*These are the bare necessary instructions for dependency installation and testing. For complete code documentation, see the relevant section on Image Tampering Detection in Deliverable D3.2. All instructions concern 64-bit systems. Attempt 32-bit installation at your own peril.*

*The current instructions are based on libjpeg-8. If you want to compile with any other version of libjpeg, you should replace all libjpeg-related .h files placed in folder /ExportDCT/ with the ones from the source of your libjpeg of choice. But leave ExportDCT.h alone.*

*Besides the instructions contained here, make sure you also have a recent version of JNA (at least 4.1) properly installed in your system.*

**Windows dependency installation instructions:**

Precompiled DLL files for both libJPEG and ExportDCT are included in the distribution (**./Executable/**). If you want to try and use them, jump to “**Placing the DIIs**” below. If compilation from source is necessary, then:

**libJPEG8d:**

-Download libjpeg8d from the Independent JPEG group (<http://www.ijg.org/files/>)

(A version is included alongside the deliverable, in folder **Auxiliary\_Sources\jpeg-8d\_distribution**. The \.lib\ folder even includes the compiled .dll -again).

-Use an 64-bit version of MSYS (MSYS64) to compile libjpeg:

a) run ./configure from within the **jpeg-8d\_distribution** folder.

b) edit the generated **jconfig.h** file by replacing the line:

#undef HAVE\_PROTOTYPES

with

#define HAVE\_PROTOTYPES

(this file is replaced during configuration, do the manual change after completing the configuration)

c) ./make and then ./make install

d) enter the **/.libs** subdirectory and take **libjpeg-8.dll**

**ExportDCT:**

-Copy **libjpeg-8.dll** and **jconfig.h** to the **Auxiliary\_Sources\ExportDCT\** subdirectory.

-Go to the **ExportDCT**\ subdirectory and run (in MSYS64):

gcc -shared -o ExportDCT.dll ExportDCT.c libjpeg-8.dll -Wl,--output-def,ExportDCT.def,--out-implib,libExportDCT.a

**Placing the DLLs:**

-In order to run the project from Netbeans, take both the resulting **ExportDCT.dll** and **libjpeg-8.dll** and copy them to **\Executable\** alongside **MultimediaManipulationToolbox.jar**.

For testing that the native C code works independently from Java, a test file is included in the distribution. You can build **TestDCT.c** with:

gcc –o TestDCT.exe TestDCT.c ExportDCT.dll libjpeg-8.dll (in MSYS64)

Then run **TestDCT.exe**, having **068.jpg** in the folder. The output should be:

193

129

5

If yes, then the C code is working correctly and any errors are due to Java/JNA integration.

**Ubuntu installation instructions:**

A precompiled version of ExportDCT can be found in **Executable/**. You can see if it works by placing it somewhere in your LD\_LIBRARY\_PATH and seeing if it works, but it’s probably a better idea to build it yourself.

**libJPEG8d:**

The libjpeg8 (libjpeg8-dev) package is needed for various Ubuntu applications and is most likely already included in your distribution. If not, install using apt-get. Make sure LD\_LIBRARY\_PATH includes libjpeg8. You can always install libjpeg8 from source using the folder provided, if you are confident you know what you are doing.

**ExportDCT:**

**IMPORTANT: the file “jconfig.h” in folder “ExportDCT” is necessary for Windows compilation, but will disrupt Linux compilation. Be sure to remove jconfig.h from the Auxiliary\_Sources/ExportDCT/** **folder if compiling ExportDCT in Linux.**

Build **libExportDCT.so** by entering the **Auxiliary\_Sources/ExportDCT/** directory and running:

gcc -c -Wall -fpic ExportDCT.c

gcc -shared -o libExportDCT.so ExportDCT.o -l:libjpeg.so.8

and copy it somewhere in your LD\_LIBRARY\_PATH. Don't forget to adjust its permissions.

You can test native C functionality independently from trying Java, by compiling the provided TestDCT.c code. First run:

gcc -o TestDCT TestDCT.c -lExportDCT -ljpeg

And then execute ./TestDCT, having **068.jpg** in the folder. The output should be:

193

129

5

If yes, then the C code is working correctly and any errors are due to Java/JNA integration.