**** **ISO/IEC JTC 1/SC 29/WG 2 N0072**

**ISO/IEC JTC 1/SC 29/WG 2**

**MPEG Technical requirements   
Convenorship: SFS (Finland)**

**Document type:** Output Document

**Title:** Encoder Input Format for MPEG Haptics

**Status:** Approved

**Date of document:** 2021-04-30

**Source:** ISO/IEC JTC 1/SC 29/WG 2

**Expected action:** none

**Action due date:** none

**No. of pages:** 19 (without cover page)

**Email of Convenor:** igor.curcio@nokia.com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg2>

**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 2**

**MPEG TECHNICAL REQUIREMENTS**

**ISO/IEC JTC 1/SC 29/WG 2 N0072**

**Online – April 2021**

|  |  |
| --- | --- |
| **Title:** | **Encoder Input Formats for MPEG Haptics** |
| **Source:** | **WG 2 MPEG Technical Requirements** |
| **Editors:** | **Yeshwant Muthusamy, Chris Ullrich, Jamal Saboune (Immersion Corporation)**  **Philippe Guillotel, Fabien Danieau, Quentin Galvane (InterDigital Corporation)**  **Camille Moussette (Apple, Inc.)**  **Gwydion ap Dafydd (Lofelt GmbH)** |

**Status: Approved**

**Serial number: 20229**

**Abstract**

This document contains detailed descriptions of the Encoder Input Formats (AHAP, IVS, PCM, and OHM) to be used in the Haptics CfP – Phase 1.

Table of Contents

[**Abstract** 1](#_Toc70678958)

[1 Introduction 2](#_Toc70678959)

[2 Object Haptic Metadata (OHM) file format 2](#_Toc70678960)

[3 PCM Signals 6](#_Toc70678961)

[4 Haptic Descriptors in JSON (AHAP) 7](#_Toc70678962)

[4.1 Define Patterns at the Top Level 8](#_Toc70678963)

[4.2 Build a Pattern from Events 8](#_Toc70678964)

[4.3 Example AHAP 10](#_Toc70678965)

[5 Parametric Effect Descriptors in XML (IVS) 11](#_Toc70678966)

[5.1 Details 11](#_Toc70678967)

[5.2 Basis Effects 12](#_Toc70678968)

[5.2.1 Magsweep Effects 12](#_Toc70678969)

[5.2.2 Periodic Effects 13](#_Toc70678970)

[5.3 Waveform Effects 14](#_Toc70678971)

[5.4 Media Effects 15](#_Toc70678972)

[5.5 Timeline Effects 16](#_Toc70678973)

[5.6 Interpolated Effects 18](#_Toc70678974)

[6 References 19](#_Toc70678975)

# Introduction

This document fully describes the input formats used for the MPEG Call for Proposals (CfP) on the Coded Representation of Haptics [1] evaluation:

* Object Haptic Metadata (OHM) - that can be used in conjunction with any of the following three input formats
* PCM signals encoded using WAV
* Haptic descriptors in JSON (i.e., AHAP file format)
* Parametric effect descriptors in XML (i.e., IVS file format)

This may form the basis of a more comprehensive input format to be used for the subsequent collaborative technology development process.

# Object Haptic Metadata (OHM) file format

Input data is provided through a list of haptic files. A first OHM metadata file contains a description of the haptic content and configuration. It provides the necessary information on the haptic experience and the associated input signal files. All the other input files contain the different signals and can be provided using any of the formats described in Sections 3, 4, and 5.

This metadata file contains a description of the haptic system and setup. In particular, it provides the name of each associated file 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 body. To perform this mapping, each body part of a user is associated with a binary mask as shown in Figure 1 and Table 1. These masks can be combined to define larger body surfaces as illustrated in Table 2. The syntax of the OHM metadata file is given below.

|  |  |  |
| --- | --- | --- |
| **Syntax** | **No. of bytes** | **Data format** |
| file\_description () {  format\_id\_string  format\_version  number\_of\_haptic\_elements  description\_string  for (i=0; i<number\_of\_haptic\_elements; i++) {  haptic\_element\_file\_name  element\_description\_string  number\_of\_haptic\_channels  for (i=0; i<number\_of\_haptic\_channels; i++) {  channel\_description\_string  channel\_gain  body\_part\_mask  }  }  } | 4  2  2  32    64  32  2    32  4  4 | char  unsigned int  unsigned int  char    char  char  unsigned int    char  single float  unsigned int |

format\_id\_string – unique character identifier “OHM”

format\_version – version number of the file format: 1.

number\_of\_haptic\_elements – number of haptic elements compiling the content. An element typically maps to an end-user haptic device.

description\_string – description string containing a human readable content description. If shorter than 32 bytes, it is followed by padding null characters. If the string is 32 bytes long, the string is terminated without a null character.

haptic\_element\_file\_name – description string containing the file name of the according haptic element file (.wav, .ivs, .ahap). If shorter than 64 bytes, it is followed by padding null characters. If the string is 64 bytes long, the string is terminated without a null character. Note that an element might include more than one channel. This file is assumed to be located in the same directory as the ohm file (i.e., same path).

element\_description\_string – description string containing a human readable content description. If shorter than 32 bytes, it is followed by padding null characters. If the string is 32 bytes long, the string is terminated without a null character.

number\_of\_haptic\_channels – number of simultaneous channels for each haptic element (up to 65535).

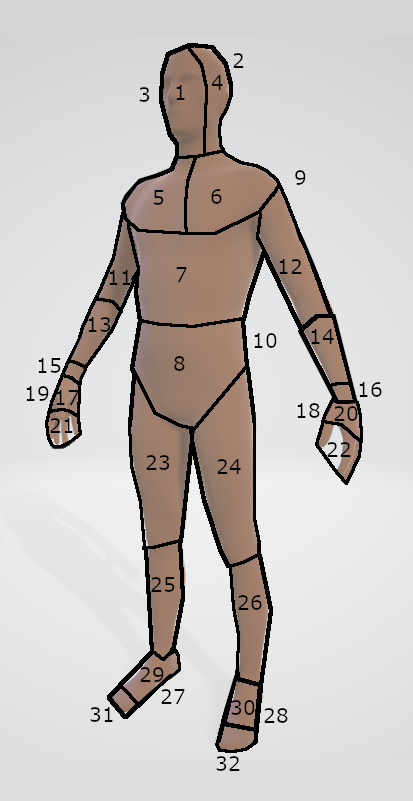
channel\_description\_string – description string containing a human readable content description. If shorter than 32 bytes, it is followed by padding null characters. If the string is 32 bytes long, the string is terminated without a null character.

body\_part\_mask – binary mask specifying the body part(s) on which to apply the effect.

channel\_gain – a single precision float value that describes the amplitude gain for the haptic track. A value of 1.0 indicates that the track should be rendered at nominal voltage. Higher values indicate an overdrive state for the actuator. Defaults to 1.0, if not specified when the OHM file is generated.

**Table 1: Body Part Masks**

|  |  |  |  |
| --- | --- | --- | --- |
| ***body part ID*** | **Name** | **body\_part\_mask** | **Hexa** |
| 0 | Unspecified | 00000000000000000000000000000000 | 0x00000000 |
| 1 | Head front | 00000000000000000000000000000001 | 0x00000001 |
| 2 | Head back | 00000000000000000000000000000010 | 0x00000002 |
| 3 | Head right | 00000000000000000000000000000100 | 0x00000004 |
| 4 | Head left | 00000000000000000000000000001000 | 0x00000008 |
| 5 | Right upper chest | 00000000000000000000000000010000 | 0x00000010 |
| 6 | Left upper chest | 00000000000000000000000000100000 | 0x00000020 |
| 7 | Abdomen | 00000000000000000000000001000000 | 0x00000040 |
| 8 | Waist | 00000000000000000000000010000000 | 0x00000080 |
| 9 | Upper back | 00000000000000000000000100000000 | 0x00000100 |
| 10 | Lower back | 00000000000000000000001000000000 | 0x00000200 |
| 11 | Right upper arm | 00000000000000000000010000000000 | 0x00000400 |
| 12 | Left upper arm | 00000000000000000000100000000000 | 0x00000800 |
| 13 | Right forearm | 00000000000000000001000000000000 | 0x00001000 |
| 14 | Left forearm | 00000000000000000010000000000000 | 0x00002000 |
| 15 | Right wrist | 00000000000000000100000000000000 | 0x00004000 |
| 16 | Left wrist | 00000000000000001000000000000000 | 0x00008000 |
| 17 | Right hand palm | 00000000000000010000000000000000 | 0x00010000 |
| 18 | Left hand palm | 00000000000000100000000000000000 | 0x00020000 |
| 19 | Right hand dorsum | 00000000000001000000000000000000 | 0x00040000 |
| 20 | Left hand dorsum | 00000000000010000000000000000000 | 0x00080000 |
| 21 | Right hand fingers | 00000000000100000000000000000000 | 0x00100000 |
| 22 | Left hand fingers | 00000000001000000000000000000000 | 0x00200000 |
| 23 | Right thigh | 00000000010000000000000000000000 | 0x00400000 |
| 24 | Left thigh | 00000000100000000000000000000000 | 0x00800000 |
| 25 | Right calf | 00000001000000000000000000000000 | 0x01000000 |
| 26 | Left calf | 00000010000000000000000000000000 | 0x02000000 |
| 27 | Right foot palm | 00000100000000000000000000000000 | 0x04000000 |
| 28 | Left foot palm | 00001000000000000000000000000000 | 0x08000000 |
| 29 | Right foot dorsum | 00010000000000000000000000000000 | 0x10000000 |
| 30 | Left foot dorsum | 00100000000000000000000000000000 | 0x20000000 |
| 31 | Right foot fingers | 01000000000000000000000000000000 | 0x40000000 |
| 32 | Left foot fingers | 10000000000000000000000000000000 | 0x80000000 |



**Figure 1: Body Part Segmentation**

**Table 2: Body Parts Combinations**

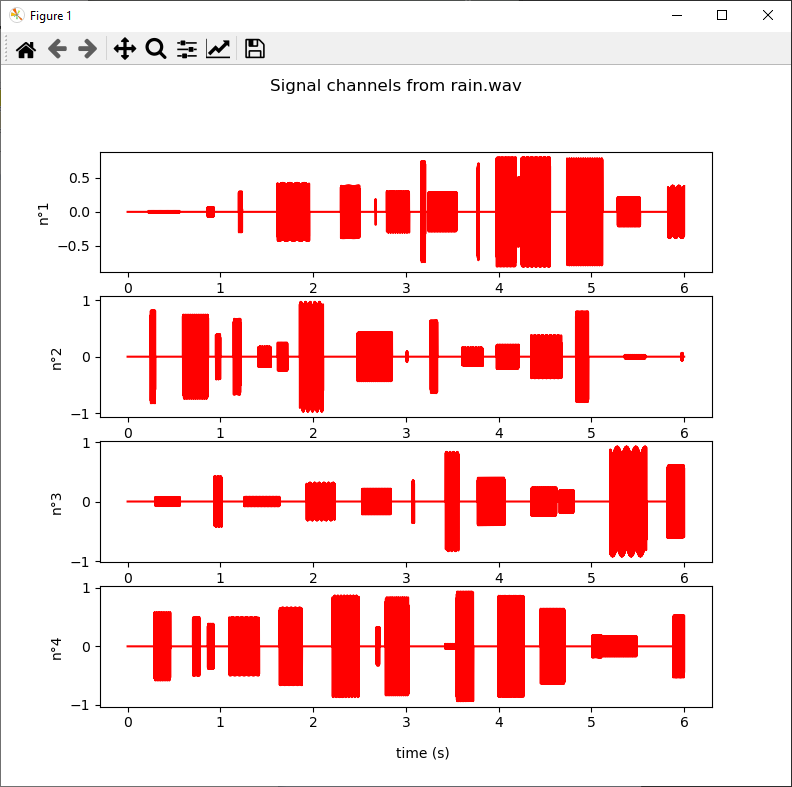
|  |  |  |
| --- | --- | --- |
| ***body part ID*** | ***body\_part\_mask*** | ***Hexa*** |
| Right arm | 00000000000101010101010000000000 | 0x00015540 |
| Left arm | 00000000001010101010100000000000 | 0x002AA800 |
| Right leg | 01010101010000000000000000000000 | 0x55400000 |
| Left leg | 10101010100000000000000000000000 | 0xAA800000 |
| Upper body | 00000000001111111111111111111111 | 0x003FFFFF |
| Lower body | 11111111110000000000000000000000 | 0xFFC00000 |
| Full body | 11111111111111111111111111111111 | 0xFFFFFFFF |

**Example: rain.ohm associated to the rain.wav effects file**

output of the text parser for rain.ohm:

|  |
| --- |
| format version: 1  number\_of\_haptic\_elements: 1  description string: rain effect  haptic\_object #: 1  haptic\_object\_file\_name: rain.wav  element description string: Vibration effect  number\_of\_haptic\_channels: 4  haptic channel #: 1  channel description string: head  channel gain: 1.0  body\_part\_mask: 0b1111  body parts: ['Head front', 'Head back', 'Head right', 'Head left']  haptic channel #: 2  channel description string: left arm  channel gain: 1.0  body\_part\_mask: 0b1010101010100000000000  body parts: ['Left upper arm', 'Left forearm', 'Left wrist', 'Left hand palm', 'Left hand dorsum', 'Left hand fingers']  haptic channel #: 3  channel description string: right arm  channel gain: 1.0  body\_part\_mask: 0b101010101010000000000  body parts: ['Right upper arm', 'Right forearm', 'Right wrist', 'Right hand palm', 'Right hand dorsum', 'Right hand fingers']  haptic channel #: 4  channel description string: upper chest  channel gain: 1.0  body\_part\_mask: 0b110000  body parts: ['Right upper chest', 'Left upper chest'] |

Content of rain.wav:



# PCM Signals

PCM signals are described using the Waveform Audio File Format. This file format standard is an instance of the Resource Interchange File Format (RIFF) and is a well-suited container for any type of uncompressed PCM signal such as haptic signals. It can be composed of several channels, each containing a different signal. Its syntax is described below:

|  |  |  |
| --- | --- | --- |
| **Syntax** | **No. of bytes** | **Data format** |
| wav\_format() {  file\_type\_bloc\_ID  file\_size  file\_format\_ID    format\_bloc\_ID  bloc\_size  audio\_format  number\_channels  frequency  bytes\_per\_seconds  bytes\_per\_bloc  bits\_per\_samples    data\_bloc\_id  data\_size  for (i=0; i< data\_size/number\_channels; i++) {  for (j=0; j< number\_channels; j++) {  data\_bytes  }  }  } | 4  4  4    4  4  2  2  4  4  2  2    4  4      bits\_per\_sample/8 | char  unsigned int  char    char  unsigned int  unsigned int  unsigned int  unsigned int  unsigned int  unsigned int  unsigned int    unsigned int  unsigned int      signed int |

file\_type\_bloc\_ID: constant «RIFF»(0x52,0x49,0x46,0x46)

file\_size: file size minus 8 bytes

file\_format\_ID: constant «WAVE» (0x57,0x41,0x56,0x45)

format\_bloc\_ID: constant «fmt␣» (0x66,0x6D, 0x74,0x20)

bloc\_size: number of bytes per bloc - 16 (0x10)

audio\_format: storage format in the file (1: PCM, ...)

number\_channels: number of channels

frequency: sampling frequency of the data (in Hertz)

bytes\_per\_seconds: number of bytes to read per seconds (i.e., frequency \* bytes\_per\_bloc).

bytes\_per\_bloc: number of bytes per bloc of samples (i.e., for all channels: number\_channels\*bits\_per\_sample/8).

bits\_per\_samples: number of bits used to code each sample (8, 16, 24)

data\_bloc\_id: constant «data»(0x64,0x61,0x74,0x61)

data\_size: number of data bytes

data\_bytes: data value normalized [-1,+1]

# Haptic Descriptors in JSON (AHAP)

The Apple Haptic and Audio Pattern (AHAP) descriptor is a JSON-like file format [2] that specifies a haptic pattern through key-value pairs, to be used with Apple products that support Core Haptics (CH).

In short, an AHAP contains one Pattern made of Events and the two basic event types are:

**Transient events**, which are brief, compact experiences that feel like taps or impulses, such as the experience of tapping the Flashlight button on the Home screen.

**Continuous events**, which feel like sustained vibrations, such as the experience of the lasers effect in a message.

Regardless of the building block you choose, you can also control its sharpness and intensity. By combining transient and continuous events, varying sharpness, and intensity, and including optional audio content, you can create a wide range of different haptic experiences.

An AHAP file does not need an entry for every key. When Core Haptics loads an AHAP file, it sets missing entries to their default value and clamps out-of-range values to their minimum or maximum values, whichever is closer. The Core Haptics framework ignores unexpected and unsupported keys.

The following paragraphs detail the primary keys and describes their effect on the resulting haptic pattern.

## Define Patterns at the Top Level

The only top-level keys are Pattern and Version. The value for Pattern is an array of subdictionaries. Each AHAP file can contain a single pattern. Version indicates the lowest version of Core Haptics that can support loading and playing the file. Later versions, indicated by a higher version number, may contain keys that are not supported in older versions of the framework.

**Pattern**

Array of Event dictionaries representing a segment of the haptics to play.

**Version**

The Core Haptics version that supports the file.

## Build a Pattern from Events

Each pattern contains one or more events, defined by the Event key at the top level of the pattern dictionary. An event is a segment of the pattern with some duration and a set of properties, such as intensity or sharpness. Each event starts at its own time and can overlap other events.

**Event**

Dictionary of event type, start time, duration, and optional event parameters.

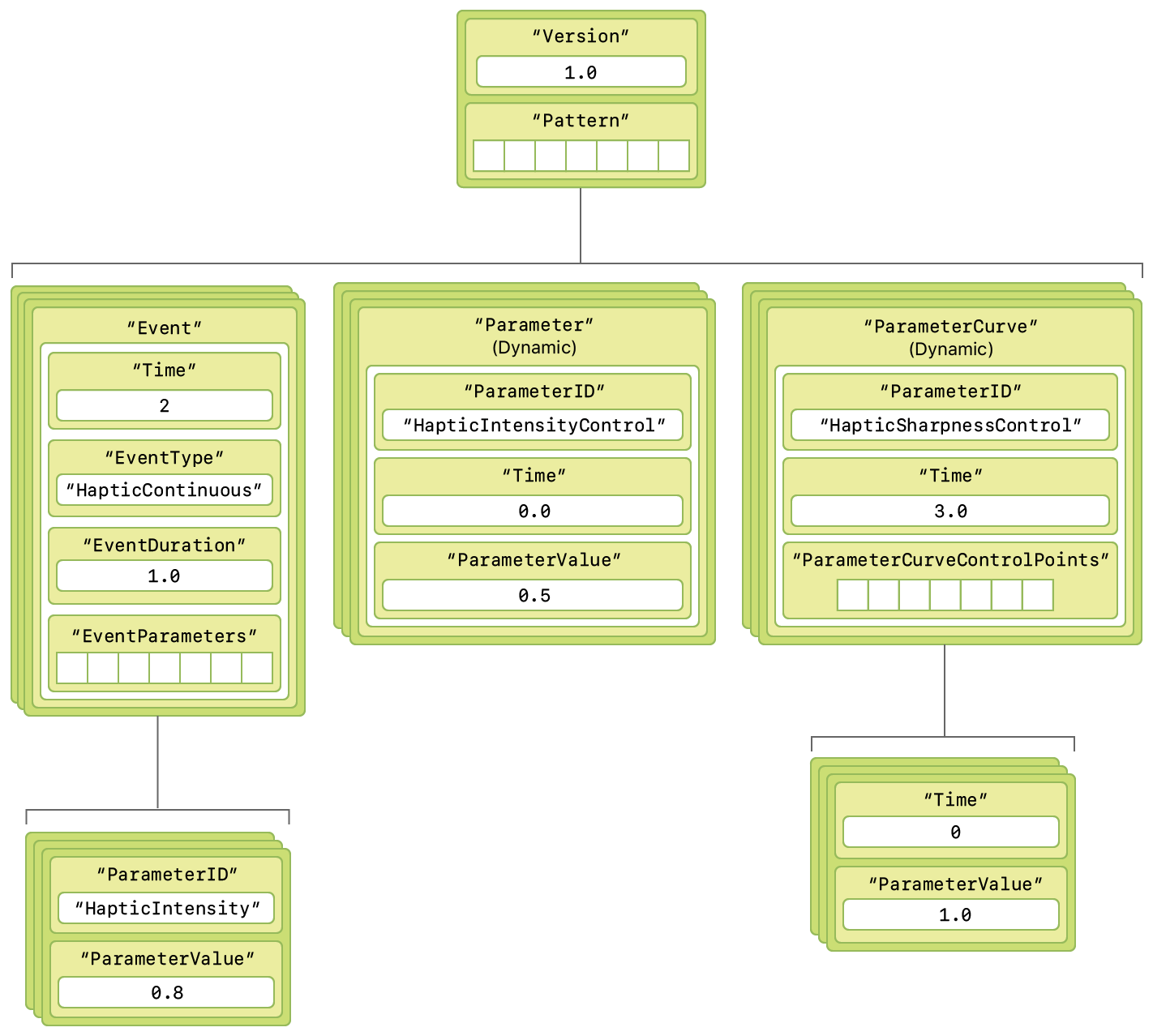
You can define two other keys in the pattern dictionary:

**Parameter**

A key in the pattern dictionary that you use to define a dynamic parameter.

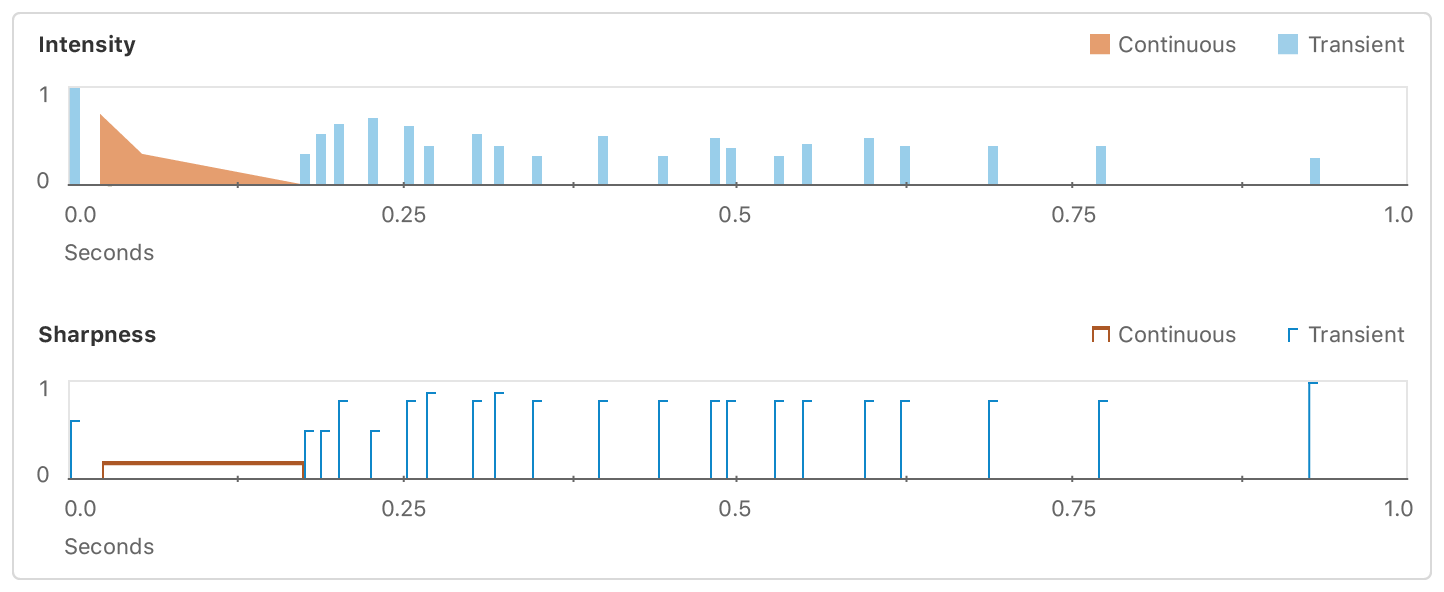
**ParameterCurve**

A key in the pattern dictionary that you use to define a parameter curve.



For additional details plus all the available keys and parameters, please refer to the Core Haptics documentation [2].

## Example AHAP



**Sparkle.ahap (abbreviated for clarity)**

{

"Version": 1.0,

"Pattern":

[

{

"Event":

{

"Time": 0.0,

"EventType": "HapticTransient",

"EventParameters":

[

{ "ParameterID": "HapticIntensity", "ParameterValue": 1.0 },

{ "ParameterID": "HapticSharpness", "ParameterValue": 0.6 }

]

}

},

{

"Event":

{

"Time": 0.024,

"EventType": "HapticContinuous",

"EventDuration": 0.150,

"EventParameters":

[

{ "ParameterID": "HapticIntensity", "ParameterValue": 0.6 },

{ "ParameterID": "HapticSharpness", "ParameterValue": 0.1 }

]

}

},

{

"ParameterCurve":

{

"ParameterID": "HapticIntensityControl",

"Time": 0.024,

"ParameterCurveControlPoints":

[

{ "Time": 0, "ParameterValue": 1.0 },

{ "Time": 0.025, "ParameterValue": 0.45 },

{ "Time": 0.15, "ParameterValue": 0.0 }

]

}

},

...

}

]

}

# Parametric Effect Descriptors in XML (IVS)

The IVS file format is an XML-based file format used by Immersion’s Haptic Studio to represent vibrotactile effects.

An IVS file defines zero or more vibrotactile effects. Because an IVS file can define multiple vibrotactile effects, it can be thought of as defining a library of vibrotactile effects. An application can play any defined effect by using an API that can play effects defined in the IVS file format.

Immersion’s software plays effects defined in IVS files by converting the IVS file contents to a proprietary IVT format, which is a binary representation of information present in the IVS file. IVS files are intended to be human-readable, whereas IVT files are intended to be more easily processed by machine.

## Details

An IVS file represents a collection of vibrotactile and audio effects. There are four types of effects further described below:

* Basis effects
* Waveform effects
* Media effects
* Timeline effects
* Interpolated effects

Example:

<?xml version="1.0"?>

<ivs-file last-modified="Monday, September 21, 2020 06:32:48PM">

<effects>

<basis-effect …/>

<waveform-effect …/>

<timeline-effect …>

…

</timeline-effect>

<media-effect …/>

<interpolated-effect …>

…

</interpolated-effect>

</effects>

</ivs-file>

## Basis Effects

Basis effects are simple, standalone effects. In other words, basis effects do not depend on other effects.

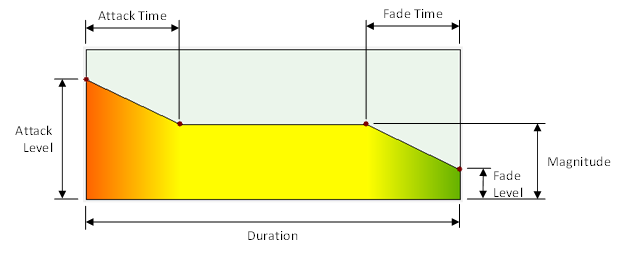
There are two flavors of basis effects:

* Magsweep effects
* Periodic effects

### Magsweep Effects

Magsweep effects define a vibration amplitude over time using a piecewise linear attack-sustain-fade envelope with the following parameters:

* Attack time – the duration of the attack phase
* Attack level – the amplitude at the start of the attack phase
* Magnitude level – the amplitude during the sustain phase. The amplitude changes linearly during the attack phase from the attack level to the magnitude level
* Fade time – the duration of the fade phase
* Fade level – the amplitude at the end of the fade phase. The amplitude changes linearly from the sustain level to the fade level during the fade phase



Magsweep effects also have a style parameter with the following possible values:

* Strong
* Smooth
* Sharp

The style affects how the effect is rendered on the target hardware, depending on the target actuator capabilities.

Magsweep effects also have an actuator ID parameter, or index, specifying the actuator on which to render the effect on a multi-actuator device.

Example:

<basis-effect name="MagSweep"

type="magsweep"

duration="1000"

magnitude="7500"

style="sharp"

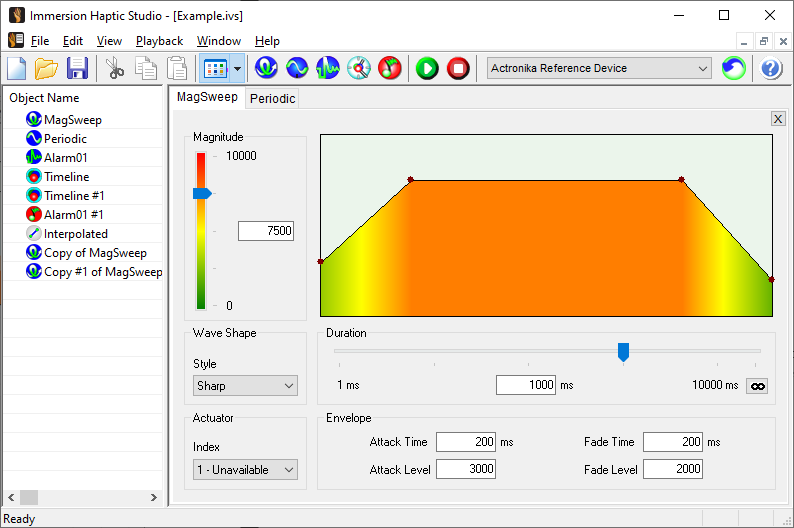
attack-time="200"

attack-level="3000"

fade-time="200"

fade-level="2000"

actuator="1"/>



### Periodic Effects

Periodic effects have the same envelope, style, and actuator ID parameters as magsweep effects, plus a period parameter and a wave type parameter. The period and wave type parameters define a recurring vibration pattern that may be enveloped.

The period parameter is the duration of one cycle of the periodic pattern and can be in milliseconds or microseconds.

The wave type parameter can have the following values:

* Square
* Triangle
* Sine
* Sawtooth up
* Sawtooth down

Example:

<basis-effect name="Periodic"

type="periodic"

duration="100"

magnitude="7977"

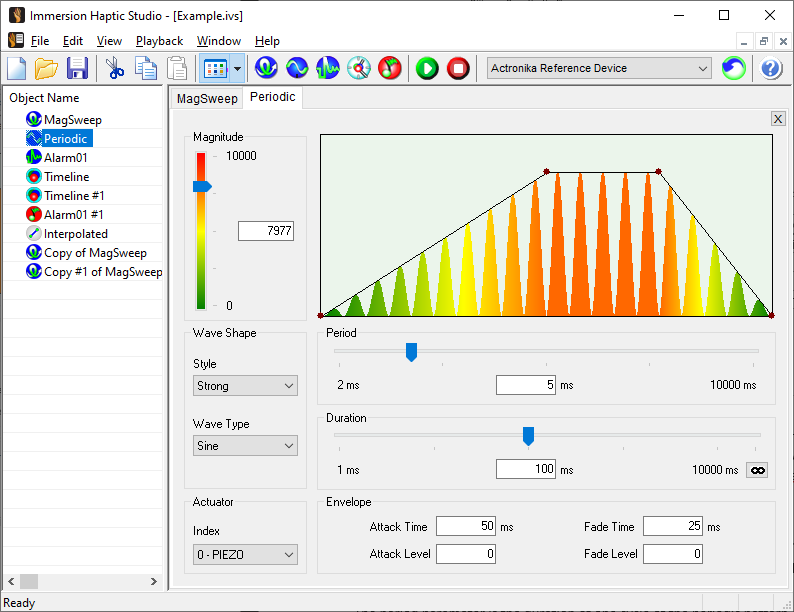
waveform="sine"

period="5"

style="strong"

attack-time="50"

fade-time="25"/>

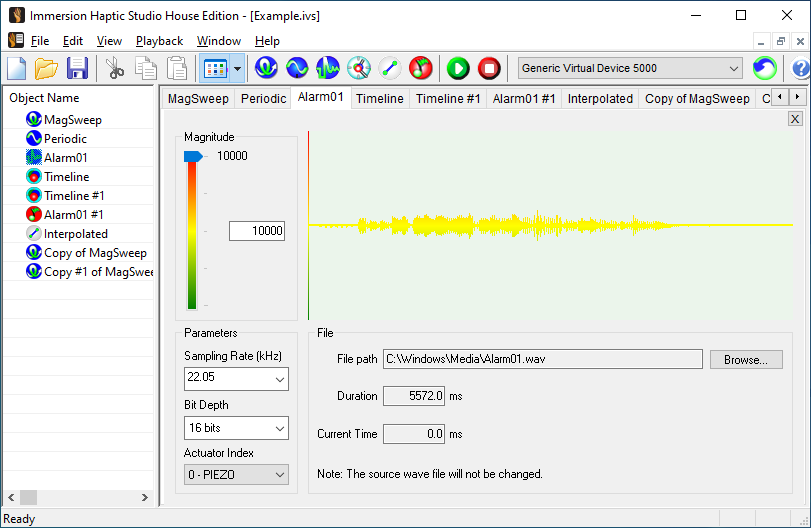


## Waveform Effects

Waveform effects represent the vibration amplitude using a PCM signal stored in a WAVE file. On HD systems, the waveform effect specifies the actuator signal rather than the vibration amplitude.

Example:

<waveform-effect name="Alarm01" path="..\..\..\Windows\Media\Alarm01.wav" magnitude="10000" actuator="0" bit-depth="16" sampling-rate="22050"/>



## Media Effects

Media effects refer to audio files that are to be played together with the vibrotactile effects in a timeline effect.

Example:

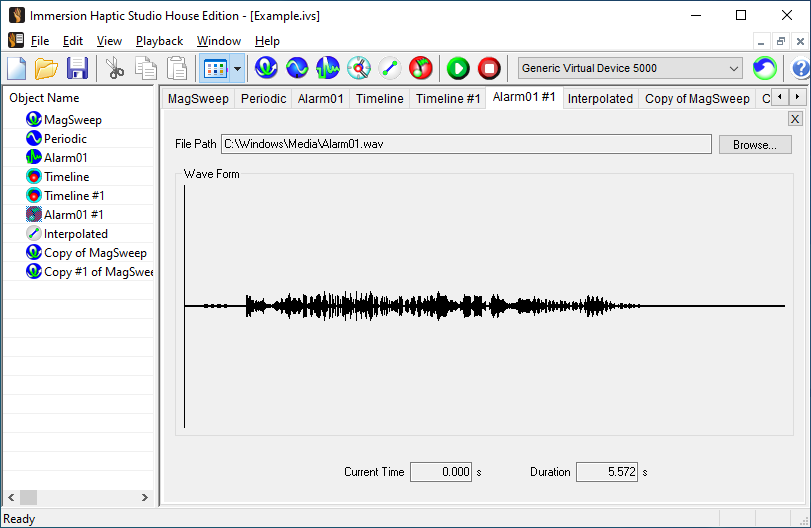
<media-effect name="Alarm01 #1"

type="wav"

path="..\..\..\Windows\Media\Alarm01.wav"

start-offset="0"

duration="5572"/>



## Timeline Effects

Timeline effects combine basis and waveform effects into more complex patterns. Instance of basis or waveform effects are represented as *launch event* in a timeline effect.

Timeline effects may contain repeat events that allow a portion of the timeline effect to be repeated several times, or indefinitely. Repeat events may be nested.

Example 1:

<timeline-effect name="Timeline">

<mute effect="Alarm01"/>

<launch-event time="500"

effect="MagSweep"

magnitude-override="10000"/>

<launch-event time="0"

effect="Alarm01"/>

<launch-event time="1500"

effect="MagSweep"

duration-override="750"/>

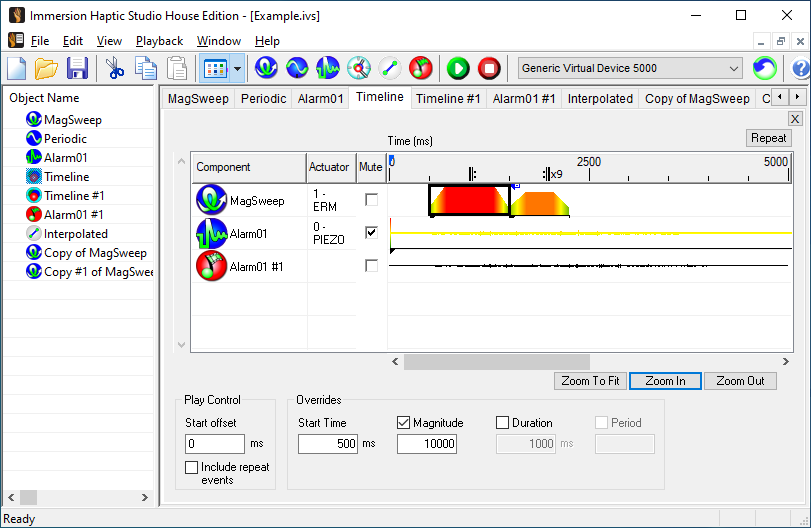
<launch-event time="0"

effect="Alarm01 #1"/>

<repeat-event time="1000"

count="9"

duration="1000"/>



Example 2:

</timeline-effect>

<timeline-effect name="Timeline #1">

<launch-event time="0"

effect="Periodic"/>

<launch-event time="1500"

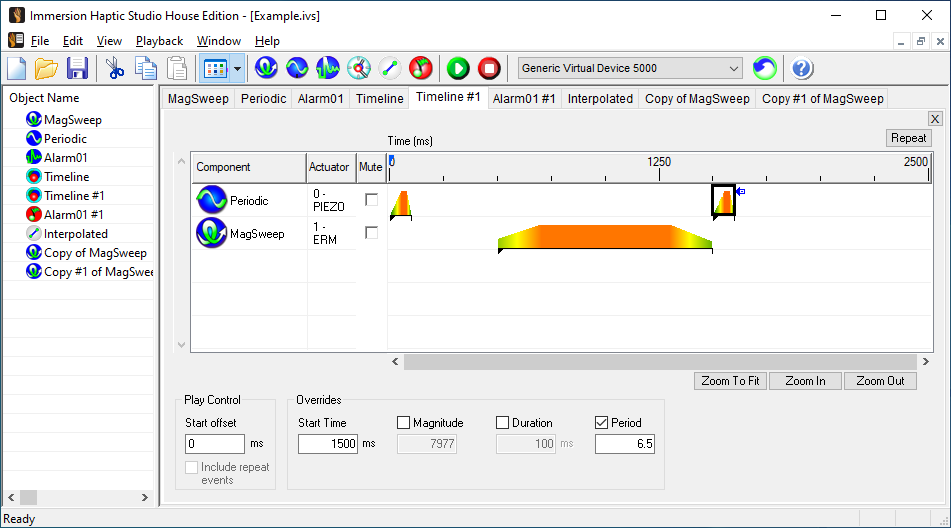
effect="Periodic"

period-override="-2147477148"/>

<launch-event time="500"

effect="MagSweep"/>

</timeline-effect>



## Interpolated Effects

Interpolated effects contain keyframes of basis effects of the same type. When an application plays an interpolated effect is played, the application provides an interpolant value that is used to interpolate between keyframes; that is, between parameters of the basis effects associated with the keyframes. The effect resulting from the interpolation may play repeatedly. The application may change the interpolant value while the interpolated effect plays, resulting in a dynamic effect that may depend on an external stimulus.

Example:

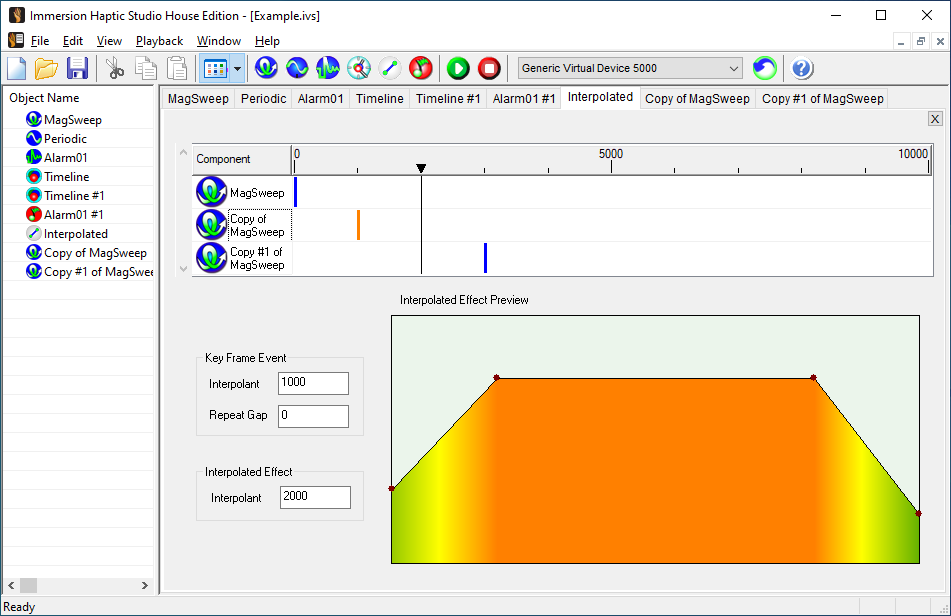
<interpolated-effect>

<key-frame effect="MagSweep" repeat-gap="0" interpolant="0"/>

<key-frame effect="Copy of MagSweep" repeat-gap="0" interpolant="1000"/>

<key-frame effect="Copy #1 of MagSweep" repeat-gap="0" interpolant="3000"/>

</interpolated-effect>



# References

1. WG02 N0070, Call for Proposals on the Coded Representation of Haptics – Phase 1
2. Apple Haptic and Audio Pattern (AHAP) file format: <https://developer.apple.com/documentation/corehaptics/representing_haptic_patterns_in_ahap_files>