1 Introduction

This document specifies the format for the dotLottie file. The dotLottie file is required to have the extension ".lottie", but can be otherwise identified by the IDENT identifier (see 3.3) in its header.

2 Notation

2.1 BNF

The dotLottie file format specification uses basic BNF to describe its syntax. Additional augmentation used in this syntax are described in 2.2.

2.2 Basic rules

The custom parts used throughout this document are:

OCTET: Any sequence of 8 consecutive bits.
CHAR: 8-bit sequence with numeric decimal values between 0 through 127.

3 Structure

The dotLottie file is designed to be byte-streamed to improve its portability. In tune with this, the dotLottie's data ordering is big endian.

All <offset> data (see 4.1) depicted are relative to the first byte of the <version> part (see 3.3).

3.1 dotLottie file structure

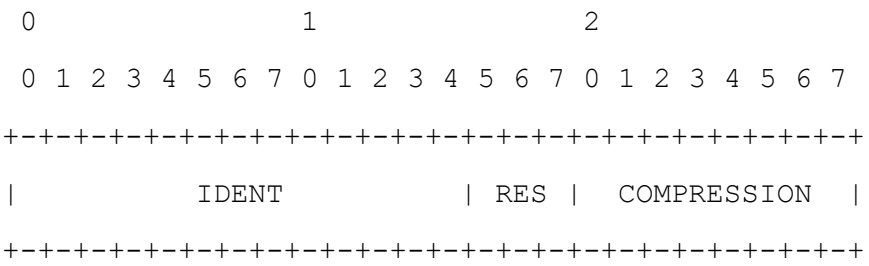
<dotlottie> ::= <header><datapart>

The uncompressed dotLottie content comprises of a <header> (see 3.2) and a <datapart> (see 3.9).

3.2 File header

<header> ::= <version><overallhead><manifest><partdesc>

3.3 <version>



IDENT: These are 13 bits set to equal the decimal value 7311. “7311” are the first four bytes yielded when the word “dotlottie” (all lowercase) is hashed using SHA256.

RES: These are reserved bits that could be used at a future date.

MAJOR, MINOR, PATCH: 3 bytes that depict the dotLottie version.

The COMPRESSION byte determines the type of compression that the file is using. This code determines the type of compression and method of compression. The method of compression refers to whether the entire <datapart> is compressed as a whole or whether <payload> parts are compressed separately.

COMPRESSION: **001** – No data compression used
002 – Brotli compress on entire <datapart>

If the value of COMPRESSION is set to 002, then the entire <datapart> should be uncompressed before the offset/length values in the <datadesc> (see 3.7) parts are used to extract the respective <datapart>.

3.4 Overall header

<overallhead> ::= <manifestloc><partloc>

The <manifestloc> part can be used to determine the location of the manifest information (see 3.5).

The <partloc> part can be used to determine the location of the part descriptors <partdesc> (see 3.7).

3.5 Location of manifest

<manifestloc> ::= <offset><length>

For the parts <offset> and <length> above, see 4.1 and 4.2, respectively.

3.6 Location of part descriptors

<partloc> ::= <offset><length>

For the parts <offset> and <length> above, see 4.1 and 4.2, respectively.

3.7 Data descriptors

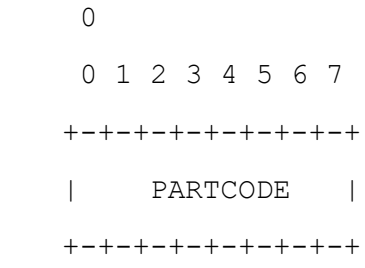
<partdesc> ::= <parttype><offset><length>

|<parttype><offset><length><partdesc>

The <partdesc> describes the offset and length of each collection of <datapart> parts that are grouped by PARTCODE in <parttype> (see 3.8). By this implication, all <datapart> parts classified under each PARTCODE must be consecutively grouped together.

For the parts <offset> and <length> above, see 4.1 and 4.2, respectively.

3.8 <parttype>



The PARTCODE is used to identify the type of the part being located at the <offset> in <partdesc>. The valid codes for PARTCODE are numbers 0 through 255, with the reserved codes as follows:

PARTCODE: **1** – "lottie" data
2 – "media" data
3 – "script" data

3.9 Data parts

<datapart> ::= <dataheader><payload>
|<dataheader><payload><datapart>

The <datapart> carries the payload for any given data element. These <payload> parts are grouped by their types as defined by PARTCODE in 3.8. For instance, the <payload> can contain either a whole lottie JSON sequence, a bitmap file, or a script.

3.10 The data header

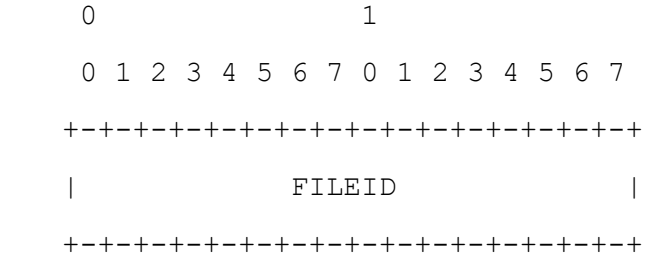
<dataheader> ::= <mediatype><offset><length><id>

Each <dataheader> part will have a <mediatype> part, which is just an 8-byte sequence (see 3.12). It is important to note that the <mediatype> part is only relevant for the appropriate data types.

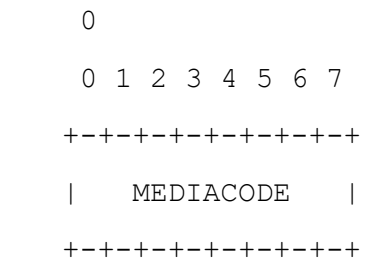
The <id> part is a unique file ID created by the dotLottie generator. The <id> part is used in the "files" array of the root object of the <manifest> part. Therefore, the "files" key reserved for use by the dotLottie generator within the root object of the manifest JSON data.

For parts <offset> and <length>, see 4.1 and 4.2, respectively.

3.11 <id>



3.12 <mediatype>



The MEDIACODE is a numeric value between 0 through 255 that describes the type of media in the given <mediapart>. The values between 0 and 99 will be used as code for unencoded (raw) media and values 100 through 199 for Base64-encoded media.

The current reserved values for these parts are:

MEDIACODE:
- **001** – PNG
- **002** – JPEG
- **003** – GIF
- **004** – BMP
- **005** – TIFF
- **101** – Base64-encoded PNG
- **102** – Base64-encoded JPEG
- **103** – Base64-encoded GIF

104 – Base64-encoded BMP
105 – Base64-encoded TIFF

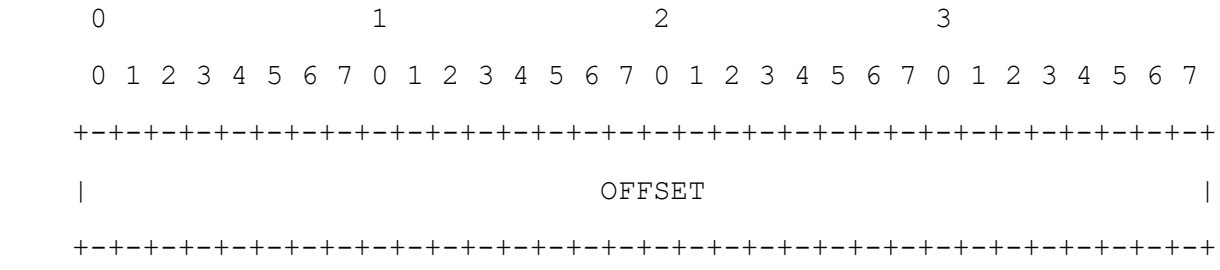
3.13 The payload

<payload> ::= OCTET|OCTET<payload>

The content of <payload> is interpreted based on the PARTCODE in the <parttype> part (see 3.8). This interpretation will be handled by the application using the dotLottie file.

4 Common parts

4.1 <offset>



4.2 <length>