jp2StructCheck - simple JP2 file structure checker

Johan van der Knijff, KB/National Library of the Netherlands

Version 31 August 2011

Contents

Overview

This document describes a simple Python-based tool that checks the general file structure of a JP2 (or JPEG 2000 Part 1) image. In particular, it verifies the following aspects of a JP2 file:

1. The presence of all required JP2 'boxes'.

2. The presence of an end-of-codestream marker at the end of the image codestream.

To be clear, the tool is *not* a full-fledged validator; rather, it performs some very simple checks that are currently not covered well by other existing tools.

The tool comes in two flavours:

1. a Python script (which will work on any operating system for which a Python interpreter is available)

2. A Windows executable (byte-code compiled using the py2Exe extension[1])

Background and scope

To understand what the tool does, it is helpful to have some idea of the general file structure of a JP2 file. The following key points are important here:

1. A JP2 file is made up of components that are called *boxes* (this is illustrated by Figure 1 below).

2. Some of these boxes are compulsory, whereas others are optional.

3. Some boxes are *superboxes*, which are boxes that contain other (child) boxes.

4. The actual image code stream is stored in a *contiguous codestream* box.

5. A well-formed code stream is always terminated by an *end of codestream marker*.

[1] Link: http://www.py2exe.org/

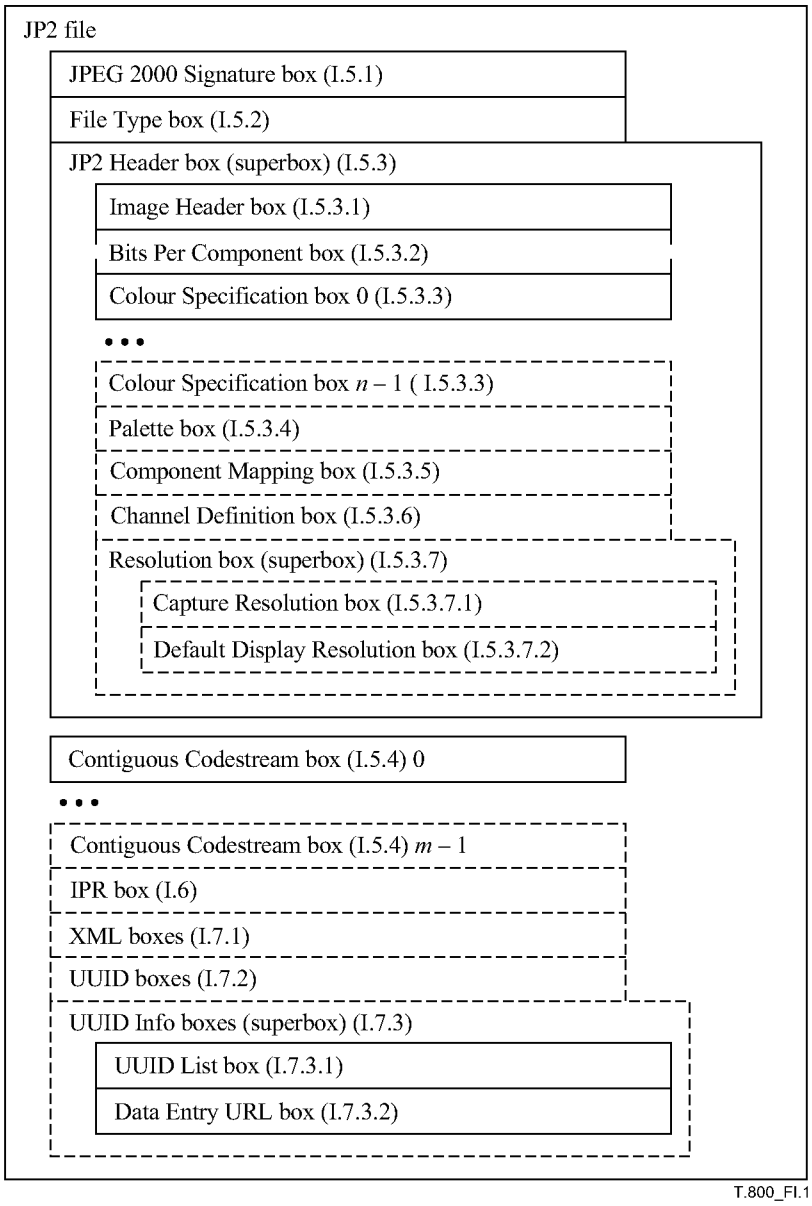

Figure 1 Conceptual structure of a JP2 file (from ISO/IEC 15444-1)

What jp2StructCheck does

When jp2StructCheck analyses a file, it first parses the top-level box structure, and collects the unique identifiers (or marker codes) of all boxes. If it encounters a contiguous codestream box, it checks if the code stream is terminated by a valid end-of-codestream marker. Finally, it checks if the file contains all the compulsory/required top-level boxes. These are:

1. JPEG 2000 signature box
2. File Type box
3. JP2 Header box
4. Contiguous Codestream box

Limitations

It is important to note here that *jp2StructCheck* only checks the top-level boxes. In case of a superbox, it does not recurse into its child boxes. For example, it does not check if a JP2 Header box (which is a superbox) contains a Bits Per Component Box (which is required by the standard). So the scope of the tool is limited to a rather superficial check of the general file structure. It is *not* a JP2 validator! The main scope is to be able to detect certain types of file corruption that may occur as a result of hardware failure (e.g. network interruptions) during the creation of an image.

In addition, the fact that a code stream is terminated by and end-of-codestream marker is no guarantee that the code stream is complete. For instance, if due to some hardware failure some part of the middle of the codestream is not written, *jp2StructCheck* will not detect this (detecting such errors would probably require decoding the entire code stream).

Dependencies

Python version

The Python version of the JP2 file structure checker requires a Python interpreter. It is important to note here that *jp2StructCheck* only works with either Python 2.7 or Python 3.2 (or more recent). It will *not* work with Python 3.0 or 3.1![2] You can download Python from the following locations:

http://www.python.org/download/releases/2.7.2/

Or:

http://www.python.org/download/releases/3.2.1/

Windows executable version

The Windows executable version doesn't require a Python interpreter. However, it does need the following Windows libraries (which are all part of most Windows-based systems, and are located in the 'WINDOWS\system32' directory):

WS2_32.dll
SHELL32.dll
USER32.dll
ADVAPI32.dll
KERNEL32.dll

[2] The reason for this is that the command-line interface uses the 'argparse' library, which was introduced in Python 2.7, subsequently dropped from Python 3.0 and 3.1, and then re-introduced in Python 3.2.

Installation

Python version

Unzip the contents of 'jp2StructCheckDDMMYYYY.zip' to an empty folder on your PC. If you are working on a Linux/Unix based system you will need to make the script executable, and convert the line breaks to Unix-style ones. To do this, use the following commands:

```
chmod 755 jp2StructCheck.py

dos2unix jp2StructCheck.py
```

Windows executable version

Unzip the contents of 'jp2StructCheckDDMMYYYYDistWin32.zip' to an empty folder on your PC.

Command-line use

This section describes the command-line interface of *jp2StructCheck* as a command-line tool. For the sake of convenience all command line descriptions and examples assume the use of the Python version of *jp2StructCheck*, with the additional assumption that Python files are associated with the (correct) Python interpreter on your system. If you are using the Windows executable version, substitute all occurrences of 'jp2StructCheck.py' below with 'jp2StructCheck' (using the full path to the executable, if needed).

Synopsis

jp2StructCheck uses the following general syntax:

```
jp2StructCheck.py jp2In
```

Where *jp2In* is the path to one or more JP2 image files. It is possible to analyse multiple files using wildcards. The results of the check are written to the standard output device.

In addition to this the following optional command-line flags are available:

-t : report output in terse, comma delimited format. First item = image name; 2nd item = found all required boxes flag (True/False); 3rd item = found end of codestream marker flag (True/False).

-h : show help message and exit

-v : show version info and exit

Examples

Example 1: intact image:

```
jp2StructCheck.py balloon.jp2
```

Result:

```
File name: "balloon.jp2"
  Found all required boxes: True
  Found end of codestream marker: True
```

Example 2: image with missing (renamed) JP2 header box:

```
jp2StructCheck.py balloon_nojp2h.jp2
```

Result:

```
File name: "balloon_nojp2h.jp2"
  Found all required boxes: False
  Found end of codestream marker: True
  Missing boxes:
   jp2h
```

Example 3: image with incomplete code stream:

```
jp2StructCheck.py balloon _trunc_3224.jp2.jp2
```

Result:

```
File name: "balloon_trunc_3224.jp2"
  Found all required boxes: True
  Found end of codestream marker: False
```

Example 4: check all .jp2 images in a folder:

```
jp2StructCheck.py *.jp2
```

Result:

```
File name: "balloon.jp2"
  Found all required boxes: True
  Found end of codestream marker: True
File name: "balloon_nojp2c.jp2"
  Found all required boxes: False
  Found end of codestream marker: False
  Missing boxes:
   jp2c
File name: "balloon_nojp2h.jp2"
  Found all required boxes: False
  Found end of codestream marker: True
  Missing boxes:
   jp2h
File name: "balloon_trunc_3224.jp2"
  Found all required boxes: True
  Found end of codestream marker: False
File name: "balloon_trunc_756119.jp2"
  Found all required boxes: True
  Found end of codestream marker: False
```

For analyzing large numbers of images it may be more convenient to use the terse output format, and redirect the output to a text file:

```
jp2StructCheck.py *.jp2 –t > results.csv
```

This results in a comma-delimited file that can be easily imported in, for example, a spreadsheet application:

```
"balloon.jp2",True,True
"balloon_nojp2c.jp2",False,False
"balloon_nojp2h.jp2",False,True
"balloon_trunc_3224.jp2",True,False
"balloon_trunc_756119.jp2",True,False
```

Here, the first item is the image name (or path), followed by the 'found all required boxes flag' and the 'found end of codestream marker' flag (both of which are either True or False).

Issues

Wildcards are misbehaving under Unix

Some preliminary tests revealed that using wildcards under Linux results in unexpected behaviour. For instance:

```
jp2StructCheck.py *.jp2
```

results in only the first encountered file that matches the pattern being checked. A simple workaround is to use quotation marks on the command line:

```
jp2StructCheck.py "*.jp2"
```

Test data

The software is accompanied by a small set of test images that illustrate its functionality. The following table gives a description of each of these images.

File	Description
balloon.jp2	JP2 image with all required boxes and intact code stream
balloon_trunc_756119.jp2	File truncated at 756119 bytes (incomplete codestream)
balloon_trunc_3224.jp2	File truncated at 3224 bytes (incomplete codestream)
balloon_nojp2c.jp2	Marker of contiguous codestream box changed to 'jp2d' (instead of 'jp2c')
balloon_nojp2h.jp2	Marker of header box changed to 'jp2i' (instead of 'jp2h')

References

ISO/IEC. “Information technology — JPEG 2000 image coding system: Core coding system”. ISO/IEC 15444-1, Second edition. Geneva: ISO/IEC, 2004a. 28 Dec 2010 <http://www.jpeg.org/public/15444-1annexi.pdf> (“Annex I: JP2 file format syntax” only).