WebP container specification - Working Draft (V0.1 Date 09/26)

WebP is a still image format that uses the VP8 key frame encoding (and, possibly, other codecs in the future) to compress image data in a lossy way. The VP8 encoding should make it more efficient than currently used formats. It is optimized for fast image transfer over the network (e.g., for WWW sites). However, it also aims for feature parity (like Color profile, XMP Metadata, Animation etc) with other formats. This document describes the structure of such a file.

The first version of WebP handled only the basic use-case - a file having a single image (being one VP8 key frame) with no metadata. However, the use of a RIFF container allowed to extend it. This document extends it by additionally introducing support for:

- **Metadata and color profiles.** We specify chunks that can contain this information, like can other popular formats.
- **Tiling.** A single VP8 frame has an inherent limitation for width or height of 2^14 pixels and a 512kB limit on the size of first compressed partition. To support larger images, we support images that are composed of multiple tiles, each encoded as a separate VP8 frame. All tiles form logically a single image - they have common metadata, color profile etc. Tiling may also improve efficiency for larger images - grass can be encoded differently than sky.
- **Animation.** An image may have pauses between frames, making it an animation.

Files not using these new features are backward compatible with the original format. Using these features will produce files that are not compatible with older programs.

Terminology

A WebP file contains either a still image (i.e. an encoded matrix of pixels) or an animation (see below) with, possibly, a color profile, metadata etc. In case we need to refer only to the matrix of pixels, we

will call is the *canvas* of the image.

The canvas of an image is built from one or multiple tiles. Each tile is a separately encoded VP8 key frame (other codec are possible in the future). Building an image from several tiles allows to overcome the size limitations of a single VP8 frame. Tiling is supposed to be an internal detail of the file - they are not supposed to be exposed to the user.

Basics

This section introduces basic terms used throughout the document.

Code reading WebP files will be referred to as *readers*, while code writing them will be referred as *writers*.

A 16-bit, little-endian, unsigned integer will be denoted as **uint16**.
A 32-bit, little-endian, unsigned integer will be denoted as **uint32**.

The basic element of a RIFF file is a **chunk**. It consist of:

- 4 ASCII characters that will be called the *chunk tag*.
- **uint32** with the size of the chunk content (that will be denoted as *ckSize*).
- *ckSize* bytes of content.
- If *ckSize* is odd, a single padding byte that SHOULD be 0.

A chunk with a tag "`ABCD`" will be also called a *chunk of type "`ABCD`"*. Note that, in this specification, all chunk tag characters are in file order, not in byte order of an uint32 of any particular architecture.

Note that the padding MUST be also added to the last chunk of the file.

A **list of chunks** is a concatenation of multiple chunks. We will call the first chunk as having *position* 0, the second as *position* 1 etc. By *chunk with index 0 among "ABCD"* we will mean the first chunk among the chunks of type "`ABCD`" in the list, the *chunk with index 1 among "ABCD"* is the second such chunk, etc.

A WebP file MUST begin with a single chunk with a tag "`RIFF`". All other defined chunks are within this chunk. It SHOULD NOT contain anything after it.

The maximum size of RIFF's *ckSize* is 2^32 – 10 bytes (i.e. the size of the whole file is at most 4GiB – 2 bytes).

Note: some RIFF libraries are said to have bugs when handling files larger than 1GiB or 2GiB. If you are using an existing library, check that it handles large files correctly.

The first four bytes of the RIFF chunk contents (i.e. bytes 8-11 of the file) MUST be the ASCII string "`WEBP`". They are followed by a list of chunks. Note that as the size of any chunk is even, the size of the RIFF chunk is also even.

The content of the chunks in that list will be described in the following sections.

Note: RIFF has a convention that all-uppercase chunks are standard chunks that apply to any RIFF file format, while chunks specific for a file format are all-lowercase. WebP doesn't follow this

convention.

Single-image WebP files

First, we will describe a subset of WebP files – files containing only one image (later, we will use it to define multi-image files - file having several different images).

Chunks layout

This section describes what chunks and in what order may appear in a single-image WebP file. The content of these chunks will be described in subsequent sections.

The first chunk inside the RIFF chunk MUST be with a tag of "`VP8 `" (note the space as the last character) or "`VP8X`". Other tags for the first chunk MAY be introduced by future specifications if we add new codecs. This tag of the first chunk determines which of the two possible layouts is used.

Rationale: we fix the possible tags of the first chunk so that it is possible to introduce other codecs, to keep the "`WEBP`" signature at the beginning of RIFF chunk, while still being able to check the codec used by the image by inspecting the byte stream at a fixed position.

The two possible layouts will be called *images without special layout* and *images with special layout*.

Images without special layout

If the first subchunk of RIFF has the tag "`VP8 `", the file contains an *image without special layout*.

This layout SHOULD be used if the image doesn't require advanced features: color profiles, XMP metadata, animation or tiling. Files with this layout are smaller and supported by older software.

Such images consist of:

- A "VP8 " chunk with the bitstream of the single tile.

.

Example: An example layout of such a file looks as follows:

```
RIFF/WEBP
+- VP8 (bitstream of the single tile of the image)
```

Images with special layout

If the first subchunk of RIFF has the tag "`VP8X`" (other tags may be introduced by future specifications, if new codecs are added), the file contains an *image with special layout*.

Note: older readers are not supporting images with special layout and will fail for images having them.

Such an image consists:

- A "`VP8X`" chunk with information about features used in this file.
- An optional "`ICCP`" chunk with color profile.
- An optional "`LOOP`" chunk with animation control data.

- Data for all the frames.
- An optional "`META`" chunk with XMP metadata.
- Some other chunks may be defined by future specifications and placed anywhere in the file.

As will be described in the "`VP8X`" chunk description, by checking a flag one can distinguish animated and non-animated images. A non-animated image has exactly one frame. An animated one may have multiple frames. Data for each frame consists of:

- An optional "`FRM `" (note the space as the last character) chunk with animation frame metadata. It MUST be present in animated images at the beginning of data for that frame. It MUST NOT be present in non-animated images.
- An optional "`TILE`" chunk with tile position metadata. It MUST be present at the beginning of data of image that's represented as multiple tile images.
- A "`VP8 `" chunk with the bitstream of the tile.

All chunks MUST be placed in the same order as listed above (except for unknown chunks, that MAY appear anywhere). If a chunk appears in a wrong place, the file is invalid, but readers MAY parse the file ignoring the chunks that come too late.

Rationale: setting the order of chunks should allow to quickly stop the search for e.g., the ICCP if it is not present in the file. The rule of ignoring late chunks should make programs that needs to do a full search give the same results as the ones stopping early.

Example: An example layout of a non-animated, tiled image may look as follows:

```
RIFF/WEBP
+- VP8X (descriptions of features used)
+- ICCP (color profile)
+- TILE (First tile parameters)
+- VP8 (bitstream - first tile)
+- TILE (Second tile parameters)
+- VP8 (bitstream - second tile)
+- TILE (third tile parameters)
+- VP8 (bitstream - third tile)
+- TILE (fourth tile parameters)
+- VP8 (bitstream - fourth tile)
+- META (XMP metadata)
```

Example: An example layout of an animated image may look as follows:

```
RIFF/WEBP
+- VP8X (descriptions of features used)
+- LOOP (animation control parameters)
+- FRM (first animation frame parameters)
+- VP8 (bitstream - first image frame)
+- FRM (second animation frame parameters)
+- VP8 (bitstream - second image frame)
+- META(XMP metadata)
```

Assembling the canvas from tiles and animation

Contents of the chunks will be described in details in subsequent section. Here, we provide an overview how they are used to assemble the canvas. The notation *VP8X.canvasWidth* means the field in the "`VP8X`" described as *canvasWidth*.

Decoding a non-animated canvas MUST be equivalent to the following pseudo-code:

- **assert not** *VP8X.flags.haveAnimation*
- *canvas* ← new black image of size *VP8X.canvasWidth* x *VP8X.canvasHeight*.
- *tile_params.tileCanvasX* = *tile_params.tileCanvasY* = 0
- **for** *chunk* **in** *data_for_all_frames*:
    - **if** *chunk.tag* == "TILE":
        - **assert** No other TILE chunk after the last "`VP8 `" chunk
        - *tile_params* = *chunk*
    - **if** *chunk.tag* == "VP8 ":
        - render image in *chunk* in *canvas* with top-left corner in (*tile_params.tileCanvasX*, *tile_params.tileCanvasY*) using the isometry in *VP8X.flags.rotationAndSymmetry*.
        - *tile_params.tileCanvasX* = *tile_params.tileCanvasY* = 0
    - Ignore unknown chunks
- *canvas* contains the decoded canvas.

Decoding an animated canvas MUST be equivalent to the following pseudo-code:

- **assert** *VP8X.flags.haveAnimation*
- *canvas* ← new black image of size *VP8X.canvasWidth* x *VP8X.canvasHeight*.
- **if** *LOOP.loopCount*==0:
    - *LOOP.loopCount*=∞
- *current_FRM* ← **nil**
- **for** *LOOP.loop* = 0, …, *LOOP.loopCount*-1
    - **assert** First chunk in *data_for_all_frames* is a FRM
    - **for** *chunk* **in** *data_for_all_frames*:
        - **if** *chunk.tag* == "FRM ":
            - **if** *current_FRM* != **nil**:
                - Show the contents of canvas for *current_FRM.frameDuration**10ms.
            - *current_FRM* = *chunk*
        - **if** *chunk.tag* == "VP8 ":
            - **assert** *tile_params.tileCanvasX* >= *current_FRM.frameX*
            - **assert** *tile_params.tileCanvasY* >= *current_FRM.frameY*
            - **assert** *tile_params.tileCanvasX* + *chunk.tileWidth* >= *current_FRM.frameX* + *current_FRM.frameWidth*
            - **assert** *tile_params.tileCanvasY* + *chunk.tileHeight* >= *current_FRM.frameX* + *current_FRM.frameHeight*
            - render image in *chunk* in *canvas* with top-left corner in (*tile_params.tileCanvasX*, *tile_params.tileCanvasY*) using the isometry in *VP8X.flags.rotationAndSymmetry*.
            - *tile_params.tileCanvasX* = *tile_params.tileCanvasY* = 0
        - Ignore unknown chunks

- *canvas* contains the decoded canvas.

As described earlier, if an assert related to chunk ordering fails, the reader MAY ignore the badly-ordered chunks instead of failing to decode the file.

Bitstream chunk(s) (VP8)

These chunks contain compressed image data. Currently, the only allowed bitstream is VP8 and uses "`VP8 `" (note the space as the last character) as its tag. We will refer to all chunks with this tag as *bitstream chunks*. As described earlier, images without special layout have a single bitstream chunk as the first subchunk of RIFF, while images with special layout may contain several of them - one for each tile.

The content of a "`VP8 `" chunk MUST be one VP8 key frame (with optional padding – see below). The current draft of a VP8 specification can be found at http://tools.ietf.org/html/draft-bankoski-vp8-bitstream-04. Note that the VP8 frame header contains the VP8 frame width and height. It is assumed to be the width and height of the tile.

The VP8 specification specifies how to decode the image into Y'CbCr format. To convert to RGB, Rec. 601 SHOULD be used.

For compatibility with older readers, if the size of the frame is odd, writers SHOULD append a padding byte (preferably 0) inside the chunk contents, making the chunk's *ckSize* even. Newer readers MUST support odd-sized tile chunks.

VP8X chunk (special layout)

As described earlier, a chunk with tag "VP8X", is the first chunk of images with special layout. It is used to enable advanced features of WebP.

The content of the chunk is as follows:

- **uint32** *flags*. The following bits are currently used (with 0 being the least significant bit):
    - bit 0: *haveTile*: set if the image is represented by Tiles.
    - bit 1: *haveAnimation*: set if the file is an animation. Data in "`LOOP`" and "`FRM `" chunks should be used to control the animation.
    - bit 2: *haveIccp*: set if the file contains a "`ICCP`" chunk with a color profile. If a file contains an "`ICCP`" chunk but this bit is not set, the error is flagged while constructing the Mux-Container.
    - bit 3: *haveMetadat*: set if the file contains a "META" chunk with a XMP metadata. If a file contains an "`META`" chunk but this bit is not set, the error is flagged while constructing the Mux-Container.

  Future specification MAY define other bits in *flags*. Bits not defined by this specification MUST be preserved when modifying the file.

- **uint32** *canvasWidth*: width of the canvas in pixels (after the optional rotation or symmetry - see below).
- **uint32** *canvasHeight*: height of the canvas in pixels (after the optional rotation or symmetry - see below).

Future specifications MAY add more fields. If a chunk of larger size is found, programs MUST ignore the extra bytes but MUST preserve them when modifying the file.

LOOP chunk (global animation parameters)

For images that are animations, this chunk contains the global parameters of the animation.

This chunks MUST appear if the *haveAnimation* flag in chunk VP8X is set. If the *haveAnimation* flag is not set and this chunk is present, it MUST be ignored.

The content of the chunk is as follows:

- **uint16** *loopCount* For animations, the number of times to loop this animation. 0 means infinite.

Future specifications MAY add more fields. If a chunk of larger size is found, programs MUST ignore the extra bytes but MUST preserve them when modifying the file.

FRM chunk (per-frame animation parameters)

For images that are animations, these chunks contain the per-frame parameters of the animation.

The content of the chunk is as follows:

- **uint32** *frameX*: *x* coordinate of the upper left corner of the frame. For images using the VP8 codec, it MUST be divisible by 32. Other codecs MAY specify other constraints. Described in more details later.
- **uint32** *frameY*: *y* coordinate of the upper left corner of the frame. For images using the VP8 codec, it MUST be divisible by 32. Other codecs MAY specify other constraints. Described in more details later.
- **uint32** *frameWidth*: width of the frame. For images using the VP8 codec, it MUST be divisible by 16 or such that *frameX*+*frameWidth*==*canvasWidth*. Other codecs MAY specify other constraints. Desribed in more details later.
- **uint32** *frameHeight*: height. For images using the VP8 codec, it MUST be divisible by 16 or such that *frameY*+*frameHeight*==*canvasHeight*. Other codecs MAY specify other constraints. Desribed in more details later.
- **uint16** *frameDuration* Time to wait before displaying the next tile, in 1ms unit.

.

Rationale: the requirement for corner coordinates to be divisible by 32 means that pixels on U and V planes are aligned to 16 byte boundary (even after a rotation), what may help with vector instructions on some architectures. Also, this makes the tiles also aligned to 16-pixel macroblock boundaries.

Rationale: the requirement for the width and height to be divisible by 16 or touching the edge of the canvas simplifies the handling of macroblocks that are on the edge of a tile - VP8 decoders can overwrite pixels outside the boundary in such a macroblock and this guarantees they won't overwrite any data.

Future specifications MAY add more fields. If a chunk of larger size is found, programs MUST ignore the extra bytes but MUST preserve them when modifying the file.

TILE chunks (tile parameters)

This chunk contains information about a single tile and describes the bitstream chunk that proceeds it.

The content of such a chunk is as follows:

- **uint32** *tileCanvasX*: x coordinate of the upper left corner of the tile. For VP8 tiles, it MUST be divisible by 32. Other codecs MAY specify other constraints.
- **uint32** *tileCanvasY*: y coordinate of the upper left corner of the tile. For VP8 tiles, it MUST be divisible by 32. Other codecs MAY specify other constraints.

Future specifications MAY add more fields. If a chunk of larger size is found, programs MUST ignore the extra bytes but MUST preserve them when modifying the file.

As described earlier, the TILE chunk is followed by a VP8 data. From that chun, we can read the height and width of the tile, that we will denote by *tileWidth* and *tileHeight*. In the case of VP8, we have the following constraints:

- The width of a tile MUST be divisible by 16 or there MUST be *tileCanvasX*+*tileWidth* == *canvasWidth*.
- The height of a tile MUST be divisible by 16 or there MUST be *tileCanvasY*+*tileHeight* == *canvasHeight*.

ICCP chunk (color profile)

An optional "`ICCP`" chunk contains an ICC profile. There SHOULD be at most one such chunk. The first byte of the chunk is the compression type. Two values are currently defined: a value of 0 means no compression, while a value of 1 means deflate/inflate compression. It is followed by a compressed or non-compressed ICC profile - see www.color.org for specifications.

The color profile can be a v2 or v4 profile. If this chunk is missing, sRGB SHOULD be assumed.

META chunk (compressed XMP metadata)

Such a chunk (if present) contains XMP metadata. There SHOULD be at most one such chunk. If there are more such chunks, readers SHOULD ignore all except the first one. The first byte specifies compression type. Two values are currently defined: a value of 0 means no compression, while a value of 1 means deflate/inflate compression. It is followed by a compressed or non-compressed XMP metadata packet.

XMP packets are XML text specified in http://www.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/XMPSpecificationPart1.pdf. The chunk tag is different from the one specified by Adobe for WAV and AVI (also RIFF formats) because we have the options of compression.

Additional guidance about handling metadata can be found at: http://www.metadataworkinggroup.org/pdf/mwg_guidance.pdf . Note that the sections of the document about reconciliation of EXIF, XMP and IPTC-IIM don't apply to WebP, as WebP supports only XMP, thus no reconciliation is necessary.

Other chunks

A file MAY contain other chunks, defined in some future specification. Such chunks MUST be ignored, but preserved. Writers SHOULD try to preserve them in the original order.