This refers to subclause 5.4 of the data model. A haptic experience may include several modalities.	6.2.3
MPEG_haptics_reference_device	Provides a list of characteristics associated with a targeted haptic device. This refers to subclause 5.5 of the data model. A haptic experience may include several reference devices.	6.2.4
MPEG_haptics_channel	Describes a channel within a perception. Each perception may contain multiple channels. This refers to subclause 5.6 of the data model.	6.2.6
MPEG_haptics_vector	Describes the local direction on which the signal should be defined. This information is relevant for spatial based haptic signals. This refers to subclause 5.6.3 of the data model.	6.2.7
MPEG_haptics_band	Describes a frequency band within a haptic channel. This refers to subclause 5.7 of the data model. A haptic channel may contain multiple bands.	6.2.8
MPEG_haptics_effect	Describes a haptic effect within a frequency band. This refers to subclause 5.8 of the data model. A haptic band may contain multiple effects.	6.2.9
MPEG_haptics_keyframe	Describes a keyframe within a haptic effect. Properties contained in a keyframe are interpreted differently depending on the type of haptic band and the specified encoding modality. This refers to subclause 5.9 of the data model.	6.2.10

The following subclause details the specifications of the interchange format. Each component of the data structure is defined as JSON object with a list of properties corresponding to its attributes. The following subclause give a detailed description of the JSON objects and associated schemas as specified in Table 16.

The following subclauses use types defined in Table 17 to describe properties.

Table 17 — Descriptions of the types used in the following subclauses

string	The string type is used for strings of text. It may contain Unicode characters.
Integer	The integer type is used for integral numbers
number	The number type is used for any numeric type, either integers or floating-point numbers.
Boolean	The boolean type matches only two special values: “true” and “false”.
Array	Arrays are used for ordered lists of elements.
Object	Objects are the mapping type in JSON. They map “keys” to “values”. In JSON, the “keys” are always strings. Each of these pairs is conventionally referred to as a “property”. The properties (key-value pairs) on an object are defined using the “properties” keyword.
Enum	The enum keyword is used to restrict a value to a fixed set of values.

6.2 HJIF Specifications

6.2.1 MPEG_haptics

The MPEG_haptics element is the highest level of the format and describes the global haptic experience as defined in subclause 5.2. It includes some metadata such as the date of the file and the version of the format, provides a description of the haptic experience, specifies the shape associated with the haptic experience, lists the different avatars (i.e., body representation) and defines all the perceptions.

The MPEG_haptics metadata are necessary to configure the global experience and include all the data structures introduced above, starting with perceptions. Table 18 details the list of properties of an MPEG_haptics object corresponding to the properties defined in Table 1 of subclause 5.2.

Table 18 — Description of the MPEG_haptics object

Property	Type	Default	Description	Required
version	string	N/A	Year of the edition and amendment of ISO/IEC 23090-31 that this file conforms to, in the following format: XXXX or XXXX-Y, where XXXX is the year of publication and Y is the amendment number, if any. For this document, the value shall be "2023".	Yes
profile	string	N/A	Name of the profile used to generate the encoded stream according to the profile@level definition in the normative Annex D.	Yes
level	integer	N/A	Number of the level used to generate the encoded stream according to the profile@level definition in the normative Annex D. The value shall be equal to or greater than zero.	Yes
date	string	N/A	Indicates the creation date of this haptic experience. The date format shall conform to ISO 8601-1 and ISO 8601-2.	Yes
description	string	N/A	The user-defined description of this haptic experience.	Yes
timescale	integer	1000	Number of ticks per second. The value shall be greater than zero.	No
avatars	array<MPEG_haptics.avatar>	N/A	Provides the List of MPEG_haptics.avatar as described in 6.2.2.	Yes
perceptions	array<MPEG_haptics.perception>	N/A	Provides the List of MPEG_haptics.perception as described in 6.2.3. The array shall contain at least one element.	Yes
syncs	array <MPEG_haptics.sync>	N/A	Provides the List of MPEG_haptics.sync as described in 6.2.4.	No

6.2.2 MPEG_haptics.avatar

The MPEG_haptics.avatar element defines a haptic avatar as a body representation. It is particularly useful to specify on which part of the body the haptic effect should apply. This corresponds to subclause 5.3 of the data model.

Different types of avatars may be used for different applications or different haptic properties. A custom 3D mesh can be specified for the representation of the avatars as described in subclause 5.3. When using a custom 3D mesh representation, a reference URI to a custom mesh from a companion file shall be provided (.obj or .glTF file formats may be used).

Table 19 details the list of properties of an MPEG_haptics.avatar object corresponding to the properties defined in Table 2 of subclause 5.3.

Table 19 — Description of the MPEG_haptics.avatar object

Property	Type	Default	Description	Required
id	integer	N/A	Unique ID of the avatar. The value shall be greater than zero. The 0 value is reserved for unspecified avatars.	Yes
lod	integer	N/A	If the avatar uses a mesh with several levels of detail (LODs), this indicates which LOD to use for the avatar. The value shall be equal to or greater than zero.	Yes
type	enum<string>	N/A	Indicates the type of haptic perception represented by the avatar. Possible values are: — "Vibration" — "Pressure — "Temperature" — "Custom"	Yes
mesh	string	N/A	URI path to the custom mesh file. The URI shall conform to the syntax defined in RFC 3986.	No

6.2.3 MPEG_haptics.perception

The MPEG_haptics.perception element specifies a haptic perception with an ID, a description of the perception, some metadata information (*e.g.* perception modality and type of encoding), a reference to an avatar, a list of reference devices and the list of channels. Table 20 details the list of properties of an MPEG_haptics.perception object corresponding to the properties defined in Table 3 of subclause 5.4.

Table 20 — Description of the MPEG_haptics.perception object

Property	Type	Default	Description	Required
id	integer	N/A	Unique ID of the perception in the haptic experience. The value shall be equal to or greater than zero.	Yes
perception_	enum<string>	N/A	Indicates the type of perception as defined in 5.4. Possible values are: — "Pressure" — "Acceleration" — "Velocity" — "Position" — "Temperature" — "Vibrotactile" — "Water" — "Wind" — "Force" — "Electrotactile" — "Vibrotactile Texture" — "Stiffness" — "Friction" — "Humidity" — "User-defined Temporal" — "User-defined Spatial" — "Other"	Yes
description	string	N/A	The user-defined description of the haptic perception.	Yes
priority	integer	255	Importance of the perception for scalability. A lower value indicates higher priority. The value shall be equal to or greater than zero. Given a limited bandwidth, decoders may ignore perception with a higher priority value.	No
avatar_	integer	N/A	Unique ID of the associated avatar body model from 6.2.2. The value shall reference the id of an existing avatar from the *avatars* array defined in Table 18 or be equal to 0. The 0 value means that no avatar is not specified.	Yes
effect_	array<MPEG_haptics.effect>	N/A	List of predefined MPEG_haptics.effect as defined in 6.2.8. The list may be empty. Library effects are referenced directly in the channels.	Yes
semantic_ scheme	String	`urn:mpeg:mpegi:haptics:effectsemantic:2023`	Scheme URN of the the semantic description of effects.	No
reference_	array<MPEG_haptics.reference_devices>	N/A	List of targeted MPEG_haptics_reference.device devices or actuators as defined in 6.2.4 for this haptic perception.	No

Table 20 *(continued)*

Property	Type	Default	Description	Required
channels	array<MPEG_haptics.channel>	N/A	List of MPEG_haptics.channel as defined in 6.2.6 composing this perception. The array shall contain at least one element.	Yes
unit_exponent	integer	−3	Refers to the exponent of the powers of 10 for the SI unit of effect positions and keyframe positions of spatial modalities (see 5.4 for each input unit).	No
perception_unit_exponent	integer	0	Refers to the exponent of the powers of 10 for the SI unit measure of the dependent variable (see 5.4 for each output perception unit)	No

6.2.4 MPEG_haptics.sync

The MPEG_haptics.sync element defines a temporal sync point in the haptic presentation. Table 21 details the list of properties of an MPEG_haptics.sync object.

Table 21 — Description of the MPEG_haptics.sync object

Property	Type	Default	Description	Required
timestamp	integer	N/A	Timestamp of the haptic experience in ticks, i.e., timestamp/timescale is the timestamp in seconds. The value shall be equal to or greater than zero.	Yes
timescale	integer	1000	Number of ticks per second. The value shall be greater than zero.	No

6.2.5 MPEG_haptics.reference_device

The MPEG_haptics.reference_device specifies a targeted reference device or actuator with an ID, a name and a body location. Additional properties can be optionally specified for each device. Table 22 details the list of properties of an MPEG_haptics.reference_device object corresponding to the properties defined in Table 5 of subclause 5.5.

Table 22 — Description of the MPEG_haptics.reference_device object

Property	Type	Default	Description	Required
id	integer	N/A	Unique ID of the device in the haptic perception. The value shall be greater than zero. The 0 value is reserved for unspecified reference device.	Yes
name	string	N/A	The user-defined name of the device.	Yes
body_part_mask	integer	N/A	Binary mask specifying the location of the device or actuator on the body as defined in Table 7 of subclause. The value shall be equal to or greater than zero.	No
maximum_frequency	number	N/A	Maximum frequency of the actuator (Hz). The value shall be equal to or greater than zero.	No
minimum_frequency	number	N/A	Minimum frequency of the actuator (Hz). The value shall be equal to or greater than zero.	No
resonance_frequency	number	N/A	Resonance frequency of the actuator (Hz). The value shall be equal to or greater than zero.	No
maximum_amplitude	number	N/A	Maximum amplitude value of the targeted device according to the perception_modality. The corresponding unit is specified in Table 4 subclause 5.4. The value shall be equal to or greater than zero.	No
impedance	number	N/A	Impedance of the actuator (Ω). The value shall be equal to or greater than zero.	No
maximum_voltage	number	N/A	Maximum voltage of the actuator (V). The value shall be equal to or greater than zero.	No
maximum_current	number	N/A	Maximum current of the actuator (A). The value shall be equal to or greater than zero.	No

Table 22 *(continued)*

Property	Type	Default	Description	Required
maximum_displacement	number	N/A	Maximum displacement of the actuator (mm). The value shall be equal to or greater than zero.	No
weight	number	N/A	Weight of the device (Kg). The value shall be equal to or greater than zero.	No
size	number	N/A	Size of the device (mm). The value shall be equal to or greater than zero.	No
custom	number	N/A	User-defined data. This parameter may be used to specify additional properties of the targeted device.	No
type	enum<string>	N/A	Indicates the type of actuator. "LRA", "VCA", "ERM" or "Piezo" indicates the respective type of haptic actuator. "Unknown " indicates any other actuator type.	No

6.2.6 MPEG_haptics.channel

The MPEG_haptics.channel element specifies a channel by an ID, a description, a body part, a mixing coefficient, a gain value and a list of haptic bands. Various additional properties can also be specified, such as a reference device id, the desired sampling frequency or the sample count.

Table 23 details the list of properties of an MPEG_haptics.channel object corresponding to the properties defined in Table 6 of subclause 5.6.

Table 23 — Description of the MPEG_haptics.channel object

Property	Type	Default	Description	Required
id	integer	N/A	Unique ID of the channel in the perception containing the channel. The value shall be equal to or greater than zero.	Yes
description	string	N/A	The user-defined description of the channel.	Yes
priority	integer	255	Importance of the channel for scalability. A lower value indicates higher priority. The value shall be equal to or greater than zero. Given a limited bandwidth, decoders may ignore perception with a higher priority value.	No
reference_device_id	integer	N/A	ID of the targeted reference device or actuator from the list defined in the MPEG_haptics.perception element. The value shall reference the id of an existing reference device from the *reference_devices* array defined in Table 20 or be equal to 0. The 0 value means that no reference device is specified.	No
gain	number	N/A	Gain associated with the channel. The value shall be equal to or greater than zero.	Yes
mixing_coefficient	number	N/A	Weight of the channel used when mixing different channels together. Value 0 indicates that the channel will not be mixed. The value shall be equal to or greater than zero.	Yes
body_part_mask	integer	0	Binary mask specifying the location of the effect on the body as defined in Table 7 of subclause 5.6. The value shall be greater than or equal to zero.	No

Table 23 *(continued)*

Property	Type	Default	Description	Required
actuator_resolution	MPEG_haptic_vector	N/A	Reference actuator resolution used to design the haptic experience. The values in the vector shall be equal to or greater than zero.	No
body_part_target	array<string>	N/A	Semantic identification of a unique body part or group of body parts on the human body. All possible values are described in Table 9.	No
actuator_target	array<MPEG_haptics.vector>	N/A	List of different actuators targeted by the channel and identified by its coordinates.	No
frequency_sampling	integer	N/A	Sampling frequency of the original encoded signal (Hz). If present, the value shall be equal to or greater than zero..	No
sample_count	integer	N/A	Number of samples of the original encoded signal. If present, the value shall be equal to or greater than zero..	No
vertices	array<integer>	N/A	List of indices of the vertices from the avatar impacted by the effect. The associated avatar representation is given by the avatar_id indicated in the MPEG_haptics.perception (see subclause 6.2.3).	No
bands	array<MPEG_haptics.band>	N/A	List of haptic bands comprising the channel. A channel may include one or several bands. A band corresponds to a frequency bandwidth as specified in subclause 6.2.8. If the bands array is empty, the channel does not contain haptic data.	Yes
direction	MPEG_haptic. vector	N/A	Spatial direction for the channel, defined with an MPEG_haptics.vector (see subclause 6.2.7). This property is only used with haptic modalities dependent on the space dimension (i.e. vibrotactile texture, stiffness and friction). Each integer value stored in this vector will be transformed from its initial range [–127,127] to the [–1,1] range to interpret this vector as unitary.	No

6.2.7 MPEG_haptics.vector

When specified in the MPEG_haptics.channel, the haptic rendering system will use this spatial direction to synthesize the correct haptic feedback based on the user or object movement. Usually, the movement is provided by an appropriate tracking system.

Table 24 details the list of properties of an MPEG_haptics.vector object corresponding to the properties defined in Table 10 of subclause 5.6.3.

Table 24 — Description of the MPEH_haptics.vector object

Property	Type	Default	Description	Required
X	integer	N/A	Unit right vector. The range of possible values stored in the axis is [–127,127].	Yes
Y	integer	N/A	Unit up vector. The range of possible values stored in the axis is [–127,127].	Yes
Z	integer	N/A	Unit forward vector. The range of possible values stored in the axis is [–127,127].	Yes

6.2.8 MPEG_haptics.band

The MPEG_haptics.band element defines a haptic band with the type of the band, an interpolation function, a block length, a frequency range and a list of haptic effects.

Table 25 details the list of properties of an MPEG_haptics.band object corresponding to the properties defined in Table 11 of subclause 5.7.

Table 25 — Description of the MPEG_haptics.band object

Property	Type	Default	Description	Required
band_type	enum<string>	N/A	Indicates the type of data contained in the band as specified in subclause 5.7. Possible values are: — "WaveletWave" — "VectorialWave" — "Curve" — "Transient"	Yes
priority	integer	255	Importance of the band for scalability. A lower value indicates higher priority. The value shall be equal to or greater than zero. Given a limited bandwidth, decoders may ignore perception with a higher priority value.	No
curve_type	enum<string>	Linear	Indicates the type of interpolation function that should be used by the synthesizer as specified in subclause 5.7. Possible values are: — "Linear" — "Cubic" — "Akima" — "Bezier" — "BSpline" — "Unknown"	No
block_length	integer	N/A	Number of samples of a wavelet effect block. The duration of this block is derived from the sampling frequency of the channel. This property is required when *band_type* is equal to "WaveletWave". The value shall be greater than 16 and be a power of 2.	Conditional

Table 25 *(continued)*

Property	Type	Default	Description	Required
lower_frequency_limit	number	N/A	Lower frequency limit of the band (Hz). The value should be in the range [0,10000].	Yes
upper_frequency_limit	number	N/A	Indicates the upper frequency limit of the band (Hz). The value should be in the range [0,10000].	Yes
effects	array<MPEG_haptics_effect>	N/A	List of MPEG_haptics.effect as defined in subclause 6.2.9. If the effects array is empty, the band does not contain haptic data.	Yes

6.2.9 MPEG_haptics.effect

Each MPEG_haptics.band is composed of haptic effects defined by the MPEG_haptics.effect element with an effect type, a position, a phase, a signal type, an optional composition and a list of keyframes.

Table 26 details the list of properties of an MPEG_haptics.band object corresponding to the properties defined in Table 12 of subclause 5.8.

Table 26 — Description of the MPEG_haptics.effect object

Property	Type	Default	Description	Required
id	integer	N/A	ID of an effect. For any effect from the effect library and any “Reference” effect, the value shall be defined and be equal to or greater than zero. For all other effects the property shall not be defined. For any non-”Reference" effect from the effect library , the value shall be unique in the perception.	Conditional
effect_type	enum<string>	N/A	Indicates the type of haptic effect. Effect-type value equals one of: “Basis”, “Composite” and “Reference”, corresponding to the description in subclause 5.8.	Yes
semantic_keywords	string	N/A	Semantic keywords included with the effect.	No
position	integer	N/A	Indicates the temporal or spatial position of the effect according to the perception modalities in Table 4. In the case of temporal, position/timescale is the temporal position in seconds. The value shall be equal to or greater than zero.In a band where band_type = “WaveletWave”, this property shall only exist for the first effect (and no other). For any other band type, the property is required.	Conditional
phase	number	0	Phase of the effect for Transient and Vectorial bands. The value should be in the range [0,2π].	No
base_signal	enum<string>	Sine	Indicates the type of the waveform signal for transient and vectorial wave bands. Possible values are: — “Sine” — “Square” — “Triangle” — “SawToothUp” — “SawToothDown”	No

Table 26 *(continued)*

Property	Type	Default	Description	Required
composition	array<MPEG_haptics.effect>	N/A	This attribute can only be used with composite effects. It contains a list of effects. This type of effect does not directly contain keyframes.	No
keyframes	array<MPEG_haptics.keyframe>	N/A	List of MPEG_haptics.keyframes as defined in 6.2.10. This property is required for basis effects. If the keyframes array is empty, the effect does not contain haptic data. This property shall not exist when band_type="WaveletWave".	No
wavelet_stream	string	N/A	Base64 encoding with most significant bit first (conformant to RFC 4648) of the encoded wavelet stream. This property shall exist only when band_type="WaveletWave". The decoding process is described in 8.3.3.	Conditional

6.2.10 MPEG_haptics.keyframe

An MPEG_haptics.effect is described by a set of MPEG_haptics.keyframes. Depending on the type of the haptic band and the encoding modality, a keyframe is defined with one or more of an amplitude modulation, a frequency modulation and a relative position.

Table 27 details the list of properties of an MPEG_haptics.band object corresponding to the properties defined in Table 14 of subclause 5.9.

Table 27 — Description of the MPEG_haptics.keyframe object

Property	Type	Default	Description	Required
amplitude_modulation	number	N/A	Amplitude of the keyframe. The value shall be in the range [−1,1].	No
frequency_modulation	number	N/A	Relative frequency of the keyframe. The value shall be in the range [0,10000]	No
relative_position	integer	N/A	Relative position of the keyframe. In the case of temporal, relative_position/timescale is the relative position in seconds. The value shall be equal to or greater than zero.	No

7 MPEG-I haptic stream (MIHS) format

7.1 Overview

7.1.1 General

This subclause defines a self-contained stream format to transport MPEG-I haptic data. The transport mechanism uses a packetized approach. Different packet types are defined to transport diverse types of information.

The packets are of variable duration and include two levels of packetization:

— MIHS unit which covers a duration of time and includes zero or more MIHS packets.

— MIHS packet which includes metadata or haptic effect data.

Figure 14 shows the high-level concept of MIHS packetization.

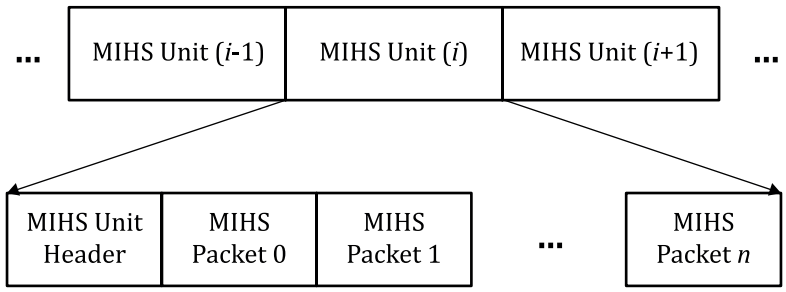


Figure 14 — MIHS packetization

Each MIHS unit covers a duration of haptic presentation time defined by a timestamp and a duration. Consecutive MIHS units shall either be temporally aligned, sharing the same timestamp and duration or be in sequence, *i.e.* a MIHS Unit starts at the end of the previous MIHS unit. The MIHS unit is followed by the next MIHS unit, unless it is the last MIHS unit of the haptic experience. All MIHS packets of a MIHS unit have the same starting time and duration of the containing MIHS unit.

An MIHS unit may have one of the following types:

— **Initialization**. An initialization unit shall contain one timing MIHS packet and may include one or more metadata MIHS packets. The initialization unit starts a new time anchor in haptics presentation time by setting the timestamp and may also set or change the timescale. The duration of an initialization unit duration shall be zero.

— **Temporal**. A temporal unit shall contain one or more MIHS packets. The duration of a temporal unit shall be a positive number.

— **Spatial**. A spatial unit shall contain one or more MIHS packets. The duration of a spatial unit shall be zero. If compatibility with the HJIF format is desired, all effects of a spatial modality need to be included in the first MIHS data unit.

— **Silent**. A silent unit indicates that there is no effect that starts during this time interval and shall not include any MIHS packets other than MHS packets of type PACTYPE_TIMING. The duration of a silent unit shall be a positive number.

The first MIHS unit in a haptic stream shall be an initialization unit.

A MIHS unit may be a sync unit or a non-sync unit:

— A sync unit resets the previous effects and therefore provides an independent haptic experience from the previous MIHS units.

— A non-sync unit is the continuation of previous MIHS units and cannot be independently decoded and rendered without decoding the previous MIHS unit(s).

The timestamp in the timing packet of a temporal or silent MIHS Unit, if exists, shall have a value consistent with the timing derived from the previous MIHS Units timings.

Two consecutive temporal MIHS units in the bitstream may be either temporally aligned or sequential in the timeline.

— If temporally aligned in the timeline, they shall have the same duration and contain timing packets with identical timestamps values. Temporally aligned MIHS Units shall not have any other MIHS Units in between.

— If sequential in the timeline, the timestamp in the timing packet of the second MIHS Unit, if exists, shall have a value consistent with the timing derived from the first MIHS Unit.

An MIHS unit comprises the unit header and MIHS packets as shown in Figure 15. Names in the figure are shortened with respect to their syntax counterparts for easier reading.

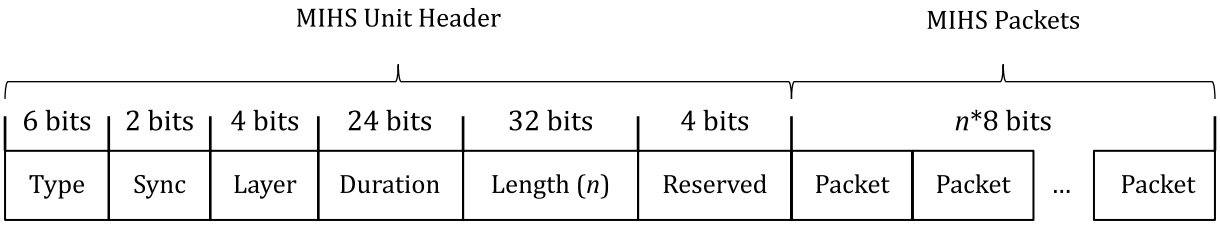


Figure 15 — MIHS unit structure

7.1.2 Initialization units

An initialization unit is a sync unit and shall have the following constraints:

- Exactly one packet of type PACTYPE_ INIT_TIMING
- Zero or more packets of the following types:
  - PACTYPE_METADATAEXPERIENCE
  - PACTYPE_METADATAPERCEPTION
  - PACTYPE_METADATACHANNEL
  - PACTYPE_METADATABAND
  - PACTYPE_LIBRARYEFFECTS
- Zero or more packets of the following types:
  - PACTYPE_CRC16
  - PACTYPE_CRC32
  - PACTYPE_GlobalCRC16
  - PACTYPE_GlobalCRC32
- No MIHS packet of the following types:
  - PACTYPE_DATA
  - PACTYPE_TIMING

The sync field shall be ignored for an initialization unit since it is a sync unit by definition.

The duration field shall be ignored for an initialization unit since it does not have a duration by definition.

7.1.3 Temporal and spatial units

Temporal and spatial units shall have the following constraints:

- One or more packets of the following types:
  - PACTYPE_DATA
- Zero or more packets of the following types:
  - PACTYPE_CRC16
  - PACTYPE_CRC32
  - PACTYPE_GlobalCRC16

  - PACTYPE_GlobalCRC32
- No packets of following types:
  - PACTYPE_INIT_TIMING
  - PACTYPE_METADATAEXPERIENCE
  - PACTYPE_METADATAPERCEPTION
  - PACTYPE_METADATACHANNEL
  - PACTYPE_METADATABAND
  - PACTYPE_LIBRARYEFFECTS

Temporal units shall have the following additional constraint:

- Zero or one packet of type PACTYPE_TIMING
- Only data for the following perception modalities:
  - Pressure
  - Acceleration
  - Velocity
  - Position
  - Temperature
  - Vibrotactile
  - Water
  - Wind
  - Force
  - Electrotactile
  - Humidity
  - User-defined Temporal
  - Other

Spatial units shall have the following additional constraint:

- No packet of type PACTYPE_TIMING
- Only data for the following perception modalities:
  - Vibrotactile texture
  - Stiffness
  - Friction
  - User-defined Spatial
  - Other

Silent units shall have the following additional constraints:

— Zero or one packet of type PACTYPE_TIMING

A perception may include temporal or spatial units but not both.

The sync field shall be ignored for a spatial unit since it is a sync unit by definition.

The duration field shall be ignored for a spatial unit since it does not have a duration by definition.

7.1.4 MIHS packets

All packets follow the structure shown in Figure 16. Names in the figure are shortened with respect to their syntax counterparts for easier reading. The packet header is common to all MIHSPacketType values. The payload structure varies based on the MIHSPacketType.

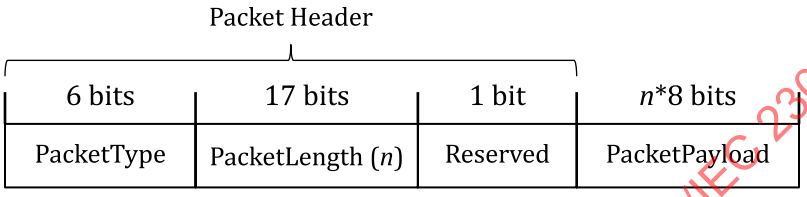


Figure 16 — MIHS packet structure

The data contained in a mpegiHapticPacket() corresponds to the data of single band. Packets contained in the same Unit are independent as they store data for different bands.

A MIHS packet shall have one of the following types:

— **InitializationTiming**. Defines a timestamp, timescale, nominalDuration, durationDeviation and overlapping flag.

— **Timing**. Defines a timestamp for subsequent MIHS packets.

— **MetadataExperience**. Contains metadata for the haptic experience.

— **MetadataPerception**. Contains metadata for a haptic perception.

— **MetdataChannel**. Contains metadata for a haptic channel.

— **MetadataBand**. Contains metadata for a haptic band.

— **Data**. Contains haptic effect data.

— **EffectLibrary**. Contains a library of haptic effects that haptic effect data may reference.

— **CRC16** or **CRC32**. Contains a CRC value which applies to the directly following MIHS packet.

— **GlobalCRC16** or **GlobalCRC32**. Contains a CRC value which applies to one or more directly following MIHS packets.

7.2 Syntax and semantics

7.2.1 mpegiHapticStream()

Table 28 details the syntax of a complete MPEG-I Haptic Stream.

Table 28 — Syntax of mpegiHapticStream()

Syntax	No. of bits	Mnemonic

mpegiHapticStream()
{
  while (bitsAvalaible() != 0) {
  mpegiHapticUnit();
  }
}		

7.2.2 mpegiHapticUnit()

Table 29 details the syntax of a MIHS Unit.

Table 29 — Syntax of mpegiHapticUnit()

Syntax	No. of bits	Mnemonic
mpegiHapticUnit() {		
MIHSUnitType;	**6**	**uimsbf**
MIHSUnitSync;	**2**	**uimsbf**
MIHSLayer	**4**	**uimsbf**
MIHSUnitDuration;	**24**	**uimsbf**
MIHSUnitLength;	**32**	**uimsbf**
reserved	**4**	**uimsbf**
while (packetsAvalaible() != 0) {		
mpegiHapticPacket();		
}		
}		

MIHSUnitType MIHS unit type. The possible values for MIHSUnitType are listed in Table 30. Decoders shall skip units with an unknown MIHSUnitType.

Table 30 — Value of MIHSUnitType

MIHSUnitType	Value
UNITTYPE_INITIALIZATION	0
UNITTYPE_TEMPORAL	1
UNITTYPE_SPATIAL	2
UNITTYPE_SILENT	3
/* reserved */	4–63

MIHSUnitSync Flag indicating whether the MIHS unit is independently interpretable. Table 31 lists the possible values for MIHSUnitSync.

Table 31 — Value of MIHSUnitSync

Value	Meaning
0	Independent
1	Dependent
2–3	/* reserved */

MIHSLayer Priority of MIHS Unit for scalability. The lower the value the more important the MIHS Unit.

MIHSUnitDuration Duration of the MIHS unit. MIHSUnitDuration shall be zero for MIHSUnitType UNITTYPE_INITIALIZATION and UNITTYPE_SPATIAL and greater than zero for MIHSUnitType UNITTYPE_TEMPORAL and UNITTYPE_SILENT. The duration is in the timescale of the readMetadataTiming() structure in the most recently sent mpegHapticUnit() with MIHSUnitType UNITTYPE_INITIALIZATION.

MIHSUnitLength Total length in bytes of the MIHS unit's MIHS packets (mpegiHapticPacket() structures).

mpegiHapticPacket() MIHS packet structure defined in subclause 7.2.3.

7.2.3 mpegiHapticPacket()

Table 32 details the syntax of a MIHS Packet.

Table 32 — Syntax of mpegiHapticPacket()

Syntax	No. of bits	Mnemonic
mpegiHapticPacket()		
{		
MIHSPacketType;	**6**	**uimsbf**
MIHSPacketLength;	**17**	**uimsbf**
reserved;	**1**	
MIHSPacketPayload(MIHSPacketType);		
ByteAlignement	**0-7**	**uimsbf**
}		

MIHSPacketType Payload type in the packet. Table 33 lists the possible values for MIHSPacketType. Subclause 7.3 provides further details. Decoders shall skip packets with an unknown MIHSPacketType.

Table 33 — Value of MIHSPacketType

MIHSPacketType	Value
PACTYPE_TIMING	0
PACTYPE_METADATAEXPERIENCE	1
PACTYPE_METADATAPERCEPTION	2
PACTYPE_METADATACHANNEL	3
PACTYPE_METADATABAND	4
PACTYPE_DATA	5

Table 33 *(continued)*

MIHSPacketType	Value
PACTYPE_LIBRARYEFFECTS	6
PACTYPE_CRC16	7
PACTYPE_CRC32	8
PACTYPE_GlobalCRC16	9
PACTYPE_GlobalCRC32	10
PACTYPE_INIT_TIMING	11
/* reserved */	12–15

Reserved This value is reserved bits; decoders should ignore it.

MIHSPacketLength Length of the payload in bytes.

MIHSPacketPayload() Payload for the MIHS packet defined in subclause 7.2.4.

ByteAlignement Padding with up to seven bits set to 0 for the MIHSPacket to be byte-aligned.

7.2.4 MIHSPacketPayload()

Table 34 details the syntax of the payload of a MIHS Packet.

Table 34 — Syntax of MIHSPacketPayload()

Syntax	No. of bits	Mnemonic

mpegiPacketPayload(MIHSPacketType)
{
switch (MIHSPacketType) {
case PACTYPE_INIT_TIMING:
readMetadataInitializationTiming();
break;
case PACTYPE_TIMING:
readMetadataTiming();
break;
case PACTYPE_METADATAEXPERIENCE:
readMetadataExperience();
break;
case PACTYPE_METADATAPERCEPTION:
readMetadataPerception();
break;
case PACTYPE_METADATACHANNEL:
readMetadataChannel();
break;
case PACTYPE_METADATABAND:
readMetadataBand();
break;
case PACTYPE_DATA:
readData ();		

Table 34 *(continued)*

Syntax	No. of bits	Mnemonic

break;
case PACTYPE_LIBRARYEFFECTS:
readLibrary();
break;
case PACTYPE_CRC16:
case PACTYPE_CRC32:
case PACTYPE_GlobalCRC16:
case PACTYPE_GlobalCRC32:
readCRC();
break;
}
}		

readMetadataInitializationTiming() Metadata related to the timing of Initialization MIHS Units, described in subclause 7.2.5.

readMetadataTiming() Metadata related to the timing of subsequent MIHS packets, described in subclause7.2.6.

readMetadataExperience() Metadata of the haptic experience being streamed, described in subclause 7.2.7.

readMetadataPerception() Metadata of a perception of the haptic experience, described in subclause 7.2.9.

readMetadataChannel() Metadata of a channel of the haptic experience, described in subclause 7.2.11.

readMetadataBand() Metadata of a band of the haptic experience, described in subclause 7.2.12.

readData() Effects for a band of the haptic experience, described in subclause 7.2.15.

readLibrary() Predefined effects to be referenced from the bands of the haptic experience, described in subclause 7.2.13.

readCRC() Cyclic redundancy check value used to protect one or more directly following packets, described in subclause 7.2.23.

7.2.5 readMetadataInitializationTiming()

Table 35 details the syntax of an initialization timing packet.

Table 35 — Syntax of readMetadataInitializationTiming()

Syntax	No. of bits	Mnemonic
readMetadataInitializationTiming()		
{		
timestamp;	**32**	**uimsbf**
timescale;	**32**	**uimsbf**
nominalDuration;	**24**	**uimsbf**
durationDeviation;	**24**	**uimsbf**
overlapping	**1**	**boolean**
}		

timestamp Timestamp of the haptic experience in ticks, i.e., the timestamp in seconds is timestamp/timescale.

timescale Number of ticks per second.

nominalDuration The nominal duration of the following MIHS temporal and silent units. The nominalDuration is in the timescale, i.e. nominalDuration/timescale indicates the nominal duration of MIHS units in seconds. MIHSUnitDuration value defines the exact duration for each MIHS unit.

durationDeviation The maximum deviation of a MIHS temporal or silent unit duration from the nominalDuration. The durationDeviation is in the timescale, i.e. durationDeviation/timescale indicates the deviation duration of MIHS units in seconds. The value 0 means that all MIHS temporal and silent units have exact duration equal to the nominal duration. The value 0xFFF indicates that there is no deviation limit for the MIHS packets.

The last MIHS temporal or silent unit before the next MIHS initialization unit may not follow the requirements for duration deviation.

overlapping Indicates if the temporal and silent MIHS Units may be overlapping. If false, no temporal or silent MIHS Units shall contain timing packets with the same timestamp until the next Initialization MIHS Unit. If true, subsequent temporal or Silent MIHS Units may be temporally aligned.

7.2.6 readMetadataTiming()

Table 36 details the syntax of a timing packet.

Table 36 — Syntax of readMetadataTiming()

Syntax	No. of bits	Mnemonic
readMetadataTiming()		
{		
timestamp;	**32**	**uimsbf**
}		

timestamp Timestamp of the haptic experience in ticks, i.e., the timestamp in seconds is timestamp/timescale.

Timescale Number of ticks per second. The default value is 1000.

7.2.7 readMetadataExperience()

Table 37 details the syntax of an experience metadata packet.

Table 37 — Syntax of readMetadataExperience()

Syntax	No. of bits	Mnemonic
readMetadataExperience()		
{		
versionLength;	**8**	**uimsbf**
version;	**versionLength*8**	**vlcs8**
profileLength;	**8**	**uimsbf**
profile;	**profileLength*8**	**vlcs8**
level;	**8**	**uimsbf**
dateLength;	**8**	**uimsbf**
date;	**dateLength*8**	**vlcs8**
descriptionLength;	**8**	**uimsbf**
description;	**descriptionLength*8**	**vlcs8**
perceptionCount;	**8**	**uimsbf**
avatarCount;	**8**	**uimsbf**
for (i = 0; i < avatarCount; i++) {		
readAvatar();		
}		
}		

versionLength Number of chars in version string.

version List of chars representing the year of the edition and amendment of ISO/IEC 23090-31 that this file conforms to, in the following format: XXXX or XXXX-Y, where XXXX is the year of publication and Y is the amendment number, if any. For this document, the value shall be “2023”.

profileLength Number of chars in profile string.

profile Name of the profile used to generate the encoded stream according to the profile and level definition in the normative Annex D.

level Number of the level used to generate the encoded stream according to the profile and level definition in the normative Annex D.

dateLength Number of chars in date string.

date List of chars representing the creation date of the haptic experience in human-readable form. The date format shall conform to ISO 8601-1 and ISO 8601-2.

descriptionLength Number of chars in description string.

description List of chars representing a description of the haptic experience.

perceptionCount Number of perceptions in the haptic experience.

avatarCount Number of avatars in the haptic experience.

readAvatar() Avatar object defined in subclause 7.2.8.

7.2.8 readAvatar()

Table 38 details the syntax of an avatar metadata packet.

Table 38 — Syntax of readAvatar()

Syntax	No. of bits	Mnemonic
readAvatar()		
{		
id;	**8**	**uimsbf**
lod;	**8**	**uimsbf**
avatarType;	**8**	**uimsbf**
if (avatarType == 0) {		
meshLength;	**8**	**uimsbf**
mesh;	**meshLength*8**	**vlcs8**
}		
}		

id ID of the avatar.

lod Level of detail of the avatar.

avatarType Type of haptic perception represented by the avatar. The possible values for avatarType are listed in Table 39.

Table 39 — Value of avatarType

Value	Meaning
0	Custom
1	Vibration
2	Pressure
3	Temperature
4–255	/* reserved */

meshLength Number of chars in the mesh string. Only present if avatarType is custom.

mesh URI to the custom mesh file. Only present if avatarType is custom.

7.2.9 readMetadataPerception()

Table 40 details the syntax of a perception metadata packet.

Table 40 — Syntax of readMetadataPerception()

Syntax	No. of bits	Mnemonic
readMetadataPerception()		
{		
id;	**8**	**uimsbf**
priority	**8**	**uimsbf**
descriptionLength;	**8**	**uimsbf**
description;	**descriptionLength*8**	**vlcs8**
perceptionModality;	**8**	**uimsbf**
avatarId	**8**	**uimsbf**
effectLibraryCount;	**16**	**uimsbf**
flagScheme;	**1**	**boolean**
if(**flagScheme**){		
schemeLength	**8**	**uimsbf**
schemeURN;	**schemeLength *8**	**vlcs8**
}		
unitExponent;	**8**	**imsbf**
perceptionUnitExponent;	**8**	**imsbf**
referenceDeviceCount;	**8**	**uimsbf**
for (i = 0; i < referenceDeviceCount; i++) {		
readReferenceDevice();		
}		
channelCount;	**16**	**uimsbf**
}		

id ID of the perception in the haptic experience. The value shall be equal to or greater than zero.

priority Importance of the perception for scalability. A lower value indicates higher priority. Given a limited bandwidth, decoders may ignore perception with a higher priority value.

descriptionLength Number of chars in the description string.

description Description of the perception.

perceptionModality Type of perception represented. Table 41 lists the possible values for perceptionModality.

Table 41 — Value of perceptionModality

Value	Meaning
0	Other
1	Pressure
2	Acceleration
3	Velocity
4	Position
5	Temperature
6	Vibrotactile
7	Water

Table 41 *(continued)*

Value	Meaning
8	Wind
9	Force
10	Vibrotactile texture
11	Electrotactile
12	Stiffness
13	Friction
14	Humidity
15	User-defined-temporal
16	User-defined-spatial
17–255	/* reserved */

avatarId Unique ID of the associated avatar body model defined in subclause 7.2.8. The 0 value means that no avatar is specified.

effectLibraryCount Number of effects in the perception's effect library.

flagScheme Flag for signalling the effect scheme other than the one defined by this document. If this flag is 0, the scheme presented in Table 55 is used. Semantic keyword is a two-layers hierarchical metadata structure illustrated in Table 13. It may be added as a supplementary information to an effect. At most only one layer 1 and one layer 2 semantic keywords shall be included with one effect. These keywords indicate the desired designer intention. When a presentation engine is incapable of rendering the exact specified effect it may decide to render a similar effect or not render at all. One example is a gunshot designed for a VR controller with a specific frequency range. A VR gun prop may be developed with its own haptics library. Table 13

schemeLength Number of chars in the effect semantic scheme identifier string.

schemeURN Effect semantic scheme identifier which is in form of a URN.

unitExponent Refers to the 10x exponent for the SI unit of effect position and keyframe positions of spatial modalities.

perceptionUnitExponent Refers to the 10x exponent for the SI unit of the dependent variable (see Table 4).

referenceDeviceCount Number of reference devices associated with the perception.

readReferenceDevice() Reference device structure described in subclause 7.2.10.

channelCount Number of channels in the perception.

7.2.10 readReferenceDevice()

Table 42 details the syntax of the reference device data of a perception metadata packet.

Table 42 — Syntax of readReferenceDevice()

Syntax	No. of bits	Mnemonic
readReferenceDevice()		
{		
id;	**8**	**uimsbf**
nameLength;	**8**	**uimsbf**
name;	**nameLength *8**	**vlcs8**
bodyPartMask;	**32**	**uimsbf**
optionalFieldMask;	**12**	**uimsbf**
if (optionalFieldMask & 0x00'01) {		
maximumFrequency;	**32**	**duimsbf**
}		
if (optionalFieldMask & 0x00'02) {		
minimumFrequency;	**32**	**duimsbf**
}		
if (optionalFieldMask & 0x00'04) {		
resonanceFrequency;	**32**	**duimsbf**
}		
if (optionalFieldMask & 0x00'08) {		
maximumAmplitude;	**32**	**duimsbf**
}		
if (optionalFieldMask & 0x00'10) {		
impedance;	**32**	**duimsbf**
}		
if (optionalFieldMask & 0x00'20) {		
maximumVoltage;	**32**	**duimsbf**
}		
if (optionalFieldMask & 0x00'40) {		
maximumCurrent;	**32**	**duimsbf**
}		
if (optionalFieldMask & 0x00'80) {		
maximumDisplacement;	**32**	**duimsbf**
}		
if (optionalFieldMask & 0x01'00) {		
weight;	**32**	**duimsbf**
}		
if (optionalFieldMask & 0x02'00) {		
size;	**32**	**duimsbf**
}		
if (optionalFieldMask & 0x04'00) {		
custom;	**32**	**duimsbf**
}		
if (optionalFieldMask & 0x08'00) {		
type;	**4**	**uimsbf**
}		
}		

id	ID of the reference device. The value shall be greater than zero. The 0 value is reserved for unspecified reference device.
nameLength	Number of chars in the name string.
name	Name of the reference device.
bodyPartMask	Binary mask specifying the location of the device on the human body according to subclause 5.6 and Table 7.
optionalFieldMask	Binary mask for the optional parameters of the reference device.
maximumFrequency	Maximum frequency of the actuator in Hertz. The range of this decimal number is [0,10000]. Present only if optionalFieldMask & 0x00'01 is true.
minimumFrequency	Minimum frequency of the actuator in Hertz. The range of this decimal number is [0,10000]. Present only if optionalFieldMask & 0x00'02 is true.
resonanceFrequency	Resonance frequency of the actuator in Hertz. The range of this decimal number is [0,10000]. Present only if optionalFieldMask & 0x00'04 is true.
maximumAmplitude	Maximum amplitude of the device according to the perception_modality. The range of this decimal number is [0,10000]. Present only if optionalFieldMask & 0x00'08 is true.
impedance	Impedance of the actuator in Ohms. The range of this decimal number is [0,10000]. Present only if optionalFieldMask & 0x00'10 is true.
maximumVoltage	Maximum voltage of the actuator. The range of this decimal number is [0,10000]. Present only if optionalFieldMask & 0x00'20 is true.
maximumCurrent	Maximum current of the actuator in Amperes. The range of this decimal number is [0,10000]. Present only if optionalFieldMask & 0x00'40 is true.
maximumDisplacement	Maximum displacement of the actuator in millimetres. The range of this decimal number is [0,10000]. Present only if optionalFieldMask & 0x00'80 is true.
weight	Weight of the device in kilograms. The range of this decimal number is [0,10000]. Present only if optionalFieldMask & 0x01'00 is true.
size	Size of the device in millimetres. The range of this decimal number is [0,10000]. Present only if optionalFieldMask & 0x02'00 is true.
custom	Custom data. The range of this decimal number is [–10000,10000]. Present only if optionalFieldMask & 0x04'00 is true.
type	Type of actuator, present only if optionalFieldMask & 0x08'00 is true. Table 43 lists the possible values for type.

Table 43 — Value of type

Value	Meaning
0	Unknown
1	LRA
2	VCA
3	ERM
4	Piezo
5–15	/* reserved */

7.2.11 readMetadataChannel()

Table 44 details the syntax of a channel metadata packet.

Table 44 — Syntax of readMetadataChannel()

Syntax	No. of bits	Mnemonic
readMetadataChannel()		
{		
id;	**16**	**uimsbf**
perceptionId;	**8**	**uimsbf**
priority	**8**	**uimsbf**
descriptionLength;	**8**	**uimsbf**
description;	**descriptionLength*8**	**vlcs8**
deviceId;	**8**	**uimsbf**
gain;	**32**	**duimsbf**
mixingCoefficient;	**32**	**duimsbf**
optionalMetadataMask;	**8**	**uimsbf**
if ((optionalMetadataMask & 0x01) != 0) {		
bodyPartMask;	**32**	**uimsbf**
}		
if ((optionalMetadataMask & 0x02) != 0) {		
actuatorResolution.X;	**8**	**imsbf**
actuatorResolution.Y;	**8**	**imsbf**
actuatorResolution.Z;	**8**	**imsbf**
bodyPartTargetCount;	**8**	**uimsbf**
for (i = 0; i < bodyPartTargetCount; i++) {		
bodyPartTarget[i];	**8**	**uimsbf**
}		
actuatorTargetCount;	**8**	**uimsbf**
for (i = 0; i < actuatorTargetCount; i++) {		
actuatorTarget[i].X	**8**	**imsbf**
actuatorTarget[i].Y	**8**	**imsbf**
actuatorTarget[i].Z	**8**	**imsbf**
}		
}		
frequencySampling;	**32**	**uimsbf**
if (frequencySampling > 0) {		
sampleCount;	**32**	**uimsbf**

Table 44 *(continued)*

Syntax	No. of bits	Mnemonic
}		
if ((optionalMetadataMask & 0x04) != 0) {		
direction.X;	**8**	**duimsbf**
direction.Y;	**8**	**duimsbf**
direction.Z;	**8**	**duimsbf**
}		
verticesCount;	**16**	**uimsbf**
for (i = 0; i < verticesCount; i++) {		
vertex;	**32**	**uimsbf**
}		
bandCount;	**8**	**uimsbf**
}		

id Unique ID of the channel in the perception containing the channel. The value shall be equal to or greater than zero.

perceptionId ID of the perception to which the channel is attached.

priority Importance of the channel for scalability. A lower value indicates higher priority. Given a limited bandwidth, decoders may ignore perception with a higher priority value.

descriptionLength Number of chars in the description string.

description Description of the channel.

deviceId ID of the associated device. For unspecified devices, the value shall be 0.

gain Gain associated with the channel. The range of this decimal number is [–10000,10000].

mixingCoefficient Mixing coefficient of the channel. The range of this decimal number is [0,10000].

optionalMetadataMask Binary mask to define if optional information is stored or not.

bodyPartMask Binary mask specifying body parts on which to apply the effect body according to subclause 5.6 and Table 7.

actuatorResolution Reference actuator resolution used to design the haptic experience for each spatial coordinate (X, Y, Z).

bodyPartTargetCount Number of bodyPartTarget in the channel.

bodyPartTarget Semantic identification of a unique body part or group of body parts on the human body

actuatorTargetCount Number of actuatorTarget in the channel.

actuatorTarget List of different actuators targeted by the channel and identified by its coordinates (X, Y, Z).

frequencySampling Sampling frequency of the original encoded signal in Hertz.

sampleCount Sample count of the original encoded signal. Present only if frequencySampling is greater than zero.

directionFlag Flag indicating the presence of a direction.

direction Spatial direction of the encoded signal. This value is based on a local representation of the 3D space given by the presentation engine.

verticesCount Number of vertices in the vertices list.

vertices List of vertices on the avatar representation affected by the effect.

bandCount Number of bands associated with the channel.

7.2.12 readMetadataBand()

Table 45 details the syntax of a band metadata packet.

Table 45 — Syntax of readMetadataBand()

Syntax	No. of bits	Mnemonic
readMetadataBand() {		
id;	**8**	**uimsbf**
perceptionId;	**8**	**uimsbf**
channelId;	**16**	**uimsbf**
priority	**8**	**uimsbf**
bandType;	**3**	**uimsbf**
if (bandType == 1) {		
curveType;	**4**	**uimsbf**
} else if(bandType == 3) {		
blockLengthLog;	**8**	**uimsbf**
}		
lowerFrequency;	**16**	**duimsbf**
upperFrequency;	**16**	**duimsbf**
effectsCount;	**16**	**uimsbf**

id Unique ID of the band in the channel containing the band.

perceptionId ID of the perception to which the band belongs.

channelId ID of the channel to which the band belongs.

priority Importance of the band for scalability. A lower value indicates higher priority. Given a limited bandwidth, decoders may ignore perception with a higher priority value.

bandType Type of band. Table 46 lists the possible values for bandTtype.

Table 46 — Value of bandType

Value	Meaning
0	Transient
1	Curve
2	Vectorial wave
3	Wavelet wave
4–7	/* reserved */

curveType Interpolation function to use. Table 47 lists the possible values for curveType .

Table 47 — Value of curveType

Value	Meaning
0	Unknown
1	Cubic
2	Linear
3	Akima
4	Bézier
5	B-spline
6–15	/* reserved */

blockLengthLog The blockLength is coded in samples. Additionally, it is transformed to the logarithmic domain using the formula

$$blockLength\_log = \log_2(blockLength) - 4 .$$

This leads to an allowed minimum blockLengthLog of 16 and only powers of 2 in samples, which is required for wavelet coding.

lowerFrequency Lower frequency limit of the band in Hertz. The range of this decimal number is [0,10000].

upperFrequency Upper frequency limit of the band in Hertz. The range of this decimal number is [0,10000].

effectsCount Number of effects present in the band.

7.2.13 readLibrary()

Table 48 details the syntax of an effect library packet.

Table 48 — Syntax of readLibrary()

Syntax	No. of bits	Mnemonic
readLibrary() {		
perceptionId;	**8**	**uimsbf**
effectCount;	**16**	**uimsbf**
for (i = 0; i < effectCount; i++) {		
readLibraryEffect()		
}		
}		

perceptionId ID of the perception to which the library is attached.

effectCount Number of effects in the library.

readLibraryEffect() Effect structure as described in subclause 7.2.14.

7.2.14 readLibraryEffect()

Table 49 details the syntax of an effect in an effect library packet.

Table 49 — Syntax of readLibraryEffect()

Syntax	No. of bits	Mnemonic
readLibraryEffect() {		
id;	**16**	**uimsbf**
effectType	**2**	**uimsbf**
hasSemantic	**1**	**boolean**
if(hasSemantic){		
semanticKeywords	**12**	**uimsbf**
}		
position;	**25**	**imsbf**
if (effectType == 0) {		
phase;	**16**	**uimsbf**
baseSignal;	**4**	**uimsbf**
}		
keyframesCount;	**16**	**uimsbf**
for (i = 0; i < keyframesCount; i++) {		
mask;	**3**	**uimsbf**
if (mask & 0x001) {		
relativePosition;	**16**	**uimsbf**
}		
if (mask & 0x010) {		
amplitude;	**8**	**duimsbf**
}		
if (mask & 0x100) {		
frequency;	**16**	**uimsbf**
}		
}		
compositeEffectCount;	**16**	**uimsbf**
for (i = 0; i < compositeEffectCount; i++) {		
readLibraryEffect();		
}		
}		

id ID of the effect in the library. The value shall be unique in the perception.

position Position of the effect, relative to the position of the calling effect.

hasSemantic Flag signalling the presence of a semantic keyword for the current effect. Value 0 means no semantic is included.

semanticKeywords Semantic keyword as defined in subclause 5.8.

phase Phase of the effect.

baseSignal Base signal of the effect. Table 50 lists the possible values for baseSignal.

Table 50 — Value of baseSignal

Value	Meaning
0	Sine
1	Square
2	Triangle
3	Sawtooth up
4	Sawtooth down
5–15	/* reserved */

effectType Type of the effect. Table 51 lists the possible values for effectType.

Table 51 — Value of effectType

Value	Meaning
0	Basis
1	Reference
2	Composite
3	/* reserved*/

keyframesCount Number of keyframes in the effect.

mask Information mask indicating which keyframe parameters are present.

relativePosition Keyframe position relative to the effect position, present only if mask & 0x001 is true.

amplitude Keyframe amplitude, present only if mask & 0x010 is true.

frequency Keyframe frequency in Hertz, present only if mask & 0x100 is true.

compositeEffectCount Number of effects comprising the composite effect.

readLibraryEffect() Recursive structure if the effect is a composite effect and contains other effects.

7.2.15 readData()

Table 52 details the syntax of a data packet.

Table 52 — Syntax of readData()

Syntax	No. of bits	Mnemonic
readData() {		
packetDependency;	**1**	**bool**
perceptionId;	**8**	**uimsbf**
channelId;	**16**	**uimsbf**
bandId;	**8**	**uimsbf**
effectsCount;	**16**	**uimsbf**
if (bandType != 3) {		
for (i = 0; i < effectsCount; i++) {		
readEffect();		

Table 52 *(continued)*

Syntax	No. of bits	Mnemonic

}
} else {
readWaveletEffect();
}
}		

packetDependency Type of packet. If 1, the packet depends on one or more previous packets to be interpreted. If 0, the packet can be interpreted independently of previous packets. The value of this property in the first data packet of an MIHS Unit with MIHSUnitSync =0 shall be set to 0.

perceptionId ID of the perception associated with the packet.

channelId ID of the channel associated with the packet.

bandId ID of the band associated with the packet.

effectsCount Number of effects in the packet.

readEffect() Effect structure described in subclause 7.2.16.

readWaveletEffect() Wavelet effect structure described in subclause 7.2.22.

7.2.16 readEffect()

Table 53 details the syntax of an effect in a data packet for a band type other than a WaveletWave.

Table 53 — Syntax of readEffect()

Syntax	No. of bits	Mnemonic
readEffect() {		
id;	**16**	**uimsbf**
effectType;	**2**	**uimsbf**
effectPosition;	**25**	**imsbf**
if (effectType == 0) {		
hasSemantic	**1**	**boolean**
if(hasSemantic){		
semanticKeywords	**12**	**uimsbf**
}		
readEffectBasis();		
}		
}		

id ID of the effect in the effect library of the perception containing the effect. The value shall be equal to an existing effect id from the effect library. Each property for all effects with the same id value shall be either identical or in the case of position shall refer to the same absolute position value.

effectType Type of the effect. Table 54 lists the possible values for effectType.

Table 54 — Value of effectType

Value	Meaning
0	Basis
1	Reference

hasSemantic Flag signalling the presence of a semantic keyword for the current effect. Value 0 means no semantic is included.

semanticKeywords Semantic keyword representing a semantic keyword as defined in subclause 5.8. The bits code of the default semantic keyword scheme is given in Table 55. For the default semantic scheme, the first four bits correspond to the semantic information of the first semantic layer. The next eight bits correspond to the semantic information of the second layer. The syntax of this property for other semantic schemes is defined by the scheme owner.

effectPosition Effect position relative to the packet timestamp for temporal data. For temporal data, this value is in the timescale of the readMetadataTiming() structure in the most recently sent mpegHapticUnit() with MIHSUnitType UNITTYPE_INITIALIZATION, and shall be smaller than the value of MIHSUnitDuration of the mpegiHapticUnit() that contains this effect. The value may be negative if the effect was started in a previous mpegiHapticUnit(). In this case, the MIHSUnitSync and the mpegiHapticPacket() packetDependency values shall be set to 1. For spatial data, the effect position is relative to the origin.

NOTE: The temporal or spatial position of the effect is defined by the value of perceptionModality and according to Table 4.

readEffectBasis() Basis effect structure, described in subclause 7.2.17.

Table 55 — Default semantic keywords structure with corresponding bits code

Layer 1	Layer 1 bits code	Layer 2	Layer 2 bits code	Decimal Value corresponance (combined Layer1 and Layer2)
UX	0000	Undefined (default)	0000 0000	0
		Click	0000 0001	1
		Double click	0000 0010	2
		Success	0000 0011	3
		Error	0000 0100	4
		Alarm	0000 0101	5
		Confirmation	0000 0110	6
		Wrong	0000 0111	7
		Ring	0000 1000	8
		Message	0000 1001	9

Table 55 *(continued)*

Layer 1	Layer 1 bits code	Layer 2	Layer 2 bits code	Decimal Value corresponance (combined Layer1 and Layer2)
		Reserved	0000 1010 - 1111 1111	10-255
Avatar	0001	Undefined (default)	0000 0000	256
		Jumping	0000 0001	257
		Fall	0000 0010	258
		Crawl	0000 0011	259
		Swim	0000 0100	260
		Collision	0000 0101	261
		Grab	0000 0110	262
		Touch	0000 0111	263
		Swip	0000 1000	264
		Footstep	0000 1001	265
		Reserved	0000 1010 - 1111 1111	266-511
Special effect	0010	Undefined (default)	0000 0000	512
		Washout	0000 0001	513
		Noise	0000 0010	514
		Reserved	0000 0011 - 1111 1111	515-767
Weapons & Combat	0011	Undefined (default)	0000 0000	768
		Blade	0000 0001	769
		Hit	0000 0010	770
		Hand-thrown	0000 0011	771
		Elastic propulsion	0000 0100	772
		Pneumatic	0000 0101	773
		Handguns	0000 0110	774
		Rifles	0000 0111	775
		Shotgun	0000 1000	776
		Gun	0000 1001	777
		Machinegun	0000 1010	778
		Taser	0000 1011	779
		Electric shock	0000 1100	780
		Mines	0000 1101	781
		Missile	0000 1110	782
		Grenade	0000 1111	783
		Blast	0001 0000	784
		Reserved	0001 0001 - 1111 1111	785-1023
		Undefined (default)	0000 0000	1024
		Wind low	0000 0001	1025
		Heat	0000 0010	1026
		Cold	0000 0011	1027

Table 55 *(continued)*

Layer 1	Layer 1 bits code	Layer 2	Layer 2 bits code	Decimal Value corresponance (combined Layer1 and Layer2)
Ambient	0100	Rain	0000 0100	1028
		Waterfall	0000 0101	1029
		Water drop	0000 0110	1030
		Electric buzz	0000 0111	1031
		Ignition	0000 1000	1032
		Cracks	0000 1001	1033
		Earthquake	0000 1010	1034
		Sparks	0000 1011	1035
		Thunderbolt	0000 1100	1036
		Reserved	0000 1101 - 1111 1111	1037-1279
Texture	0101	Undefined (default)	0000 0000	1280
		Rock	0000 0001	1281
		Gravel	0000 0010	1282
		Sand	0000 0011	1283
		Wood	0000 0100	1284
		Metal	0000 0101	1285
		Plastic	0000 0110	1286
		Reserved	0000 0111 - 1111 1111	1287-1535
Vehicles	0110	Undefined (default)	0000 0000	1536
		Engine	0000 0001	1537
		Doors	0000 0010	1538
		Brake	0000 0011	1539
		Mechanical Contraption	0000 0100	1540
		Drift	0000 0101	1541
		Road friction	0000 0110	1542
		Brake	0000 0111	1543
		Road bump	0000 1000	1544
		Tires	0000 1001	1545
		Air friction	0000 1010	1546
		Reserved	0000 1011 - 1111 1111	1547-1791
Music	0111	Undefined (default)	0000 0000	1792
		Hard material	0000 0001	1793
		Bouncy material	0000 0010	1794
		Pucking	0000 0011	1795
		Bowing	0000 0100	1796
		Dtriking	0000 0101	1797
		Brass instruments	0000 0110	1798
		Woodwind instruments	0000 0111	1799

Table 55 *(continued)*

Layer 1	Layer 1 bits code	Layer 2	Layer 2 bits code	Decimal Value corresponance (combined Layer1 and Layer2)
		Reserved	0000 1000 - 1111 1111	1800-2047
Reserved	1000 - 1111	Reserved	0000 0000 - 1111 1111	2048-4095

7.2.17 readEffectBasis()

Table 56 details the syntax of a basis effect in a data packet.

Table 56 — Syntax of readEffectBasis()

Syntax	No. of bits	Mnemonic
readEffectBasis() {		
keyframesCount;	**16**	**uimsbf**
if (bandType == 2) {		
phase;	**16**	**duimsbf**
baseSignal;	**4**	**uimsbf**
}		
for (i = 0; i < keyframesCount; i++) {		
readKeyframe();		
}		
}		

keyframesCount Number of keyframes in the packet belonging to the current effect.

phase Phase of the effect. Present only if bandType == 2 (vectorial wave).

baseSignal Base signal of the effect. The possible values for baseSignal are listed in Table 57. Present only if bandType == 2 (vectorial wave).

Table 57 — Value of BaseSignal

Value	Meaning
0	Sine
1	Square
2	Triangle
3	Sawtooth up
4	Sawtooth down
5–15	/* reserved */

readKeyframe() Keyframes in the packet belonging to the current effect, described in subclause 7.2.18.

7.2.18 readKeyframe()

Table 58 details the syntax of a keyframe in an effect.

Table 58 — Syntax of readKeyframe()

Syntax	No. of bits	Mnemonic

readKeyframe() {
  if (bandType == 0) {
  readTransientKeyframe();
  } else if (bandType == 1) {
  readCurveKeyframe();
  } else if (bandType == 2) {
  readVectorialKeyframe();
  }
}		

readTransientKeyframe() Transient effect keyframes, described in subclause 7.2.19..

readCurveKeyframe() Curve keyframes, described in subclause 7.2.20..

readVectorialKeyframe() Vectorial wave keyframes, described in subclause 7.2.21.

7.2.19 readTransientKeyframe()

Table 59 details the syntax of a keyframe of an effect in a band of type Transient.

Table 59 — Syntax of readTransientKeyframe()

Syntax	No. of bits	Mnemonic

readTransientKeyframe() {
  amplitude;
  position;
  frequency;
}
	
	8
	16
	16
		
		duimsbf
		uimsbf
		uimsbf

amplitude Amplitude of the keyframe.

position Keyframe position relative to effect position. For temporal keyframes, the position is in the timescale of the readMetadataTiming() in the most recently sent mpegHapticUnit() with MIHSUnitType UNITTYPE_INITIALIZATION.

frequency Frequency of the keyframe.

7.2.20 readCurveKeyframe()

Table 60 details the syntax of a keyframe of an effect in a band of type Curve.

Table 60 — Syntax of readCurveKeyframe()

Syntax	No. of bits	Mnemonic
readCurveKeyframe() {		
amplitude;	**8**	**duimsbf**
position;	**16**	**uimsbf**
}		

amplitude Amplitude of the keyframe.

position Keyframe position relative to effect position. For temporal keyframes, the position is in the timescale of the readMetadataTiming() in the most recently sent mpegHapticUnit() with MIHSUnitType UNITTYPE_INITIALIZATION.

7.2.21 readVectorialKeyFrame()

Table 61 details the syntax of a keyframe of an effect in a band of type Vectorial Wave.

Table 61 — Syntax of readVectorialKeyframe()

Syntax	No. of bits	Mnemonic
readVectorialKeyframe() {		
informationMask;	**2**	**uimsbf**
if (informationMask & 0x01) {		
amplitude;	**8**	**duimsbf**
}		
position;	**16**	**uimsbf**
if (informationMask & 0x02) {		
frequency;	**16**	**uimsbf**
}		
}		

informationMask Information mask indicating which parameters are present.

amplitude Amplitude of the keyframe. Present only if informationMask & 0x01 is true.

position Keyframe position relative to effect position. For temporal keyframes, the position is in the timescale of the readMetadataTiming() in the most recently sent mpegHapticUnit() with MIHSUnitType UNITTYPE_INITIALIZATION.

frequency Relative frequency of the keyframe. Present only if informationMask & 0x02 is true.

7.2.22 readWaveletEffect()

In wavelet bands, the signal is stored as a series of wavelet blocks. Each block contains the encoded data of a fixed number of samples of the original signal, and each effect contains the data of a single block.

Since wavelet bands assume a continuous stream of samples, each effect directly follows the previous one, meaning that position in time is not explicitly signalled for wavelet blocks. The position of the first wavelet effect in each MIHS packet is defined by the timestamp of the MIHS packet itself defined in subclause 7.2.5. This means that any MIHS packet containing wavelet effects shall have the timestamp equals to absolute position of the first wavelet effect it contains.

As detailed in Table 62, blocks are stored with a length value streamsize in bytes, and the encoded bitstream arithmetic_stream. The size of the aritmethic_stream is given by streamsize. For empty wavelet blocks, which only contain zero coefficients, the streamsize is equal to 0, and no arithmetic_stream shall be decoded.

Table 62 — Syntax of readWaveletEffect()

Syntax	No. of bits	Mnemonic
readWaveletEffect()		
{		
id;	**16**	**uimsbf**
effectType;	**2**	**uimsbf**
hasSemantic	**1**	**boolean**
if(hasSemantic){		
semanticKeywords	**12**	**uimsbf**
}		
streamsize;	**16**	**uimsbf**
arithmetic_stream;	**streamsize * 8**	**vlclbf**
SPIHT_decode(arithmetic_stream, block_length);		
}		

id ID of the effect. If the effect is of type Reference, this ID allows to retrieve the effect in the effect library.

effectType Type of the effect. Table 54 lists the possible values for effectType.

hasSemantic Flag signalling the presence of a semantic keyword for the current effect. Value 0 means no semantic is included.

semanticKeywords Semantic keyword as defined in Table 55.

streamsize Size of the arithmetic_stream.

arithmetic_stream Bitstream generated by the arithmetic encoder. This is the final compressed bitstream for a wavelet effect.

block_length Block length of the band

SPIHT_decode() binary decoding of the arithmetic_stream.

Inside the function SPIHT_decode(), the arithmetic_stream is decoded as detailed in subclause 8.3.3. The arithmetic decoding is embedded in the SPIHT decoding. This function allows to decode the stream and output the data necessary for the rendering: wavmax, maxallocBits and wavelet_coefficients.

wavelet_coefficients Array of wavelet coefficients of the current effect, scaled to [–1,1]. This is an output of the SPIHT_decode() function. The coefficients are ordered from low to high frequencies.

wavmax Quantized absolute maximum amplitude of the wavelet block. This is an output of the SPIHT_decode() function and needed to recover the original amplitude of the wavelet_coefficients in the synthesizer.

maxallocBits Maximum for quantization allocated bits in the wavelet effect. This is only used inside the SPIHT_decode() and SPIHT_encode() function and needed for correct scaling of the wavelet_coefficients.This is an output of the SPIHT_decode() function.

Each effect gets an individual position in samples. The wavelet_coefficients vector, obtained from the SPIHT decoder, is scaled to max(abs(wavelet_coefficients[i]))==1. From that, the original amplitude of the signal can be restored by multiplying all coefficients with wavmax (later also referred to as $\widehat{w_{max}}$ in formulae).

Next, the original scaling is recovered by multiplying each sample with $\widehat{w_{max}}$, and inverse wavelet transformation is applied to generate the PCM signal with the same sampling frequency as the original signal. To obtain the structure of the original signal, each effect needs to be added to the output signal at the correct position in time. Since the input of the wavelet band processing was a continuous PCM file split into blocks of equal length, every wavelet effect directly follows the previous one (with the first sample directly after the last sample of the previous effect). Therefore, no additional position information is saved in the binary format. This decoded high frequency part is then added to the low frequency part.

Further details on the modules are provided in subclause 8.3.3.

7.2.23 readCRC()

Table 63 details the syntax of a CRC packet.

Table 63 — Syntax of readCRC()

Syntax	No. of bits	Mnemonic
readCRC()		
{		
if (MIHSPacketType == PACTYPE_CRC16) {		
CRC16Value;	**16**	**uimsbf**
} else if (MIHSPacketType == PACTYPE_CRC32) {		
CRC32Value;	**32**	**uimsbf**
} else if (MIHSPacketType == PACTYPE_GlobalCRC16) {		
protectedPacketsCount;	**8**	**uimsbf**
CRC16Value;	**16**	**uimsbf**
} else if (MIHSPacketType == PACTYPE_GlobalCRC16) {		
protectedPacketsCount;	**8**	**uimsbf**
CRC32Value;	**32**	**uimsbf**
}		
}		

CRC16Value The cyclic redundancy check value computed using the polynomial

$$x^{15} + x^{14} + x^5 + 1 .$$

The algorithm to compute CRC value is given subclause 7.3.8.

CRC32Value The cyclic redundancy check value computed using the polynomial

$$x^{31} + x^{25} + x^{22} + x^{21} + x^{15} + x^{11} + x^{10} + x^9 + x^7 + x^6 + x^4 + x^2 + x + 1 .$$

The algorithm to compute CRC value is given subclause 7.3.8.

protectedPacketsCount Number of packets protected by the CRC check. Present only if MIHSPacketType is PACTYPE_GlobalCRC16 or PACTYPE_GlobalCRC32.

7.3 Description of MIHSPacketType

7.3.1 InitializationTiming

The MIHSPacketType PACTYPE_INIT_TIMING identifies a timing MIHS packet which embeds a structure, readMetadataInitializationTiming() defined in subclause 7.2.5, in the MIHSPacketPayload() defined in subclause 7.2.4, containing the timestamp of haptic effects in ticks, the timescale which defines the number of ticks per second, a nominal duration for MIHS temporal and silent units, a duration deviation indicating the maximum deviation of a MIHS temporal or silent unit duration from the nominal duration and an overlapping flag indicating if the temporal and silent MIHS Units may be overlapping.

An initialization timing MIHS packet shall be included in an initialization MIHS packet.

7.3.2 Timing

The MIHSPacketType PACTYPE_TIMING identifies a timing MIHS packet which embeds a structure, readMetadataTiming() defined in subclause 7.2.6, in the MIHSPacketPayload() defined in subclause 7.2.4, containing the timestamp of haptic effects in ticks.

7.3.3 MetadataHaptics

The MIHSPacketType PACTYPE_METADATAEXPERIENCE identifies a haptic experience metadata MIHS packet which embeds a haptic experience structure, readMetadataExperience() defined in subclause 7.2.7, in the MIHSPacketPayload() defined in subclause 7.2.4.

A haptic experience metadata MIHS packet may be sent at regular intervals in an initialization MIHS unit to enable random access of the haptic data.

Figure 17 illustrates details of the haptic experience metadata MIHS packet payload.

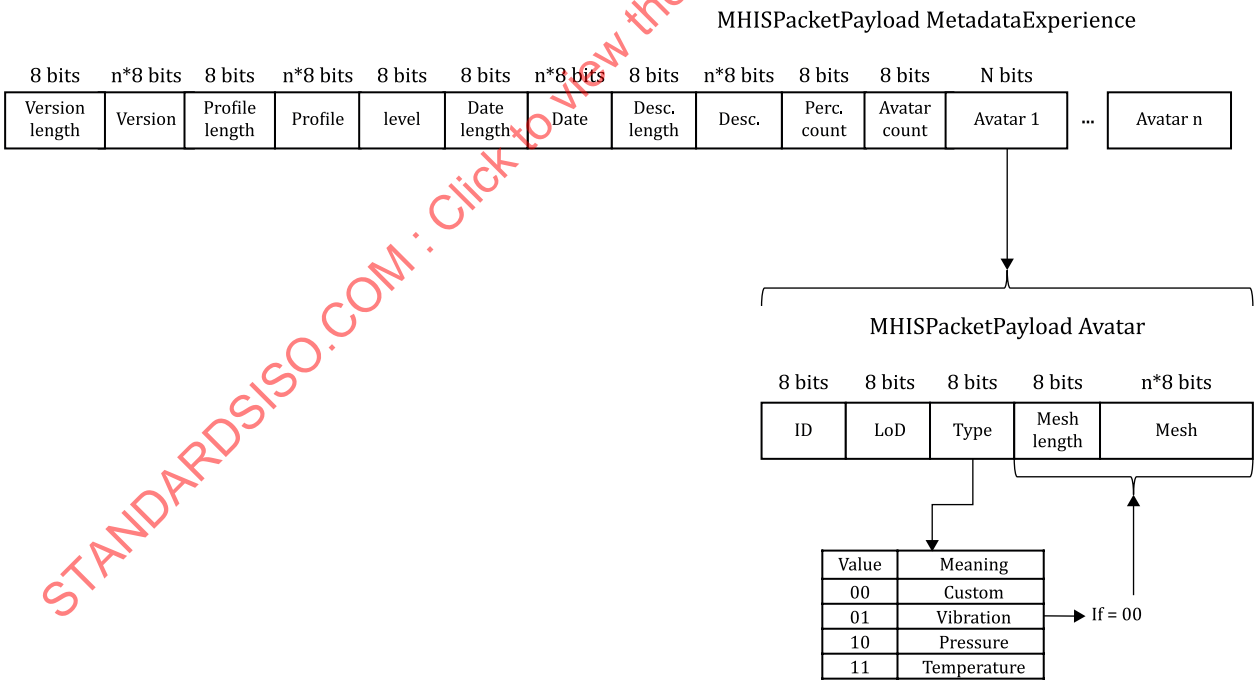


Figure 17 — MIHSPacketPayload structure for a packet type PACTYPE_METADATAEXPERIENCE

7.3.4 MetadataPerception

The MIHSPacketType PACTYPE_METADATAPERCEPTION identifies a haptic perception metadata MIHS packet which embeds a haptic perception structure, readMetadataPerception() defined in subclause 7.2.9, in the MIHSPacketPayload() defined in subclause 7.2.4.

Haptic perception metadata MIHS packets may be sent at regular intervals in an initialization MIHS unit to enable random access of the haptic data.

Figure 18 illustrates details of the haptic perception metadata MIHS packet payload.

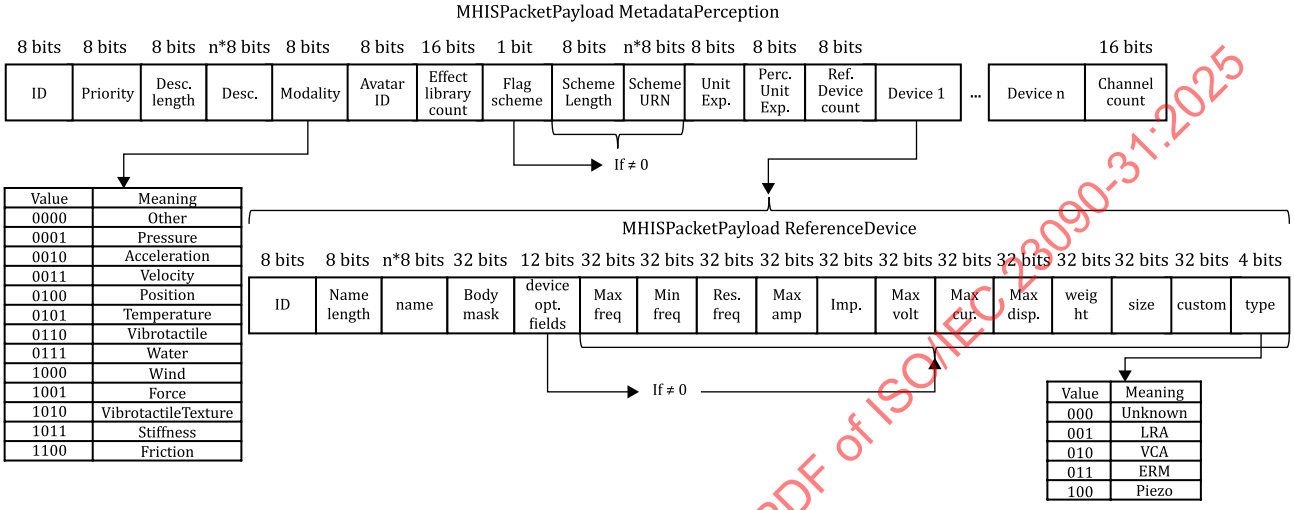


Figure 18 — MIHSPacketPaylaod structure for packet type PACTYPE_METADATAPERCEPTION

7.3.5 MetadataChannel

The MIHSPacketType PACTYPE_METADATACHANNEL identifies a haptic channel metadata MIHS packet which embeds a haptic channel structure, readMetadataChannel() defined in subclause 7.2.11, in the MIHSPacketPayload() defined in subclause 7.2.4.

Haptic channel metadata MIHS packets may be sent at regular intervals in an initialization MIHS unit to enable random access of the haptic data.

Figure 19 illustrates details of the haptic channel metadata MIHS packet payload.

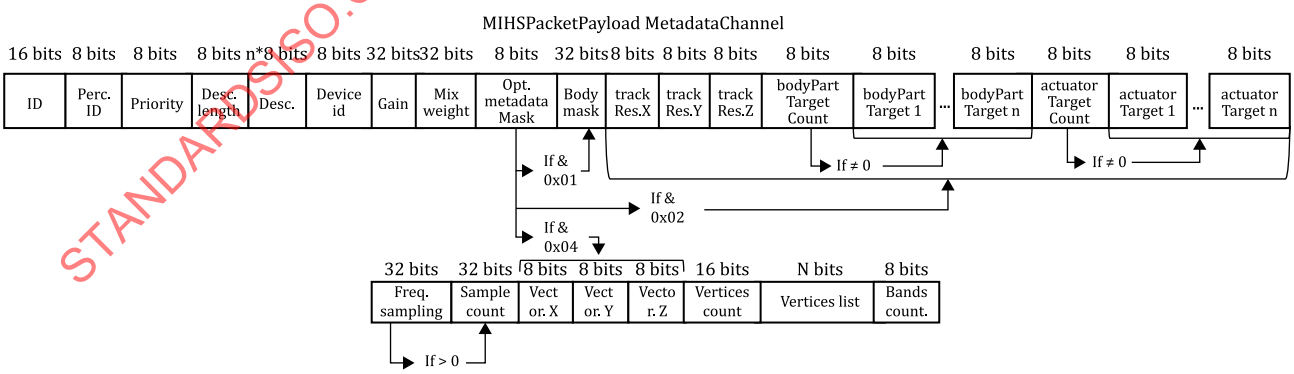


Figure 19 — MIHSPacketPayload structure for packet type PACTYPE_METADATACHANNEL

7.3.6 MetadataBand

The MIHSPacketType PACTYPE_METADATABAND identifies a haptic band metadata MIHS packet which embeds a haptic band structure, readMetadataBand() defined in subclause 7.2.12, in the MIHSPacketPayload() defined in subclause 7.2.4.

Haptic band metadata MIHS packets may be sent at regular intervals in an initialization MIHS unit to enable random access of the haptic data.

Figure 20 illustrates details of the haptic band metadata MIHS packet payload.

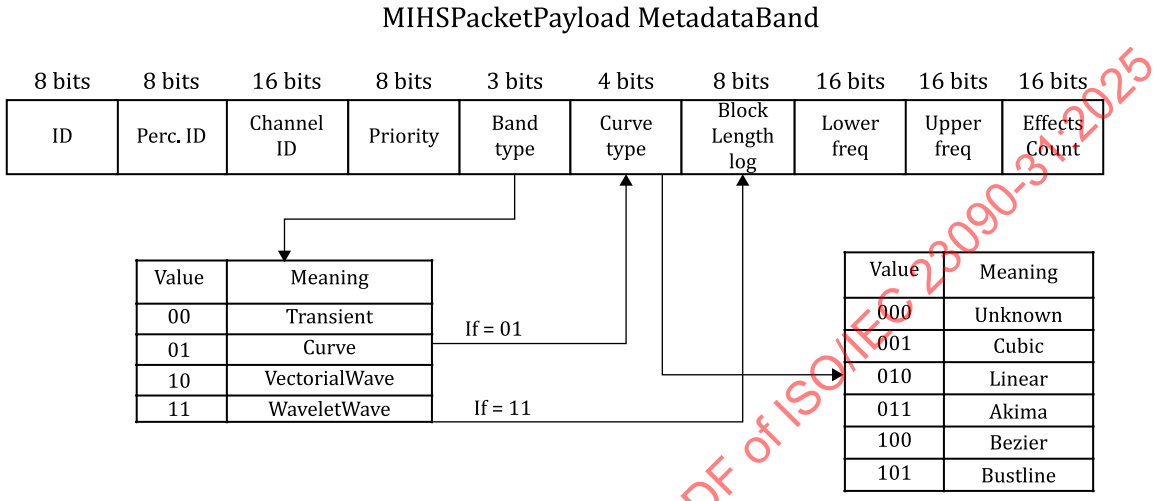


Figure 20 — MIHSPacketPayload structure for packet type PACTYPE_METADATABAND

7.3.7 LibraryEffect

The MIHSPacketType PACTYPE_LIBRARYEFFECTS identifies a haptic effect library MIHS packet which embeds a haptic effect library structure, readLibrary() defined in subclause 7.2.13, in the MIHSPacketPayload() defined in subclause 7.2.4 .

A haptic effect library MIHS packet may be sent at regular intervals in an initialization MIHS unit to enable random access of the haptic data. The effect library may change dynamically during the haptic experience.

Figure 21 illustrates details of the haptic effect library MIHS packet payload.

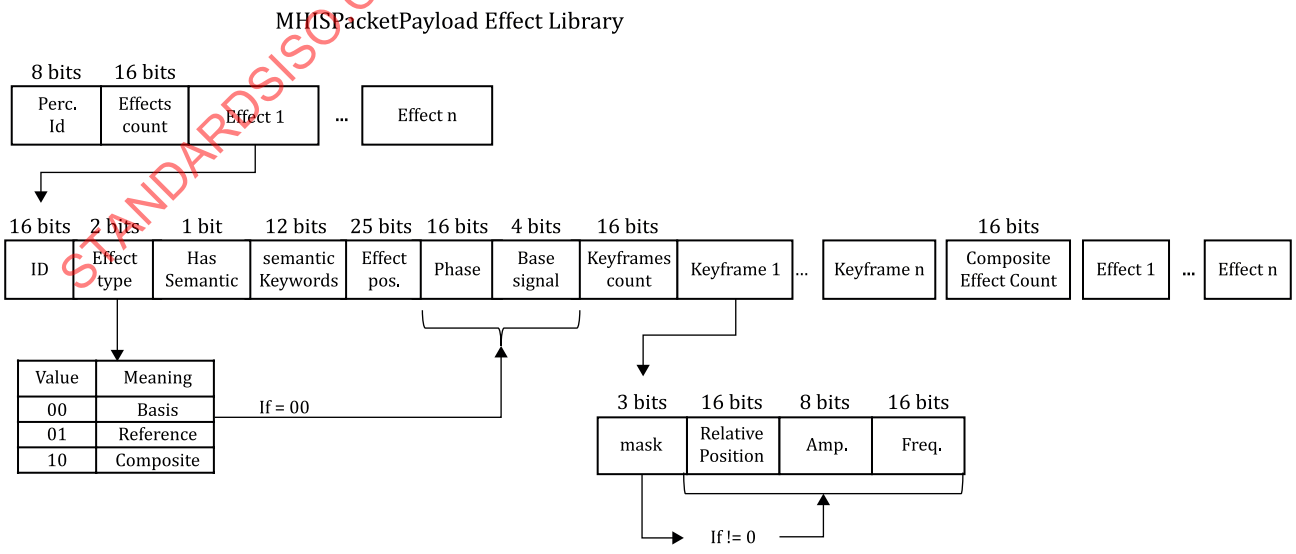


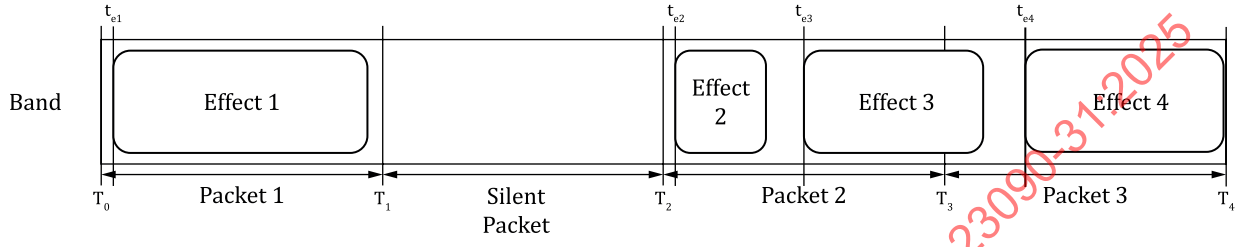
Figure 21 — MIHSPacketPayload structure for packet type PACTYPE_LIBRARYEFFECTS

7.3.8 Data

The MIHSPacketType PACTYPE_DATA identifies a haptic data MIHS packet which embeds data for effects and keyframes using a haptic data structure, readData() defined in subclause 7.2.15, in the MIHSPacketPayload() defined in subclause 7.2.4.

Haptic data MIHS packets contain the haptic data of the experience being streamed. The structure depends on the type of band and the effect type. The packetDependency flag indicates whether the packet is independent or whether the packet needs information from the previous packet.

An MIHS packet may include one or more effects as shown in Figure 22.



Packet 1 Payload

Packet Dependency	Perception ID	Channel ID	Band ID	Effects count	Effect ID	Effect Type	Effect position	DATA
Regular Packet	x	x	x	1	1	x	t_{e1}-T_0	

Silent Packet Payload

Packet Dependency	Perception ID	Channel ID	Band ID	Effects count
Regular Packet	x	x	x	0

Packet 2 Payload

Packet Dependency	Perception ID	Channel ID	Band ID	Effects count	Effect ID	Effect Type	Effect position	DATA	Effect ID	Effect Type	Effect position	DATA
Regular Packet	x	x	x	2	2	x	t_{e2}-T_2		3	x	t_{e3}-T_2	

Packet 3 Payload

Packet Dependency	Perception ID	Channel ID	Band ID	Effects count	Effect ID	Effect Type	Effect position	DATA	Effect ID	Effect Type	Effect position	DATA
Dependent Packet	x	x	x	2	3	x	t_{e3}-T_3		4	x	t_{e4}-T_3	

Figure 22 — Example of an MIHS packet

Figure 23 illustrates details of the packet payload.

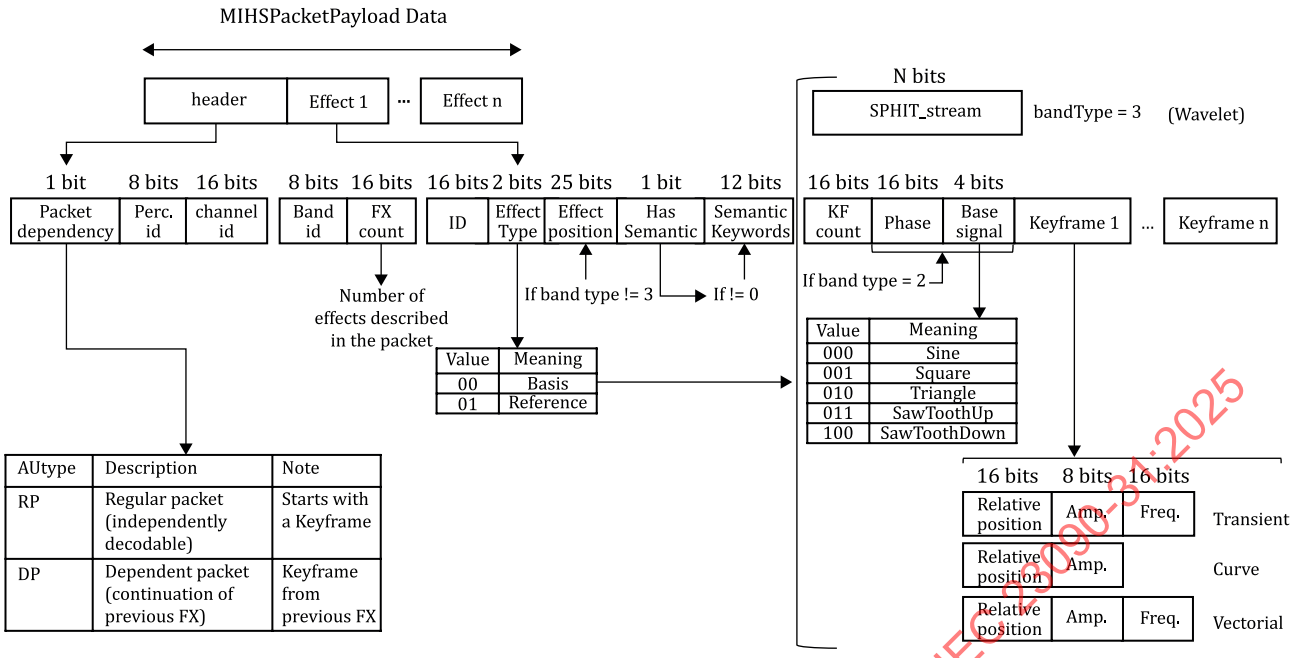


Figure 23 — MIHSPacketPayload structure for packet type PACTYPE_DATA

7.3.9 CRC16 and CRC32

The MIHSPacketType PACTYP_CRC16 or PACTYP_CRC32 identifies a CRC MIHS packet which may be used to detect errors in the subsequent MIHS packet including both the MIHSPacketHeader and MIHSPacketPayload portions of the packet. CRC MIHS packets contain a readCRC() structure defined in subclause 7.2.23 in the MIHSPacketPayload() defined in subclause 7.2.4.

A CRC MIHS packet shall be directly followed by the MIHS packet to which the CRC MIHS packet's CRC16Value or CRC32Value applies.

CRC MIHS packets may be beneficial when an MIHS stream is conveyed over an error prone channel.

The CRC value is computed using the following algorithm:

```
FUNCTION ComputeCRC(message, polynomial):
   Append(message, M) /*Add M 0 bits to message
   FOR (i in range(0:N)):
     MSB = message(i)
     IF (MSB == 1):
        message[MSB : MSB + (M-1)] ^= polynomial /* bit-wise XOR*/
   RETURN message[N-M : N]
```

Where *message* is a binary number of size N, it is the payload to check.

polynomial is a binary number of size M, it is used to compute the CRC and is known by both the encoder and decoder.

The return value is the CRC value that needs to be appended to the payload.

7.3.10 GlobalCRC16 and GlobalCRC32

The MIHSPacketType PACTYPE_GlobalCRC16 or PACTYPE_GlobalCRC32 identifies a global CRC MIHS packet which may be used to detect errors in the subsequent protectedPacketsCount MIHS packets including both the MIHSPacketHeader and MIHSPacketPayload portions of the packets. Global CRC MIHS packets contain a readCRC() structure defined in subclause 7.2.23 in the MIHSPacketPayload() defined in subclause 7.2.4.

A global CRC MIHS packet shall be directly followed by the protectedPacketsCount MIHS packets to which the global CRC MIHS packet's CRC16Value or CRC32Value applies.

Global CRC MIHS packets may be beneficial when an MIHS stream is conveyed over an error prone channel.

7.4 Application examples

7.4.1 Initialization units

Initialization MIHS units, introduced in subclause 7.1.2, contain information essential to render the haptic signal and should be sent regularly to allow random access. The typical MIHS packets that may be present in an initialization unit are illustrated in Figure 24 and may be sent in any order.

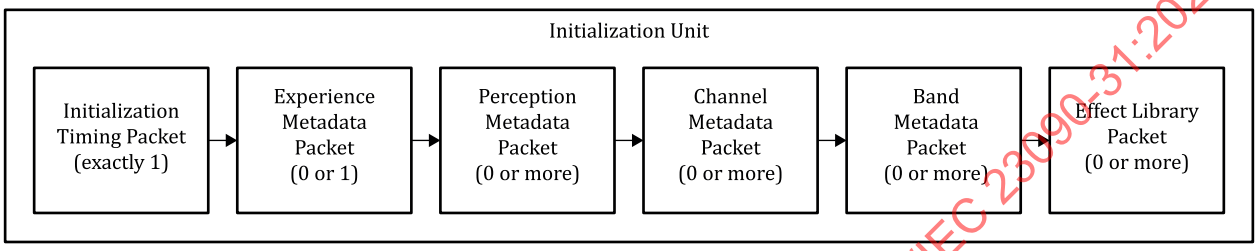


Figure 24 — Initialization unit

7.4.2 Temporal and spatial units

Temporal and Spatial MIHS units, introduced in subclause 7.1.3, contain haptic effect data defining the haptic signal. Temporal units contain data defining time-dependent effects. Spatial units contain data defining effects controlled by a spatial position. Both temporal and spatial units contain data MIHS packets as illustrated in Figure 25.

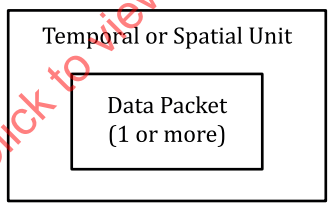


Figure 25 — Temporal and spatial units

Figure 26 shows a basic example on how MIHS units can be conveyed to stream a haptic experience.

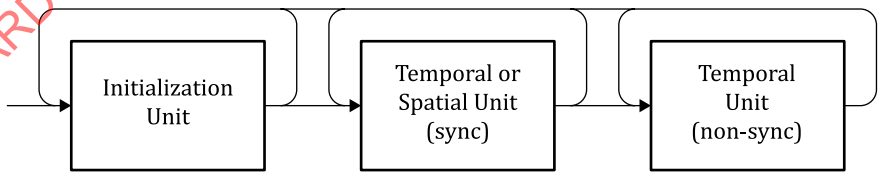


Figure 26 — Basic example of MIHS unit sequencing

Temporal MIHS units contain a non-zero duration. To improve synchronization in the stream, initialization MIHS units, which contain a timing MIHS packet with a timestamp and timescale, may be sent periodically.

7.4.3 Silent units

During intervals when there is no haptic data, silent MIHS units, which indicate a duration and contain no MIHS packets, may be added to the stream as illustrated in Figure 27

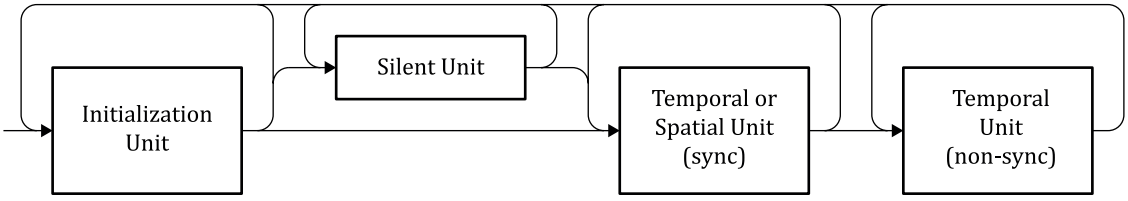


Figure 27 — Example with silent units

7.4.4 CRC packets

MIHS packets of type PACTYPE_CRC16, PACTYPE_CRC32, PACTYPE_GlobalCRC16, or PACTYPE_GlobalCRC32 may be used for error detection within initialization, temporal, or spatial MIHS units as shown Figure 28 and Figure 29.

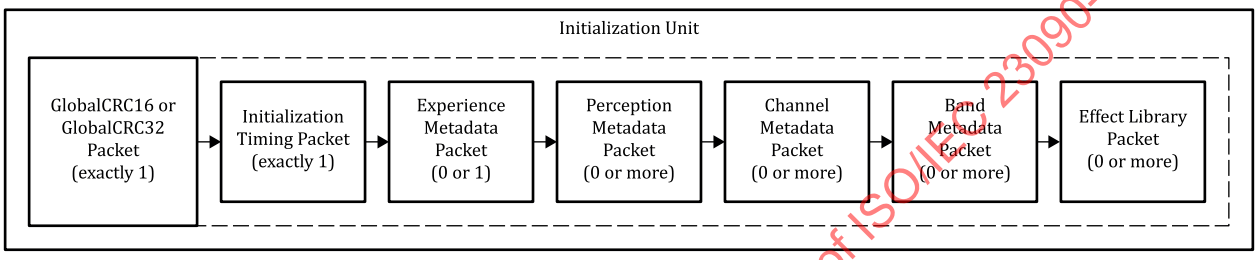


Figure 28 — GlobalCRC16 or GlobalCRC32 packet in an initialization unit

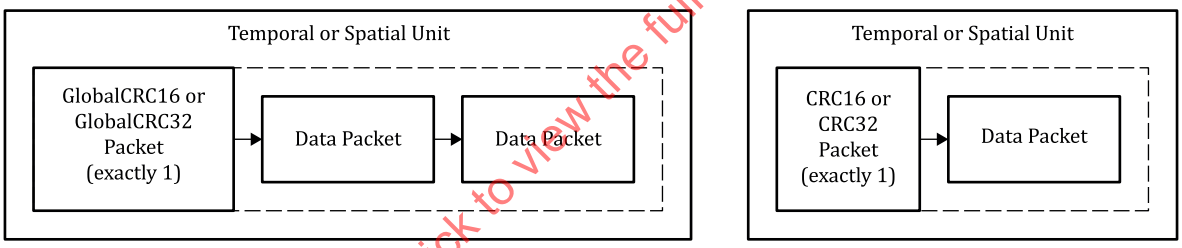


Figure 29 — CRC packets in temporal or spatial units

7.5 Random access support with MIHS (informative)

The current streaming format supports random-access for streaming applications. This subclause explains how applications should use random access.

Random access means that playback of a haptic experience can start at any time after a pre-defined random access point.

There are two signalling mechanisms with respect to random access in the MIHS format:

a) An MIHS unit is a sync unit if its MIHSUnitSync is set to 0. A decoder can decode those MIHS units and the units afterwards without decoding any previous MIHS units. Therefore, using this property, one can create a random-access point in the stream.

b) A data packet is decodable independently from previous data packets if its packetDependency is set to 0. Consequently, the packetDependency of every data packet of an MIHS unit with MIHSUnitSync =0 is set to 0.

There are two ways to create a streamable haptic experience:

1) Assuming the content creation wants to ensure that applications can decode haptic streams every random-access period (say *n* ms), the content editor adds random access point every *n* ms to the haptic

stream to ensure independently decodable MIHS packets. At every n ms, an MIHS unit (MIHSUnitType == UNITTYPE_TEMPORAL) with the MIHSUnitSync flag indicating an independently decodable unit is introduced in the stream (see subclause 7.2.2). The first data packed in that unit has a packetDependency flag that indicates that the packet us independently decodable (see subclause 7.2.15).

2) The content has been created and stored in a different format, and no regular or specific random access has been planned. It may correspond to an experience created for file-based systems where content is played back from the start of the stream. In that case some transcoding of the original content is done (usually during the content preparation process of traditional distribution pipelines and before encoding). This transcoding aims at creating periodic random-access pointsThese random access point are signaled as being independently decodable and grouped into MIHS units that are also signaled as independently decodable.

8 Processing model

8.1 Overview

This Clause details the processing model of the codec. It describes every component of the encoder and decoder architecture presented in Figure 1. While the encoding algorithm proposed is globally informative to provide a typical encoding scheme, some processing parts are normative, especially when the decoder needs to perform the inverse processing.

This Clause details the processing model for both the PCM and descriptive input formats.

8.2 Encoder (informative)

8.2.1 Encoder architecture

Figure 30 depicts the encoder architecture in detail. The encoder is able to process three types of input files: OHM metadata files, descriptive haptics files (.ivs, .ahap and .hjif) or waveform PCM files (.wav). The behaviour of the encoder for each of these inputs is detailed respectively in subclauses 8.2.2, 8.2.3 and 8.2.4. The encoder supports two types of output coded representation formats: the interchange format detailed in Clause 6 and a packetized binary format detailed in Clause 7.

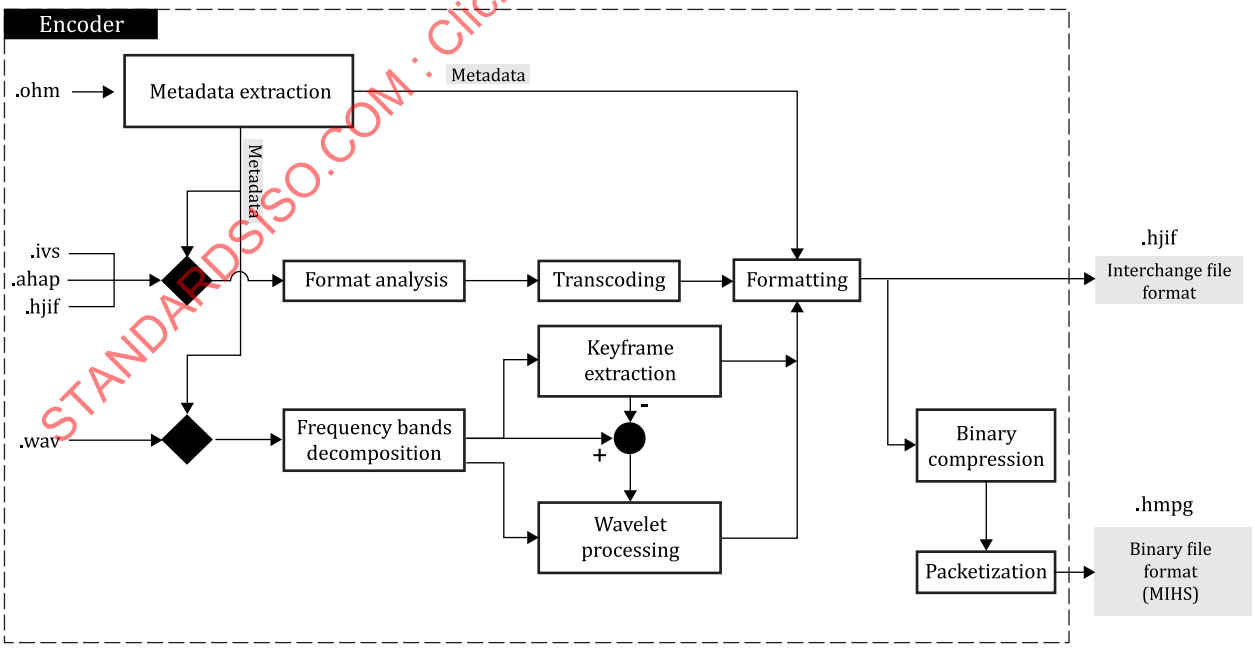


Figure 30 — Encoder architecture

8.2.2 OHM metadata input file

This metadata file contains a description of the haptic system and setup. In particular, it provides the name of each associated haptic file (either descriptive or PCM) along with a description of the signals. It also provides a mapping between each channel of the signals and the targeted body parts on the user's body. The body mapping used in the OHM file is identical to the one detailed in Table 7, subclause 5.6.2. The syntax of the OHM metadata file is given in Annex B.

When an OHM file is given as input, the encoder performs two operations:

— For each haptic_element in the OHM file, it retrieves the associated haptic file from the URI (given in the haptic_element_file_name field) and encodes it based on its type (see subclauses 7.2.9 and 7.2.14). An OHM file may reference multiple haptic files of different types.

— It extracts metadata information from the OHM file and maps it to metadata information of the data model presented in Clause 5. The mapping of the metadata between the two formats is detailed in Table 64.

Table 64 — Mapping of the fields from an OHM file to the proposed data model

OHM field	Data model component	Property
description_string	Experience (subclause 5.2)	description
element_description_string	Perception (subclause 5.4)	description
channel_description_string	Channel (subclause 5.6)	description
channel_gain	Channel (subclause 5.6)	gain
body_part_mask	Channel (subclause 5.6)	body part mask

8.2.3 Descriptive input files

Descriptive input files are encoded through a simple process. The encoder first identifies the input format. If the input format is a HJIF file, then no transcoding is necessary, the file can be further edited and eventually packetized into the binary format. It should be noted that when encoding a HJIF input file into the compressed binary format, the conversion may not be lossless due to the limited number of bits allocated for each property. The complete list of affected properties and their relative limitations is provided in ISO/IEC 23090-33 on conformance and reference software for haptics. If AHAP or IVS input files are used, a transcoding is necessary. The encoder first analyses semantically the input file information and transcodes it to be formatted into the data model presented in Clause 5. The transcoding process for AHAP and IVS files is detailed in subclause 8.2.5. After transcoding, the data can be exported as a .hjif interchange file or a .hmpg binary file (MIHS bitstream).

8.2.4 PCM input file

PCM (.wav) files cannot be transcoded directly into the output format. A signal analysis is performed to interpret the signal structure and convert it into the selected encoded representation. The different encoding types provide different ways to encode the input PCM data (i.e., curve, transient, vectorial and wavelet). In this informative part a hybrid approach is proposed to overcome the limitations of traditional frequency-based analysis processes on low frequencies. The encoding is split into two sub-processes. After performing a frequency band decomposition (detailed in subclause 8.2.6) to separate the low frequencies from the high frequencies of the signal, the first process extracts the local extrema from the low frequencies and stores them as keyframes in a curve band. This process is detailed in subclause 8.2.7. This first frequency band is then reconstructed using the appropriate interpolation method (as done at the synthesizer), and the residual error with the original signal is computed. This residual error is then added to the original high frequency bands. Finally, the second process applies a wavelet transformation on the high frequency spectrum (completed with the residual error). The wavelet processing is detailed in greater length in subclause 8.2.8. Other optimized encoding algorithms can be defined based on the normative tools.

In the case that several low frequency bands are used, the residual errors from all the low frequency bands are added to the high frequency band before encoding. In the case that several high frequency bands are

used, the residual errors from the low frequency band(s) are added to the first high frequency band before encoding. The user may choose to encode the input PCM signal using only one frequency band coding structure and a specific band type.

The following paragraphs provide an informative description of how to achieve rate-scalability with wavelet encoding. If a specific target bitrate is desired, encoder parameters can be adjusted. In general, it is possible to design a custom encoder or at least a function that determines the encoder parameters based on the specified target bitrate. This can be done without having to change normative parts of the codec. For PCM files, a parametrization function is proposed that can be used if the signal is encoded only with the wavelet coding module. It determines the bit budget for the wavelet quantization while all other parameters are kept constant, and takes the target bitrate as input. The basis to develop this function was a range of training files. They were encoded using bit budgets ranging from 1 to 135 kbits/s to obtain the resulting bitrates. The maximum bitrate over all files for each bit budget was then calculated. These maximum values were used to fit a cubic function as approximation such that any desired target bitrate can be taken to calculate the bit budget.

The determined function is

$$f(x) = -6.506x^3 + 3.433x^2 - 0.04421x + 0.0002573.$$

x denotes the target bitrate in kbits/s. The output of this function is rounded to the next lower integer and limited to the range [1,(log2(blockLength)−1)*15] such that a valid bit budget is set.

If desired, this concept can be extended to an optimization algorithm that adapts parameters to the input signal, for example, by iteratively encoding the signal and changing parameters after each iteration.

8.2.5 Transcoding descriptive content

8.2.5.1 Overview

Descriptive haptic codecs store parametrized commands allowing synthesizer or rendering software to reproduce the expected output. The specified descriptive format provides the necessary commands to transcode any of the three input formats to the coding format of this specification. In the current specification, three types of descriptive files can be ingested and transcoded: AHAP, IVS and HJIF files.

Any mapping from the input format to the output format may be designed. Typical existing implementations use mainly vectorial and transient effects, or a combination thereof to describe haptic effects. Other effect types can also be used to describe more complex or specific haptic effects.

AHAP and IVS are existing proprietary formats to describe haptic effects currently supported. The current specification has been designed to ingest those descriptions and translate them to the MPEG descriptive format, HJIF. But it is not mandatory to describe an effect using those formats. The HJIF format can also be used directly as input.

8.2.5.2 IVS

The IVS codec represents the expected haptic output by a set of basis effects defined by a set of parameters:

— An attack: duration and level

— A fade: duration and level

— An effect duration

— A period duration

— A wave type

Figure 31 provides an example of an IVS basis effect.

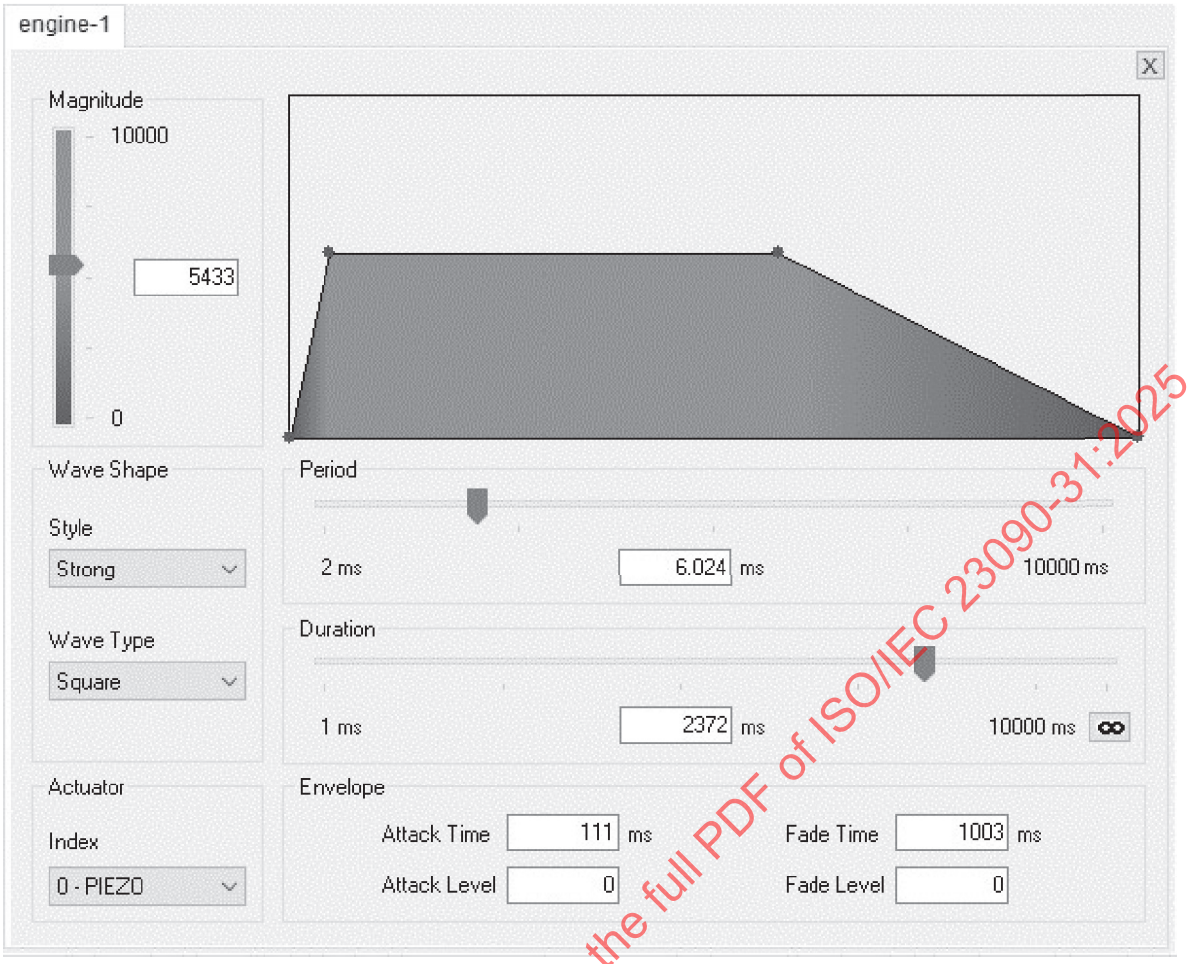


Figure 31 — Effect extracted from an IVS file

Then a timeline is defined in which two types of commands can be raised:

— A launch event which will play a referenced basis effect at a specific time and can override some of the basis effect's properties

— A repeat event which will create a loop inside the timeline to play the basis effects raised inside the repeated interval multiple times

The MPEG haptic encoder will ingest the timeline of the IVS file to transcode it following these simple rules:

For each launch event:

— Find the corresponding basis effect.

— Create an effect with matching waveform, magnitude and frequency. The frequency is computed by $f = \frac{1}{0.001 * \lambda}$ with λ the period length in milliseconds.

— Modulate the effect amplitude to match the configured attack or fade, if any. A linear interpolation is needed here. At least two keyframes will be stored in the effect. If an attack or a fade is configured, one or two keyframes will be inserted inside the effect to recreate a linearly interpolated amplitude at either the beginning or ending of the effect, or both.

— Find a vectorial wave band in which no effect is overlapping the created one. If none exists, create a new vectorial wave band with 0 to 1000 Hz as frequency range.

— Insert the new effect inside the band.

For each repeat effect:

— Find every effect after the repeat range and delay it.

— Find every effect in the repeat range and copy it.

8.2.5.3 AHAP

The AHAP codec encodes the design intent of the haptic experience based on amplitude and sharpness properties organised in continuous signals and transients.

— Continuous: parametrized sine function configured by:

  — Time: the timestamp at which the signal will be rendered.

  — Duration: the duration of the signal.

  — Intensity: the amplitude of the signal.

  — Sharpness: a value between 0 and 1, where 0 is the minimum frequency of the device and 1, its maximum frequency.

— Transients: parameterized API call configured by:

  — Time: the timestamp at which the transient will be raised.

  — Intensity: the amplitude of the transient.

  — Sharpness: a value between 0 and 1, where 0 is the minimum frequency of the device and 1, its maximum frequency.

After these primitive haptic effects are defined, the designer can modulate the amplitude and frequency of the whole experience by modulation functions composed of a set of keyframes.

The haptic encoder proposed will ingest the .ahap file to transcode it following these simple rules:

— For each transient:

  — Create a transient effect composed of one keyframe.

  — Fill the keyframe amplitude with the transient intensity.

  — Fill the keyframe frequency with the transient sharpness. The sharpness will be transformed and remapped from the original range [0,1] into the frequency band range which will be configured as [65,300] Hz.

— For each continuous:

  — Create a vectorial effect with a sine waveform.

  — The modulated parameters of the continuous will be stored in keyframes so two keyframes will be generated by default corresponding to the beginning and end of the effect. Then, the keyframes can be modified and others can be added in this range to recreate the design intention drawn by the different modulation functions of the original AHAP file.

  — Find a vectorial wave band in which no effect is overlapping the created one. If none exists, create a new vectorial wave band with [65,300] Hz as frequency range.

  — Store the effect in the previously found band.

8.2.6 Frequency band decomposition

The optional frequency band decomposition splits the signal into a low frequency and a high frequency band. In the proposed encoding implementation, this is done using Butterworth highpass and lowpass filters of

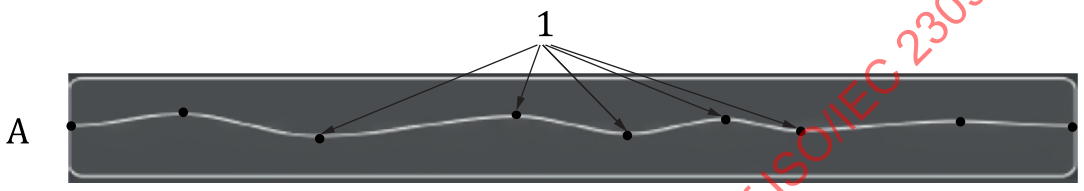
order 8 in a forward and a consecutive backward pass for zero phase filtering. It follows the scipy.signal.filtfilt method from the scipy Python library.[1] For filter coefficients, the function scipy.signal.butter[1] can be used as reference since the coefficients will change based on the chosen sampling frequency and cutoff frequency. This filter is not normative so different filters can be utilized.

It is recommended to set the cutoff frequency in a range of 20 to 72.5 Hz, or to 0 Hz, which is equivalent to using only the wavelet band for the complete signal.

The number of low and high frequency bands is not fixed; however, a common practice is to use two as described here as it provides a good efficiency-complexity tradeoff. The user can select other configurations.

8.2.7 Keyframe extraction for low frequencies processing

The optional curve band processing takes the lower frequency band from the frequency band decomposition and analyses its content in the time domain. An algorithm of local extremum extraction is proposed and applied to the signal to extract points defined by their timestamps and amplitude. Each one of these points is then encoded into keyframes composing a unique effect on a curve band (see Figure 32).



Key

1 keyframes

A band (*curve* band)

Figure 32 — Curve bands processing

The proposed algorithm used for this keyframe extraction is described in Table 65, where the input filteredSignal is a list of samples extracted and filtered by the method described in subclause 8.2.6 and sampleRate is the sample rate of the signal previously mentioned.

Table 65 — Pseudocode for the curve band encoding

Syntax

EncodeCurveBand(filteredSignal, sampleRate)
{
  encodedEffect = new Effect();
  encodedEffect.insertKeyframeAt(0, 0);
  for (i = 1; i < filteredSignal.size() - 1; i++) {
  lastSample = filteredSignal[i - 1];
  currentSample = filteredSignal[i];
  nextSample = filteredSignal[i + 1];
 	
  isFlat = lastSample == currentSample and currentSample == nextSample;
  isMinima = lastSample >= currentSample and currentSample <= nextSample;
  isMaxima = lastSample <= currentSample and currentSample >= nextSample;
  if(not isFlat and (isMinima or isMaxima)) {
  timestamp = 1000 * i / samplerate;
  amplitude = currentSample;
  encodedEffect.insertKeyframeAt(timestamp, amplitude);

Table 65 *(continued)*

Syntax

}
}
encodedEffect.insertKeyframeAt(1000 * filteredSignal.size() / sampleRate, 0);
encodedBand = new Band();
encodedBand.insert(encodedEffect);
return encodedBand;
}

8.2.8 Wavelet encoding

8.2.8.1 Encoding overview

The optional wavelet band (wavelet wave band type) processing of the encoder takes the high frequency band from the frequency band decomposition and the low frequency residual, and splits it into blocks of equal size, which is called blockLength. The blockLength is required to be a power of 2 and at least 16 samples. It is typically set to 1024.

The signal block is then analysed in the psychohaptic model. The lossy compression is achieved by first applying the wavelet transform to the block and then quantizing it, aided by the psychohaptic model. Each block is saved into a separate effect in a single band. This operation is identified as formatting. Finally the binary compression is applied using the SPIHT algorithm and Arithmetic Coding (AC).

The bitrate scaling is achieved by adjusting the bit budget parameter as detailed in subclause 8.2.4.

8.2.8.2 Wavelet transformation

The signal block is wavelet transformed using CDF9/7 filters.[2] Their coefficients are shown in Table 66.

Table 66 — Coefficients of the wavelet filters

N	LP	HP	LP_reconstruction	HP_reconstruction
−4	0.037828455506995			−0.037828455506995
−3	−0.023849465019380	−0.064538882628938	−0.064538882628938	−0.023849465019380
−2	−0.110624404418423	0.040689417609559	−0.040689417609559	0.110624404418423
−1	0.377402855612654	0.418092273222212	0.418092273222212	0.377402855612654
0	0.852698679009404	−0.788485616405665	0.788485616405665	−0.852698679009404
1	0.377402855612654	0.418092273222212	0.418092273222212	0.377402855612654
2	−0.110624404418423	0.040689417609559	−0.040689417609559	0.110624404418423
3	−0.023849465019380	−0.064538882628938	−0.064538882628938	−0.023849465019380
4	0.037828455506995			−0.037828455506995

The transformation separates the block into a higher and a lower frequency band and applies down-sampling with factor 2 on both bands. This is iteratively repeated for the lowest band. The number of iterations, or dwtlevels, is calculated by

$$\log_2(blockLength) - 2,$$

where *blockLength* is the length of the signal block to process. The iterative filtering leads to increasing band size from low to high frequencies, with each band twice as large as the next lower one. Figure 33 illustrates this splitting of coefficients into sub-bands for 32 samples total size of the block.

4	4	8	16

Figure 33 — Wavelet block coefficients in sub-bands.

8.2.8.3 Psychohaptic model

The psychohaptic model is used to evaluate the perceptual importance of each wavelet band. This component is informative since it only steers the quantization of wavelet coefficients and can be replaced by any other model. Masking in the frequency domain and the frequency dependence of the sensitivity of the human sense of touch are assessed. Masking in the frequency domain occurs when content with higher amplitude overshadows other components of the signal with lower amplitude and similar frequency.

To model this phenomenon, first all local maxima, also called peaks, are detected in the FFT spectrum (logarithmic domain, root power). To get as many frequency bins as input samples, the signal block is zero padded to twice its size before the FFT. Then, some of the peaks are sorted out. Peaks with an amplitude at least 45 dB below the maximum amplitude in the spectrum are omitted as well as peaks with a prominence below 12 dB. The prominence is obtained by first finding the two surrounding local minima on the left and right side of the peak and then calculating the amplitude difference between the peak and the larger one of the minima. For each remaining peak with index p a masker function $m_p(f)$ is generated. It is a polynomial function centred at the frequency of the local peak and calculated using the formula:

$$m_p(f) = a_p - 5\,\mathrm{dB} + 5\,\mathrm{dB}\frac{2f_p}{f_S} - \frac{30\,\mathrm{dB}}{f_p^2}\left(f - f_p\right)^2,$$

where f_S is the sampling frequency and f_p the centre frequency of the peak. a_p is the amplitude of the spectrum at f_p. The masking threshold is then calculated by taking the maximum at each frequency as:

$$m_{max}(f) = max_p\left(m_p(f)\right).$$

P denotes the quantity of peaks.

The frequency dependence of the human perception is modelled by the absolute threshold of perception $t(f)$, calculated by:

$$t(f) = \left| \frac{50\,\mathrm{dB}}{\left[\left(\frac{6}{11}\right)\right]^2}\left[\left(\frac{f}{550\,\mathrm{Hz}} + \frac{6}{11}\right)\right]^2 \right| - 45\,\mathrm{dB}.$$

Additionally, this function is limited to 0 dB. This way, high frequency content will not be ignored in the evaluation and is a measure similar to the threshold of pain from the audio domain. There it is assumed that when a high frequency signal reaches a certain amplitude, it will be painful and therefore perceivable.

The masking threshold and absolute threshold of perception are then combined to the global masking threshold. This is done by addition in the linear domain:

$$m_{global}(f) = 10^{m_p(f)/10} + 10^{m_{max}(f)/10}.$$

As the last step, a Signal-to-Mask-Ratio (SMR) is calculated. For this, the signal energy $E_{sig,b}$ for each wavelet band b and the mask energy $E_{mask,b}$ are calculated. This is done by summing up all coefficients of the input signal block and the mask corresponding to the respective band (linear domain). The SMR then calculates as

$$SMR_b = 10\log_{10}\frac{E_{sig,b}}{E_{mask,b}}.$$

8.2.8.4 Quantization

The quantization is performed using an embedded quantizer with individual bit depths for each wavelet band. The embedded property guarantees that the quantization steps for low bit depths are also present when using a higher bit depth.

The quantizer model used for the wavelet quantization is a uniform quantizer. The quantization step size Δ is:

$$\Delta = \frac{\widehat{w_{max}}}{2^b},$$

where $\widehat{w_{max}}$ is the quantized maximum wavelet coefficient of the signal block and b is the bit depth in the band. The quantization is performed using:

$$\hat{w} = sgn(w) \cdot floor\left(\frac{w}{\Delta}\right).$$

Here, $\hat{w}$ is the quantized version of wavelet coefficient w .

To determine the bit depth for each wavelet band, a bitBudget is iteratively distributed on the bands. This bit budget represents the sum of bit depths over the wavelet bands. The bit budget is used to control the quality of the encoded signal and also scales the resulting length of the bitstream and therefore the compression ratio. In each iteration, the band with the lowest Mask-to-Noise-Ratio (MNR) is found and a bit is allocated to it. The MNR is calculated from the SMR and the Signal-to-Noise-Ratio (SNR) by

$$MNR_b = SNR_b - SMR_b.$$

The bit allocation is only informative since other methods to determine the bit depths can be found. The proposed encoding algorithm is detailed in Table 67.

Table 67 — Pseudocode of the bit allocation algorithm

Syntax

bitAllocation(w, bitBudget, w_max)
{
  bitalloc_sum = 0;
  w_quant = zeros_like(w);
  while(bitalloc_sum < bitBudget) {
  index = argmin(MNR);
  bitalloc[index] += 1;
  bitalloc_sum += 1;
  w_quant = quantization(w, w_quant, index, bitalloc[index]);
  SNR = updateSNR(w, w_quant, SNR, index);
  MNR[index] = SNR[index] – SMR[index];
  if(bitalloc[index] >= 15) {
  MNR[index] = INFINITY;
  }
  }
}

w refers to the input signal block in the wavelet domain, bitBudget to the maximum sum of allocated bits. w_max is the absolute maximum wavelet coefficient in the specific block and w_quant is the output block, which is the quantized version of the input block.

zeros_like(w) refers to a function that generates a vector containing zeros with the length of vector w. quantization(w, w_quant, index, bitalloc[index]) quantizes the input w in band index using bitalloc[index] bits and updates this specific band in w_quant. updateSNR(w, w_quant, SNR, index) updates the SNR of w_quant relative to w in band index.

As the last step of the quantization, the signal is scaled to values in the range [–1,1] by:

$$\hat{w} = \frac{\hat{w}}{w_{max}},$$

and then the data are saved in a new effect including the quantized wavelet coefficients scaled to the range [–1,1], the quantized original maximum wavelet coefficient and the maximum allocated bits over all bands. The relation between w_quant and wavelet_coefficients is that w_quant is scaled to the original amplitude while wavelet_coefficients is the version scaled to [–1,1]. So, the new effect contains the lossy compression that is applied on the signal and can directly be used for the lossless binary compression.

8.2.8.5 Wavelet band binary compression

The binary compression for wavelet bands processes each quantized effect in the wavelet band and transforms it into a bitstream in a lossless fashion. The goal is to get a stream that has compact structure and can be transmitted. It consists of an 1D-SPIHT coder and an Arithmetic Coder (AC). The signal has to be scaled first from the floating-point values in the range [–1,1] to integer values before the compression can be applied. This is necessary to have a quantization step of one. This is done by multiplying it with $2^n - 1$, where n is the maximum bit depth used in quantization.

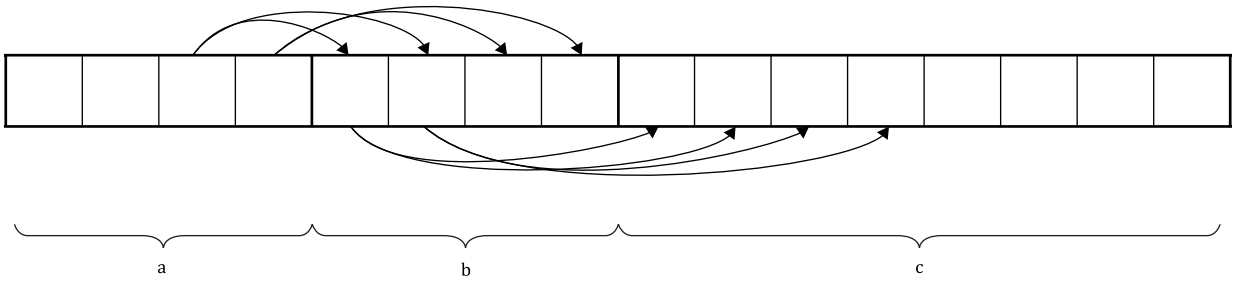
8.2.8.6 SPIHT

Set Partitioning In Hierarchical Trees (SPIHT) is an algorithm based on Embedded Zerotree Wavelet (EZW) coding and was introduced in.[3] It exploits the self-similarity of the wavelet coefficients across the bands and bitplanes and operates iteratively on each bitplane of the signal starting with the most significant bit. Its output is a binary stream.

The n^{th} bitplane consists of all n^{th} bits of the samples in a signal, with indexing starting at n=0 for the least significant bit. The signs of the coefficients are treated separately. The algorithm consists of a significance pass that finds significant coefficients in the block, and a refinement pass that only codes the next lower bitplane of the significant samples in each iteration.

Significance is a notion that separates larger and more important coefficients from the others, which can be omitted. The exact formulation for it will be presented later in this subclause. In both the significance pass and the refinement pass, bits are written to a bitstream. The bits enable the decoder to reconstruct the data.

In 1D SPIHT, coefficients are arranged in a tree structure, where each coefficient is represented by a node and the two coefficients with the same spatial orientation in the next lower band are set to be the children, or offspring, of that node. The resulting structure is called spatial orientation tree. In general, for natural signals, it is likely that coefficients with the same spatial orientation are similar across sub-bands. Also, the signal energy tends to decrease to higher frequencies. So, using the spatial orientation tree results in good magnitude ordering of the coefficients. This tree is further explained in,[3] extended to the 2D case, and illustrated in Figure 34.



a Band 0.
b Band 1.
c Band 2.

Figure 34 — SPIHT spatial orientation tree

The algorithm, reduced to the 1D case, uses the following sets to implement the spatial orientation tree:

— $O(i)$ set of indices of all offspring of node i;

— $D(i)$ set of indices of all descendants of node i;

— H set of indices of all spatial orientation tree roots;

— $L(i) = D(i) - O(i)$.

$O(i)$ is obtained using

$$O(i) = \{(2i),(2i+1)\}.$$

The partitions for the algorithm are initialized with the sets $\{(i)\}$ and $D(i)$ for all $i \in H$. If $D(i)$ is found to be significant during a sorting pass, it is partitioned into $L(i)$ and two single-element sets with $k \in O(j)$. If $L(i)$ is significant, it is divided into two sets $D(k)$ with $k \in O(i)$.

To determine the significance $S_n(\tau)$ in bitplane n of a set of samples τ, the expression

$$\{c_\tau\} \geq 2^n$$

is evaluated. $Sn(\tau)$ is 1 if the expression is valid, otherwise 0. Effectively, this is carried out as a check for ones in the respective bitplane n in the concrete algorithm. If $S_n(\tau)$ is 0, τ is called insignificant. Sets that are insignificant over all bitplanes are not coded in the refinement pass, so coding bits are saved for those sets. Due to the spatial orientation tree, it is likely that the significance is similar for whole sets and therefore good compression performance can be reached.

During the algorithm, three ordered lists are used: the list of insignificant sets (LIS), list of insignificant pixels (LIP) and list of significant pixels (LSP). The term pixels originates from image processing and correlates to coefficients in the case of wavelet transformed vibrotactile signals. In the LIP and LSP, single coefficients are saved, and in the LIS, sets of coefficients from set $L(i)$ or $D(i)$ are put. Coefficients in the set $D(i)$ are called type A, and type B if they are from $L(i)$.

In each iteration of the algorithm, first the significance is checked in the sorting pass. Pixels that were insignificant so far are in the LIP, therefore the LIP is checked in each iteration. If they are found to be significant, they are put in the LSP and their sign is transmitted. Sets are also checked and partitioned if they are found to be significant. If the newly partitioned sets are single coefficients, they are put in the LIP instead.

In the refinement pass, the current bitplane is transmitted for the pixels in the LSP, excluding those added in the latest sorting pass, because they are already known to have a 1 in that plane from the significance check.

The bits in the stream serve different purposes as explained above, so they have different so-called contexts. These contexts are written to a separate output stream and sent to the AC alongside the bitstream. The bits can either be significance bits, sign bits, refinement bits or side information bits. Since the significance is checked for different cases, it can be further subdivided. This results in the following list of possible contexts:

— significance: LIP;

— significance: LIS, type A;

— significance: descendants of type A in LIS;

— significance: LIS, type B;

— sign bits;

— refinement bits;

— side information bits.

The pseudocode detailed in Table 68 describes the functionality of the SPIHT encoder, which is informative. It generates two temporary streams: SPIHT_stream and context. The variable wavelet_coefficients refers to the quantized wavelet coefficients arranged in an array. They are originally in the range [–1,1], but scaled to integer values in the beginning of the function. dwtlevel is the number of wavelet transformation iterations, and maxallocbits is the maximum bit depth of the wavelet coefficients.

Table 68 — Pseucode of the SPIHT encoder

Syntax	No. of bits	Mnemonic
SPIHT_encode(wavelet_coefficients, dwtlevel, bitwavmax, maxallocbits, SPIHT_stream, context) {		
length = wavelet_coefficients.size();		
for (i = 0; i < length; i++) {		
wavelet_coefficients[i] = wavelet_coefficients[i] * (2^n-1);		
}		
SPIHT_stream.push_back(maxallocbits);	**4**	**uilsbf**
SPIHT_stream.push_back(bitwavmax);	**8**	**blsbf**
context.append_zeros(12); // 4 for maxallocbits, 8 for bitwavmax	12	**islif**
int bandsize = 2 << (log2(length) - dwtlevel);		
for (i = 0; i < bandsize; i++) {		
LIP.push_back(i);		
}		
for (int i = (bandsize / 2); i < bandsize; i++) {		
LIS1.push_back(i);		
LIS2.push_back(0);		
}		
initMaxDescendants(wavelet_coefficients);		
n = maxallocbits;		
while (0 <= n) {		
compare = 1 << n;		
LSP_index = LSP.size();		
for (it = LIP.begin(); it != LIP.end();) {		
if (abs(in[LIP[it]]) >= compare) {		

Table 68 *(continued)*

Syntax	No. of bits	Mnemonic

addToOutput(1, 2, SPIHT_stream, context);
addToOutput(in[LIP[it]] >= 0, 1, SPIHT_stream, context);
LSP.push_back(it);
it = LIP.erase(it);
} else {
addToOutput(0, 2, SPIHT_stream, context);
it++;
}
}
 	
it1 = LIS1.begin();
it2 = LIS2.begin();
LISsize = LIS1.size();
for (i = 0; i < LISsize; i++) {
if (in[LIS2[it2]] == 0) {
int max_d = maxDescendant(LIS1[it1], LIS2[it2]);
if (max_d >= compare) {
addToOutput(1, 3, SPIHT_stream, context);
y = LIS1[it1];
index = 2 * y;
if (abs(in[index]) >= compare) {
LSP.push_back(index);
addToOutput(1, 4, SPIHT_stream, context);
addToOutput(in[index] >= 0), 1, SPIHT_stream, context);
} else {
addToOutput(0, 4, SPIHT_stream, context);
LIP.push_back(index);
}
index = 2 * y + 1;
if (abs(in[index]) >= compare) {
LSP.push_back(index);
addToOutput(1, 4, SPIHT_stream, context);
addToOutput(in[index] >= 0), 1, SPIHT_stream, context);
} else {
addToOutput(0, 4, SPIHT_stream, context);
LIP.push_back(index);
}
if ((4 * y + 3) < length) {
LIS1.push_back(LIS1[it1]);
LIS2.push_back(1);
LISsize++;
}
it1 = LIS1.erase(it1);
it2 = LIS2.erase(it2);		

Table 68 *(continued)*

Syntax	No. of bits	Mnemonic

} else {
addToOutput(0, 3, SPIHT_stream, context);
it1++;
it2++;
}
} else {
max_d = maxDescendant(LIS1[it1], LIS2[it2]);
if (max_d >= compare) {
addToOutput(1, 5, SPIHT_stream, context);
int y = LIS1[it1];
LIS1.push_back(2 * y);
LIS1.push_back(2 * y + 1);
LIS2.push_back(0);
LIS2.push_back(0);
LISsize += 2;
it1 = LIS1.erase(it1);
it2 = LIS2.erase(it2);
} else {
addToOutput(0, 5, SPIHT_stream, context);
it1++;
it2++;
}
}
}
 	
refinementPass(wavelet_coefficients, LSP, LSP_index, n, SPIHT_stream, context);
n--;
}
}		

Syntax	No. of bits	Mnemonic

refinementPass(data, LSP, LSP_index, n, outstream, context) {
it = LSP.begin();
temp = 0;
while (temp < LSP_index) {
s = bitget(floor(abs(data[LSP[it]])), n + 1);
outstream.push_back(s);
context.push_back(6);
temp++;
it++;
}
}
	1
	1
		vlclbf
		vlislif

Syntax	No. of bits	Mnemonic
addToOutput(bit, c, outstream, context) {		
outstream.push_back(bit);	**1**	**vlclbf**
context.push_back(c);	**1**	**vlislif**
}		

```
maxDescendant(j, type) {
    if (type == 1) {
        if (j >= maxDescendants1.size()) {
            return 0;
        }
        return maxDescendants1[j];
    }
    if (j >= maxDescendants.size()) {
        return 0;
    }
    return maxDescendants[j];
}
```

```
initMaxDescendants(signal) {
    length = signal.size();
    start = length >> 1;
    p1 = start;
    p2 = p1 + 1;
    target = start >> 1;
    for (i = 0; i < (start >> 1); i++) {
        v1 = abs(signal[p1]);
        v2 = abs(signal[p2]);
        if (v1 > v2) {
            maxDescendants[target] = v1;
        } else {
            maxDescendants[target] = v2;
        }
        p1 += 2;
        p2 += 2;
        target++;
    }
    width = start >> 1;
    p1 = width;
    p2 = p1 + 1;
    target = width >> 1;
    while (target > 1) {
        for (i = 0; i < (width >> 1); i++) {
```

```
            v1 = maxDescendants[p1];
            v2 = maxDescendants[p2];
            if (v1 > v2) {
                maxDescendants1[target] = v1;
            } else {
                maxDescendants1[target] = v2;
            }
            v1 = abs(signal[p1]);
            if (v1 > maxDescendants1[target]) {
                maxDescendants[target] = v1;
            } else {
                maxDescendants[target] = maxDescendants1[target];
            }
            v2 = abs(signal[p2]);
            if (v2 > maxDescendants[target]) {
                maxDescendants[target] = v2;
            }
            p1 += 2;
            p2 += 2;
            target++;
        }
        width = width >> 1;
        p1 = width;
        p2 = p1 + 1;
        target = width >> 1;
    }
}
```

The following function describes how the quantized maximum wavelet coefficient qwavmax is encoded, which is then carried over to the encode function of SPIHT as bitwavmax:

Syntax	No. of bits	Mnemonic
maximumWaveletCoefficient(qwavmax, bitwavmax) {		
i_part = 0;		
mode = 0;		
if (qwavmax < 1) {		
i_bits = 0;		
f_bits = 7;		
} else {		
i_part = 1;		
i_bits = 4;		
f_bits = 3;		
mode = 1;		
}		