JPEG2000 Codec by Aware
with 3-D API Extensions

Version 3.7 for Win32, Win CE, Linux, MacOSX and Unix

Developer's Guide

JPEG 2000 Image Compression and Decompression Library and Associated Demonstration Tools

Aware, Inc.
40 Middlesex Turnpike
Bedford, MA 01730
Telephone: 781-276-4000
Fax: 781-276-4001
E-mail: help@aware.com

Manual generated February 28, 2005.

Aware Technical Support

Free technical support is available via email (`help@aware.com`) or via telephone for 30 days from the date of your first telephone call. Telephone support hours are from Monday through Friday, except holidays, 9:00 am to 5:00 pm eastern standard time.

(781) 276-4000 Voice

(781) 276-4001 Fax

Contents

Chapter 1

Introduction

1.1 Aware, Inc. JPEG2000 Toolkit

The Aware JPEG2000 Toolkit has been designed to:

- perform image compression and decompression using JPEG2000, and
- perform reformating of JPEG2000 images.

The basic library structure is the aw_j2k_object, which is used to store images and related parameters. Image compression, decompression and reformating can all be performed using the following sequence of function calls:

- **aw_j2k_set_input_image(aw_j2k_object, image_data)**, which reads an image into the object, and
- **aw_j2k_get_output_image(aw_j2k_object, image_data)**, which returns a new image from the object.

Compression of an image is performed by setting the input image to be the original image using the input function and seting the output image type to be a JPEG2000 image.

Decompression is performed by setting the input image to be a JPEG2000 image and the output image to be a noncompressed image type such as a BMP.

Reformating one JPEG2000 image to another is performed by setting both the input image and output images to be JPEG2000 images.

A more detailed description of the library architecture and related objects can be found in the Library Overview (page 11).

The latest information regarding added features and bug fixes can be found in the release notes, which are part of the Toolkit. The release notes are in the file `release_notes.txt`, which is found in the `docs` subdirectory of the installation.

1.2 Platforms

The Aware JPEG2000 Codec is available for the various Microsoft Windows platforms (Windows 95/98/2000/NT), Windows CE for PocketPC, as well as several flavors of UNIX and other platforms. In addition, Internet Explorer ActiveX Controls and Netscape Navigator Plug-ins that add JPEG2000 image viewing support to standard browsers, are available for Windows 95/98/2000/NT. A PocketIE ActiveX Control for Windows CE PocketPC devices is also available. Please contact Aware, Inc. for more information.

1.3 Standards Compliance

The Aware, Inc. JPEG2000 Compression/Decompression Library conforms to the JPEG2000 Part I International Standard. This new state of the art ISO/ITU still image compression standard provides a fast and flexible solution for high-quality, wavelet-based compression. JPEG2000 was designed to overcome the limitations of the original JPEG standard and provide high quality images at low bitrates. In addition, JPEG2000 includes new features and functionalities for client/server imaging applications and resource constrained wireless devices. JPEG2000's advantages include:

- High compression efficiency
- Lossless color transforms
- Lossless and lossy coding potential
- Progressive decompression of images by resolution, quality, color channel, and spatial location.
- Regions of interest compressed at different quality levels
- Error resilience in noisy environments

For further information please see `J2k.aware.com`.

1.4 Demonstration Tools Description

The Aware, Inc. JPEG2000 Demonstration Tool (J2K Tool) is designed to illustrate the compression and decompression functionality of the Aware, Inc. JPEG2000 Library. Reformatting of JPEG2000 images is not supported in the J2K Tool at this time. The J2K Tool is a Windows 95/98/2000/NT application that provides access to many of the compression/decompression parameters and abilities within the library. To accomplish this, the application utilizes a simple GUI front end written with Microsoft Foundation Classes (MFC). The J2K Tool can input imagery in

several standard formats (BMP, JPEG, JPEG 2000, TGA, TIFF, PNM, PGX) as well as raw imagery (with or without header bytes). Each input image is displayed graphically to the user along side the compressed/decompressed JPEG2000 version – when created. The user may interactively change the compression parameters as well as modify the decompressed portion of the compressed image (this concept will be discussed later in this manual).

The user may also save the compressed JPEG2000 image and save a BMP version of the decompressed image for later use and comparison.

The primary file for this demo, with respect to the Aware, Inc. JPEG2000 Codec is j2kimage.cpp. The C++ class in this file is responsible for the compression and decompression of input files. This class also holds member variables for all compression and decompression parameters. For information on the activities of the tool interface please become familiar with the j2k_toolview.cpp file.

The j2k_tool.dsw file is the Microsoft Visual C++ workspace file for this project. The other files within this project contribute to various portions of the routine operation of the program and the collection and display of associated parameters and features.

1.5 ANSI C Command Line Tool For Windows and UNIX

A command line program called j2kdriver.c is also included in this distribution. The command line program performs JPEG2000 compression, JPEG2000 decompression and reformatting of JPEG2000 images. The command line program is discussed in more detail in the Command Line Tool Overview (page 217).

1.6 Windows CE Image Viewer Application

A simple image viewer program for Windows CE for PocketPC devices is also included with the Windows CE SDK. Please see the README_CE.txt file on the SDK for Windows CE CD for more information on installing and using this application.

Chapter 2

Installation

2.1 Installation - MS-Windows

Installation is preformed via a setup program. The option exists to install to a user specified hard drive and directory. In the example below, the CD-ROM drive is assumed to be drive E: and the installation target hard drive is assumed to be drive C:.

1. Startup Microsoft Windows.
2. Insert the Aware JPEG2000 CD-ROM into the CD-ROM drive.
3. Open the E: drive in Windows Explorer and select Setup.exe.
4. The Setup program will begin. Select a directory on your hard disk to install to. The default directory is C:\Program Files\Aware\Aware JPEG2000 Codec.
5. The Setup program will install the files on you hard disk.
6. Select Finished when prompted.

2.1.1 Installed Components - MS-Windows

The SDK installs various files in the subdirectories it creates. The subdirectories and their contents are listed below, along with a short description of the contents of each directory or file.

bin Executables and dynamic libraries.

- **awj2k.dll** The full Aware JPEG2000 Compression/Decompression Dynamically-Linked Library (DLL).
- **J2K_Tool.exe** Aware JPEG2000 Compression/Decompression Demonstration Tool executable.

AWJ2kCtrl.dll Aware JPEG2000 ActiveX DLL

AwJ2kCtrl_Demo.exe Aware JPEG2000 Visual Basic interface demo executable.

j2kdriver.exe Command line executable for the JPEG2000 Library.

j2k-jp2-decode-only\awj2k.dll A limited version of the library that can only decode JPEG 2000 images into raw images.

j2k-jp2-raw-only\awj2k.dll A limited version of the library that can both compress, decompress, and reformat JPEG 2000, but only reads and writes J2K, JP2, and raw file formats.

all-file-formats\awj2k.dll The full version of the library that can read and write any of the file formats mentioned in this manual (see page 9).

---

lib Linkable libraries.

awj2k.lib Aware JPEG2000 Compression/Decompression Library link file (required for development).

---

include C and C++ header files.

j2k.h Aware JPEG2000 Compression/Decompression Library header file (required for development).

aw_j2k_dtypes.h Auxiliary include file.

aw_j2k_errors.h Auxiliary include file.

---

docs Library documentation.

j2k-sdk-manual.pdf Electronic documentation (this document).

release_notes.txt Release notes and latest information.

---

images Sample images in various formats. These images are provided for demonstration purposes only and are not to be redistributed or transferred to any third parties.

---

src Files to demonstrate the use of the Aware JPEG2000 Codec. This directory is further subdivided into:

Command Line Tools Source code, in C, implementing a command-line interface to the library.

j2kdriver.c Example C source file.

j2k.h JPEG2000 header file.

aw_j2k_dtypes.h Data type definition file.

aw_j2k_errors.h Error message definition file.

get_precise_time.c Timing routine source.

get_precise_time.h Timing routine prototype.

awj2k.lib JPEG2000 library file required for compilation.

makefile Configuration file for nmake.exe for building the command line program.

make.bat a convenience batch file for running nmake.exe with default arguments.

JPEG2000 Tool Source code, in C++, implementing a graphical interface to the library.

AwJ2kCtrl_Demo Source code, in Visual Basic, implementing a graphical interface to the library.

AwJ2kCtrl_Demo.exe The precompiled demonstration application

AwJ2kCtrl_Demo.vbp The VB Project file for this demo

frmMain.frm Main form for program. Contains user interface and processing code.

frmMain.frx Binary components for frmMain.frm.

frmImageWindow.frm Form used to display images.

IE ActiveX Example HTML for using the IE ActiveX control.

Example.htm Sample HTML code showing how to embed a JPEG 2000 image in the browser.

Girl.j2k A sample image used in this example.

Netscape Plugin Example HTML for using the Netscape plugin.

Example.htm Sample HTML code showing how to embed a JPEG 2000 image in the browser.

Girl.j2k A sample image used in this example.

2.2 Installation - UNIX

Insert the Aware JPEG2000 CD-ROM into the CD-ROM drive. Copy contents of the CD to the user-specified destination on the hard drive, or use the files directly from the CD.

2.2.1 Installed Components - UNIX

The SDK is comprised of various files organized into several directories. These directories and their contents are listed below, along with a short description of the contents of each directory or file.

bin Executables and dynamic libraries.

- **j2kdriver** Command line executable for the JPEG2000 Library.

---

lib Linkable libraries.

- **libawj2k.a** Aware JPEG2000 static library.
- **libawj2k.so.2.0.1** Aware JPEG2000 dynamic library.
- **j2k-jp2-decode-only** A limited version of the library that can only decode JPEG 2000 images into raw images.
  - **libawj2k.a** Aware JPEG2000 static library.
  - **libawj2k.so.2.0.1** Aware JPEG2000 dynamic library.
- **j2k-jp2-raw-only** A limited version of the library that can both compress, decompress, and reformat JPEG 2000, but only reads and writes J2K, JP2, and raw file formats.
  - **libawj2k.a** Aware JPEG2000 static library.
  - **libawj2k.so.2.0.1** Aware JPEG2000 dynamic library.
- **all-file-formats** The full version of the library that can read and write any of the file formats mentioned in this manual (see page 9).
  - **libawj2k.a** Aware JPEG2000 static library.
  - **libawj2k.so.2.0.1** Aware JPEG2000 dynamic library.

---

include C and C++ header files.

- **j2k.h** Aware JPEG2000 Compression/Decompression Library header file (required for development).
- **aw_j2k_dtypes.h** Auxiliary include file.
- **aw_j2k_errors.h** Auxiliary include file.

---

docs Library documentation.

- **j2k-sdk-manual.pdf** Electronic documentation (this document).
- **release_notes.txt** Release notes and latest information.

---

images Sample images in various formats. These images are provided for demonstration purposes only and are not to be redistributed or transferred to any third parties.

---

src Source code, in C, implementing a command-line interface to the library.

j2kdriver.c Example C source file.

Makefile Platform-independent configuration file for make for building the command line program.

compiler.mak Further configuration file for make, containing the platform-specific build variables.

2.3 Demo Imagery

This program is provided with several images covering the spectra of expected input imagery types and formats. These images are provided for demonstration purposes only and are not to be redistributed or transferred to any third parties.

2.4 Multiple Library Versions

Multiple versions of the JPEG 2000 library are provided. All of these versions implement the full API, as described in this document, with the exception that different subsets of the supported file format list are implemented.

One version of the library provides support for the full set of file formats. This includes support for the JPEG 2000 formats (the J2K codestream and the JP2 format), as well as the raw, BMP, DICOM, JPEG, PNM, PGX, Targa, and TIFF formats. This version of the library, provided in the `all-file-formats` subdirectory of the installation, is the default version of the library.

The second version of the library includes support only for the JPEG 2000 formats (J2K and JP2) and for raw images. This version, provided in the `j2k-jp2-raw-only` subdirectory of the installation, will compress, decompress, and reformat JPEG 2000 images, but will neither input nor output BMP, DICOM, JPEG, PNM, PGX, Targa, and TIFF formats.

The third version of the library includes support only for decoding JPEG 2000 (J2K and JP2). This version, provided in the `j2k-jp2-decode-only` subdirectory of the installation, will only accept J2K and JP2 formats as input, and produce only raw as output.

The non-default libraries are provided as smaller replacements for the full library. The dynamically-linked versions of the libraries are drop-in replacements — no recompilation is needed to switch among them. The statically-linked versions require a relink of the application in order to switch among them.

Chapter 3

Library Overview

The Aware JPEG2000 library manipulates images, turning input images into output images under the control of the input image options and output image options. As illustrated in Figure 3.1, the library accepts images in several formats as the input image. The input image is then decoded to an intermediate internal format, and then encoded to the selected output image format. JPEG 2000 Compression, decompression and reformatting can all be performed using this general framework, as described in the next section.

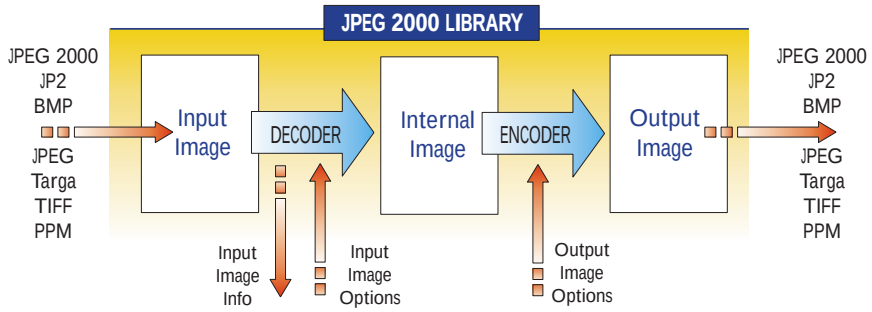

Figure 3.1: The general structure of the JPEG 2000 Library

Several sets of functions are provided to query the input images and to control both the decoding and the encoding processes.

The functions used to create and destroy the image objects used by the library are described in Chapter 4. The object creation function must be called prior to calling any of the functions described in Chapters 5 and 6, while the object destruction function should be called after all other functions.

The functions to query and manipulate the input image, those functions that affect the decoder, are described in Chapter 5. These functions can return such information as the width and the height of the input image, as well as set such parameters as the resolution level which will be extracted from a JPEG 2000 input image.

Input Format	**Output Format**	**Operation**	**Parameters**
BMP, JPEG, PPM, PGX, TIFF, Targa	JPEG 2000	Compression	JPEG 2000 Ouput parameters.
JPEG 2000	BMP, JPEG, PPM, PGX, TIFF, Targa	Decompression	JPEG 2000 Input Parameters[1].
JPEG 2000	JPEG 2000	Reformatting	JPEG 2000 Input and Output parameters

Table 3.1: Library operations as defined by the input and output image formats

The functions that affect the output image and the encoding process are described in Chapter 6. These functions can set such parameters as the compression ratio used for a JPEG 2000 image or the quality factor used for a JPEG image.

Each set of functions (input and output) is further divided by the image format that it applies to. Some functions, such as aw_j2k_get_input_image_info(), apply to all image formats; others, such as aw_j2k_get_input_j2k_progression_info(), apply only to input images that are in the JPEG 2000 format.

3.1 Compression, Decompression, and Reformatting

As show in Table 3.1, the choice of performing a JPEG 2000 compression, decompression, or reformatting operation is simply a function of the input and output image formats. **JPEG 2000 compression** is performed if the output image format is JPEG 2000 and the input image format is not. **JPEG 2000 decompression** is performed if the input image format is JPEG 2000 and the output image format is not. Finally, **JPEG 2000 reformatting** is performed if both the input and output image formats are JPEG 2000.

Also shown in Table 3.1 are the paramaters that control each of these operations.

For a **JPEG 2000 compression** operation, the JPEG2000 Output Image parameters are set to control the compression process.

Conversely, for a **JPEG 2000 decompression** operation, the JPEG2000 Input Image parameters are set to control the decompression process.

Finally, for a **JPEG 2000 reformatting** operation, both the JPEG2000 input and output parameters must be set.

[1]Note: if the output image type is a JPEG image, the JPEG image quality, which is a JPEG Output Parameter, may also be set.

3.2 Function Naming Rationale

The key reason for structuring the library as described above is that the library is designed to perform not only compression and decompression of images, but also **reformatting of JPEG2000 images**. Reformatting of JPEG2000 images involves using one JPEG000 image to extract or create another JPEG2000 image, which represents a different version of the same image.

In its simplest form, reformatting may involve a simple truncation of the image file, for a situation where you want a subresolution image and the JPEG2000 image is organized progressively by resolution, for example. At the other extreme, reformatting may involve a full decompression and compression of the image, for a situation where you want to change the wavelet filter, for example. Other reformatting operations may involve a partial decompression and partial compression operation.

By structuring the library as described above, all the reformatting situations can be handled using two simple steps:

1. Set the input image to a JPEG2000 image and set the input image options, as desired.
2. Set the output image to be a JPEG2000 image and set the output image options, as desired.

The library user does not need to decide whether a (partial or full) decompression and recompression are necessary, as the library handles that decision process.

3.3 Examples

In the rest of this section, example source code (in the C language) is provided which give simple demonstrations of using the library to perform compression, decompression and reformatting operations. More detailed example source codes in a variety of languages (C, C++, and Visual Basic) is provided with this SDK.

3.3.1 JPEG 2000 Compression Example

Input Format	**Output Format**	**Operation**	**Parameters**
BMP, JPEG, PPM, PGX, TIFF, Targa	JPEG 2000	Compression	JPEG2000 Ouput parameters.

Table 3.2: Input and Output image types, and the relevant parameters for compression

As seen in Table 3.2, to perform a JPEG 2000 compression operation on an image, the library user needs to:

1. provide the library with an input image which is not in the JPEG 2000 format;
2. set the output image format to JPEG 2000; and
3. set the relevant JPEG 2000 Output parameters.

The following code fragment, written in C, shows how to achieve JPEG 2000 compression with the library. The fragment will tell the library to read a BMP-format image from a file, set the output image format to JPEG 2000, set the JPEG 2000 compression ratio to 10:1, and then ask the library to write the new image to a new file.

```
int retval;
aw_j2k_object *j2k_object;
char *input_file_name = "sample_image.bmp";
char *output_file_name = "sample_image.j2k";

/* first, the JPEG 2000 library object must be initialized */
j2k_object = NULL;
retval = aw_j2k_create(&j2k_object);

/* Read in the input image */
retval = aw_j2k_set_input_image_file(j2k_object,
                                     input_file_name);

/* Request a JPEG 2000 format for the output image */
retval = aw_j2k_set_output_type(j2k_object, AW_J2K_FORMAT_J2K);

/* Set the relevant Output Image parameters */
/* set the compression ratio to 10:1 */
retval = aw_j2k_set_output_j2k_ratio(j2k_object, 10.0F);

/* Request the output image */
retval = aw_j2k_get_output_image_file(j2k_object,
                                      output_file_name);

/* finally, deallocate the library memory */
retval = aw_j2k_destroy(j2k_object);
```

3.3.2 JPEG 2000 Decompression Example

As seen in Table 3.3, to perform a JPEG 2000 decompression on an image, the library user needs to:

Input Format	Output Format	Operation	Parameters
JPEG 2000	BMP, JPEG, PPM, PGX, TIFF, Targa	Decompression	JPEG2000 Input Parameters

Table 3.3: Input and Output image types, and the relevant parameters for decompression

1. provide the library with an input image which is in the JPEG 2000 format,
2. set the output image format to any format but the JPEG 2000 format, and
3. set the relevant input JPEG 2000 parameters.

The following code fragment, written in C, shows how to achieve JPEG 2000 decompression with the library. The fragment will read in an image in JPEG 2000 format, set the output format to BMP, set the decoding resolution to quarter resolution, and then ask the library to write the output image.

```
int retval;
aw_j2k_object *j2k_object;
char *input_file_name = "sample_image.j2k";
char *output_file_name = "sample_image.bmp";
int resolution_level = 2; /* quarter resolution */
int full_transform = 0; /* leave image at quarter size */

/* first, the JPEG 2000 library object must be initialized */
j2k_object = NULL;
retval = aw_j2k_create(&j2k_object);

/* next, read in the input image */
retval = aw_j2k_set_input_image_file(j2k_object,
                                input_file_name);

/* next, request the desired format for the output image */
retval = aw_j2k_set_output_type(j2k_object, AW_J2K_FORMAT_BMP);

/* Set the relevant Input Image parameters */
/* next, request quarter-resolution decoding and reduced resolution
output. If the third flag were true (1), the resulting image
would be full size but blurred (as only the data for the quarter
resolution was decoded from the file). */
retval = aw_j2k_set_input_j2k_resolution_level(j2k_object,
```

```
                                        resolution_level,
                                        full_transform);

 /* next, request the output image */
retval = aw_j2k_get_output_image_file(j2k_object,
                                      output_file_name);
/* finally, deallocate the library memory */
retval = aw_j2k_destroy(j2k_object);
```

3.3.3 JPEG 2000 Reformatting Example

Input Format	**Output Format**	**Operation**	**Parameters**
JPEG 2000	JPEG 2000	Reformatting	JPEG2000 Input and Output parameters

Table 3.4: Input and Output image types, and the relevant parameters for reformatting

As seen in Table 3.4, to perform a JPEG 2000 reformatting on an image, the library user needs to:

1. provide the library with an input image which is in the JPEG 2000 format,
2. set the output image format to JPEG 2000 as well, and
3. set the relevant input and output JPEG 2000 parameters.

The following code fragment, written in C, shows how to reformat a JPEG 2000 image into a different JPEG 2000 image using the JPEG 2000 library. The code reads in an image in the JPEG 2000 format, sets the output format to be JPEG 2000 as well, sets the number of color channels decoded from the image to 1 (i.e. changes the file from full color to grayscale), changes the progression order, and finally writes out the reformatted image.

```
int retval;
aw_j2k_object *j2k_object;
char *input_file_name = "sample_image.j2k";
char *output_file_name = "sample_image.bmp";
int color_channels = 3; /* drop two color channels */
int color_transform = 0; /* do not perform color rotation */

/* first, the JPEG 2000 library object must be initialized */
j2k_object = NULL;
retval = aw_j2k_create(&j2k_object);
```

```
/* next, read in the input image */
retval = aw_j2k_set_input_image_file(j2k_object,
                                input_file_name);

/* next, request the desired format for the output image */
retval = aw_j2k_set_output_type(j2k_object, AW_J2K_FORMAT_J2K);

/* Set the relevant Input Image parameters */
/* Request a grayscale version of the image. This is done by
requesting to drop the last two color channels, leaving only
one (this, of course, assumes that the original image was in
color). */
retval = aw_j2k_set_input_j2k_color_level(j2k_object,
                                    color_channels,
                                    color_transform);

/* Set the relevant Output Image parameters */
/* Set the progression order to resolution-major */
retval = aw_j2k_set_output_j2k_progression_order(j2k_object,
           AW_J2K_PO_RESOLUTION_LAYER_COMPONENT_POSITION);

/* next, request the output image */
retval = aw_j2k_get_output_image_file(j2k_object,
                                 output_file_name);

/* finally, deallocate the library memory */
retval = aw_j2k_destroy(j2k_object);
```

Chapter 4

Global Functions

Most of the functions in this manual pertain either to the input image (as described in Chapter 5) or the output image (as described in Chapter 6). A small number of functions deals with the library object itself, and are described in this Chapter.

In the first section, the creation and destruction of the library object are described. The second section details memory management issues with using the library. The third section describes the remaining global functions, ones that set library options.

4.1 Object Creation and Destruction Functions

Every function in this manual operates on an object called the `aw_j2k_object`. The functions in this section create this object and destroy it.

One of the available object creation functions must be called before any other library function. It initializes the object and allocates the needed intial memory structures.

The object destruction function must be called after all other library functions. It will deallocate any remaining memory structures.

4.1.1 aw_j2k_create
aw_j2k_create_with_memory_manager

Description

The aw_j2k_object that is used as an argument to all the other functions is created by a call to either of two functions. Calling aw_j2k_create() will create the object and use the default memory allocation and deallocation functions, which are malloc(size) and free(ptr) on Unix and Unix-like platforms, and HeapAlloc(GetProcessHeap(), 0, size) and HeapFree(GetProcessHeap(), 0, ptr) on Win32.

It is possible to create the object and provide a custom set of memory allocation and deallocation functions for the JPEG 2000 library, using the aw_j2k_create_with_memory_manager() function call. This function takes three additional arguments: a pointer to user-supplied data, an allocation function, and a deallocation function. The allocation function takes two arguments, a pointer to the user supplied data, and an indication of the size of the desired allocation. The library, when calling this function, will provide it with the pointer to the data given as the 2nd argument to the aw_j2k_create_with_memory_manager() call. Similarly, the deallocation function, when called by the library, will be provided with a pointer to the user data, as well as a pointer to the memory that should be deallocated.

C Function Prototype

```
aw_j2k_create(aw_j2k_object **j2k_object)
aw_j2k_create_with_memory_manager(
   aw_j2k_object **j2k_object,
   void *user_data,
   void *(*alloc_function)(void *, size_t),
   void (*free_function)(void *, void *))
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object

user_data — Pointer to arbitrary user-supplied data

alloc_function — Pointer to a memory allocation function

free_function — Pointer to a memory deallocation function

Example Code

This sample code shows how to create the aw_j2k_object using the aw_j2k_create_with_memory_manager() function and using custom memory allocation and deallocation functions. In this sample code, which is written with the Win32 API, a private

memory heap is created and special memory functions are used to allocate and deallocate memory on this heap. The `aw_j2k_object` is created with those functions, and all the library-internal memory allocations will be performed using them.

```
void *
sample_allocate_memory(void *user_data, size_t size)
{
  HANDLE j2k_heap;
  j2k_heap = (HANDLE) user_data;
  return HeapAlloc(j2k_heap, 0, size);
}

void
sample_free_memory(void *user_data, void *ptr)
{
  HANDLE j2k_heap;
  j2k_heap = (HANDLE) user_data;
  HeapFree(j2k_heap, 0, ptr);
}

void
sample_main_body(void)
{
  HANDLE j2k_heap;
  aw_j2k_object *j2k_object;
  int retval;

  j2k_heap = HeapCreate(0, 0, 0);
  j2k_object = NULL;

  retval =
    aw_j2k_create_with_memory_manager(
        &j2k_object, (void *)j2k_heap,
        sample_allocate_memory, sample_free_memory);

  /* other J2K functions are called here */

  retval = aw_j2k_destroy(j2k_object);
  j2k_object = NULL;

  HeapDestroy(j2k_heap);
  j2k_heap = NULL;
}
```

4.1.2 aw_j2k_destroy

Description

This function destroys the aw_j2k_object. It deallocates all memory associated with the object, closes any open files, and performs all other necessary cleanup. It should be the last function called on the object, as it is no longer a valid object after this call.

See also the code sample given with the description of aw_j2k_create() (page 21) for the usage of this function.

C Function Prototype

```
aw_j2k_destroy(aw_j2k_object *j2k_object)
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object. The object will no longer be valid after this call.

4.2 Memory Management

Several of the functions described in this manual, listed below, require a buffer into which their output will be written. This buffer may be allocated by the calling routine, or by the library function. Whether the buffer is allocated by the library or not is controlled by the caller. If the buffer is allocated by the library, it must be freed using the `aw_j2k_free` function, as described on page 26.

The functions that may allocate memory are:

- `aw_j2k_get_output_image`, described on page 96
- `aw_j2k_get_output_image_type`, described on page 97
- `aw_j2k_get_output_image_raw`, described on page 98
- `aw_j2k_get_input_j2k_comments`, described on page 61

Each of these functions accepts a buffer pointer and a length pointer, among other arguments. If the buffer pointer points to a null buffer, the library will allocate the memory and return its length using the length pointer. If the buffer pointer points to a non-null buffer, its length must be stored in the length pointer. The library will check the length. If the buffer is too short, the library will store the required length in the length pointer and will return the error code `AW_J2K_ERROR_BUFFER_TOO_SHORT`. If the library is successful, it will write the new image to the buffer and store the length of the written data in the length pointer.

Example Code

The first example uses `aw_j2k_get_output_image` (see page 96) to decompress a JPEG 2000-compressed image into `image_buffer`. This buffer is allocated by the library, and is freed using `aw_j2k_free` (see page 26).

```
void
library_allocates_buffer(void)
{
  aw_j2k_object *j2k_object;
  int retval;
  unsigned char *image_buffer;
  unsigned long int image_buffer_length;

  /* create the library object */
  retval = aw_j2k_create(&j2k_object);

  /* load an input image into the object */
  retval = aw_j2k_set_input_image_file(j2k_object, "inputimage.j2k");
```

```
  /* set the output image type */
  retval = aw_j2k_set_output_type(j2k_object, AW_J2K_FORMAT_BMP);

  /* ensure that the library allocates this buffer */
  image_buffer = NULL;

  /* get the output image */
  retval = aw_j2k_get_output_image(j2k_object, &image_buffer,
                                   &image_buffer_length);

  /* to something with the output image */
  do_something_with_image(image_buffer, image_buffer_length);

  /* deallocate the image buffer */
  retval = aw_j2k_free(j2k_object, image_buffer);
}
```

The second example users the same `aw_j2k_get_output_image` function to decompress the image, but allocates `image_buffer` before the call. The buffer, which can be reused, is later deallocated by this code.

```
void
caller_allocates_buffer(void)
{
  aw_j2k_object *j2k_object;
  int retval;
  unsigned char *image_buffer;
  unsigned long int image_buffer_length;
  unsigned long cols, rows, bpp, channels;

  /* create the library object */
  retval = aw_j2k_create(&j2k_object);

  /* load an input image into the object */
  retval = aw_j2k_set_input_image_file(j2k_object, "inputimage.j2k");

  /* set the output image type */
  retval = aw_j2k_set_output_type(j2k_object, AW_J2K_FORMAT_BMP);

  /* find out the size of the image */
  retval = aw_j2k_get_input_image_info(j2k_object, &rows, &cols, &bpp,
                                       &channels);

  /* calculate the buffer length needed to hold the image */
```

```
  if (channels == 1) {
    /* grayscale case -- the BMP image has a color map */
    image_buffer_length = rows * cols + 1078;
  }
  else {
    /* color case -- no color map present in the BMP */
    image_buffer_length = rows * cols * channels + 54;
  }

  /* allocate a buffer for the image */
  image_buffer = (unsigned char *)malloc(image_buffer_length);

  /* get the output image */
  retval = aw_j2k_get_output_image(j2k_object, &image_buffer,
                                   &image_buffer_length);

  if (retval == NO_ERROR) {
    /* success */

    /* to something with the output image */
    do_something_with_image(image_buffer, image_buffer_length);
  }
  else if (retval == AW_J2K_ERROR_BUFFER_TOO_SHORT) {
    /* did not allocate enough memory; should start over with longer
     * buffer.
     */
  }
  else {
    /* handle other errors. */
  }

  /* deallocate the image buffer */
  free(image_buffer);
}
```

4.2.1 aw_j2k_free

Description

Free the buffer given as argument. The buffer must have been allocated using one of the following functions:

- aw_j2k_get_output_image, described on page 96
- aw_j2k_get_output_image_type, described on page 97
- aw_j2k_get_output_image_raw, described on page 98
- aw_j2k_get_input_j2k_comments, described on page 61

See also the discussion on memory management issues, beginning on page 23.

C Function Prototype

```
aw_j2k_free(aw_j2k_object *j2k_object, void *pointer)
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`pointer` — Pointer to the memory buffer that is to be deallocated.

4.3 Miscellaneous Functions

4.3.1 aw_j2k_get_library_version

Description

Returns the version of the library, both as a number and as a string. The version number is returned as a decimal number; its format is MMNNPPBB, where MM, NN, PP, and BB are the decimal digits making it up.

C Function Prototype

```
aw_j2k_get_library_version(
   unsigned long *library_version,
   char *version_string,
   unsigned long version_string_buffer_length
)
```

Function Parameters

library_version — The returned version of the library. This value may be left NULL; no value will be stored in a NULL pointer and no error will be issued.

version_string — A pointer to a preallocated array, of minimum length AW_J2K_LIBRARY_VERSION_STRING_MINIMUM_LENGTH, into which the version description string will be written. This can also be NULL, in which case no string will be written, and no error will be issued.

version_string_buffer_length — The length of the version_string buffer.

Example Code

This sample code shows one way to programatically extract the fields from the library version returned by this function call.

```
void
sample_interpret_library_version(void)
{
  unsigned long library_version; /* The returned version       */
  unsigned long major_number;    /* The version's major number */
  unsigned long minor_number;    /* The version's minor number */
  unsigned long patch_level;     /* The version's patch level  */
  unsigned long build_number;    /* The version's build number */
```

```
  char
library_version_string[AW_J2K_LIBRARY_VERSION_STRING_MINIMUM_LENGTH];

  (void)aw_j2k_get_library_version(
         &library_version,
         library_version_string,
         sizeof(library_version_string));

  major_number = library_version / 1000000;
  minor_number = (library_version / 10000) % 100;
  patch_level  = (library_version / 100) % 100;
  build_number = library_version % 100;
}
```

4.3.2 aw_j2k_reset_all_options

Description

Resets all options to their original default values.

C Function Prototype

```
aw_j2k_reset_all_options(aw_j2k_object *j2k_object)
```

ActiveX Visual Basic Function Prototype

```
reset_all_options() as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

4.3.3 aw_j2k_set_internal_option

Description

Allows access to internal implementation options.

Currently, the only available option is the selection of fixed point or floating point arithmetic. Floating point arithmetic produces higher quality images but may be slower than the fixed point arithmetic on some platforms.

C Function Prototype

```
aw_j2k_set_internal_option(
   aw_j2k_object *j2k_object,
   int option,
   int value
)
```

ActiveX Visual Basic Function Prototype

```
set_internal_option(
   option as long,
   value as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`option` — The selected internal option. The following constants are defined in j2k.h:

- `AW_J2K_INTERNAL_ARITHMETIC_OPTION` – used to select the type of arithmetic implementation.

  Allowable values for this option are:

  - `AW_J2K_INTERNAL_USE_FLOATING_POINT` – to select the use of floating point arithmetic.
  - `AW_J2K_INTERNAL_USE_FIXED_POINT` – to select the use of fixed point arithmetic.

  The default value for this option varies by platform. On Unix platforms, the default is to use floating point arithmetic. On Windows, the default is to use fixed point.

`value` — Specifies the new value for the selected option. The allowable values for each option are listed above.

4.3.4 aw_j2k_register_callbacks

Description

This function allows the user to register callback functions that the SDK will periodically call. The first function, the `TestAbortProc` callback, allows the user to stop a lengthy compression or decompression process; the rest of the functions allow the user to create and periodically update a progress indicator.

These callback functions, with the exception of the `TestAbortProc` callback, are not activated.

The first argument to each of these callback functions is a void pointer; for this argument, the SDK will supply the same user-supplied data that is provided to the `aw_j2k_create_with_memory_manager` (see page 20 for description).

C Function Prototype

```
aw_j2k_register_callbacks(
   aw_j2k_object *j2k_object,
   int (*TestAbortProc)(void *),
   void (*UpdateProgressProc)(void *),
   void (*SetProgressDoneProc)(void *, int done),
   void (*SetProgressTotalProc)(void *, int total),
   int (*GetProgressDoneProc)(void *)
)
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`TestAbortProc` — A pointer to a function that takes a single argument, a void pointer, and returns an integer that indicates whether the current SDK process should be stopped or continued. If the TestAbortProc function returns zero, the current SDK function continues; if this function returns a non-zero value, the SDK function returns immediately with the `AW_J2K_ERROR_USER_ABORT` status.

`UpdateProgressProc` — A pointer to a function that takes a single argument, a void pointer. This function is called by the SDK to increment a progress indicator by a single work unit.

`SetProgressDoneProc` — A pointer to a function that takes two arguments, a void pointer, and an integer. This function is called by the SDK to indicate the total amount of work units that has been completed thus far.

`SetProgressTotalProc` — A pointer to a function that takes two arguments, a void pointer, and an integer. This function is called by the SDK to indicate the maximum amount of work units that will be done, i. e. the total size of the progress indicator.

`GetProgressDoneProc` — A pointer to a function that takes a single argument, a void pointer, and returns an integer that indicates the current state of the progress indicator, in completed work units.

Chapter 5

Input Image Functions

The functions described in this section operate on the input image of the library, as shown in Figure 5.1. The functions described in this section are organized, based on their functionality, in the following 3 categories:

Image Reading Functions, which are used to read an input image into the object. These functions are described in Section 5.1.

Image Information Functions, which return information about the input image. They are described in Section 5.2.

JPEG2000 Decompression and Input Reformatting Functions, which are used to set parameters that control the decoding of the JPEG2000 image. They are described in Section 5.3.

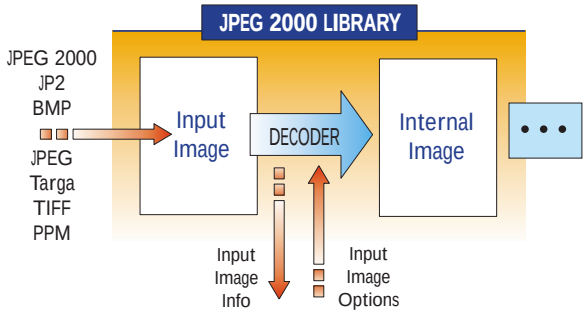

Figure 5.1: The input parameters of the Library

5.1 Image Reading Functions

The following input functions are used to read an input image into the object. These input image reading functions simply load the image data into the object. Compression, decompression or reformatting are not performed until the `aw_j2k_get_output_image` function is called.

5.1.1 aw_j2k_set_input_image

Description

Reads an image from a buffer into the aw_j2k_object. The type of the image is determined automatically, if possible.

C Function Prototype

```
aw_j2k_set_input_image(
   aw_j2k_object *j2k_object,
   unsigned char *image_buffer,
   unsigned long int image_buffer_length)
```

ActiveX Visual Basic Function Prototype

```
set_input_image(
   image_buffer as Variant(array of bytes)
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object where the input image will be stored.

image_buffer — The input image buffer. The image type will be detected automatically if it is in one of the following supported image types: JPEG2000, JPG, PPM, PGM, PGX, BMP, TGA, TIFF, DICOM.

image_buffer_length — The length of the input image buffer.

TIFF images that incorporate LZW compression and run-length coding are supported. BMP variants which incorporate run length or LZW compression are not supported.

Only the following data formats are supported within the DICOM files: JPEG, JPEG2000 (lossy and lossless) and uncompressed RAW.

5.1.2 aw_j2k_set_input_image_type

Description

Reads an image of a specific type from a buffer into the aw_j2k_object. This function should be called with a specific image type for images whose type is not correctly detected automatically using aw_j2k_set_input_image.

C Function Prototype

```
aw_j2k_set_input_image_type(
   aw_j2k_object *j2k_object,
   int image_type,
   unsigned char *image_buffer,
   unsigned long int image_buffer_length)
```

ActiveX Visual Basic Function Prototype

```
set_input_image_type(
   image_type as enumSupportImageTypes,
   image_buffer as Variant(array of bytes)
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object where the input image will be stored.

image_type — The image type of the input image. The following image types are supported: JPEG2000, JPG, PPM, PGM, PGX, BMP, TGA, TIFF, DICOM. The following constants are define in j2k.h:

- AW_J2K_FORMAT_TIF
- AW_J2K_FORMAT_PPM
- AW_J2K_FORMAT_PGX
- AW_J2K_FORMAT_BMP
- AW_J2K_FORMAT_TGA
- AW_J2K_FORMAT_JPG
- AW_J2K_FORMAT_J2K
- AW_J2K_FORMAT_DCM
- AW_J2K_FORMAT_DCMJ2K
- AW_J2K_FORMAT_DCMJPG

TIFF images that incorporate runlength coding and LZW compression are supported. DICOM file format is supported in both uncompressed raw, JPEG compressed format, and JPEG2000 compressed format (the last three constants allow for differentiation between the data formats within the DICOM file). Both 8 and 12 bit DICOM JPEG images are supported.

`image_buffer` — The input image buffer

`image_buffer_length` — The length of the input image data

5.1.3 aw_j2k_set_input_image_raw aw_j2k_set_input_image_raw_ex

Description

Reads a raw image into the aw_j2k_object. For images with more than 8 bits per channel, you can specify the endianness of the raw image using the function aw_j2k_set_input_raw_endianness on page 53.

There are two versions to this function. The first, aw_j2k_set_input_image_raw, reads raw data with each sample (that is, each color sample of each pixel) stored separately, padded to the next byte boundary. For example, an image with 3 color channels and 12-bit data per color channel per pixel is stored in 2 bytes per sample per pixel for a total of 6 bytes per pixel.

The second version, aw_j2k_set_input_image_raw_ex, allows the caller to specify how the data is padded, by specifying how many bits are allocated per pixel in addition to the number of bits stored per pixel. For the example image above, if the allocated bits parameter is specified as 48 (which is 16 bits per sample per pixel) than the default behavior of the first function is obtained. If the same image is not padded, the caller can specify the allocated bits as 36 (12 bits per sample per pixel).

C Function Prototype

```
aw_j2k_set_input_image_raw(
   aw_j2k_object *j2k_object,
   unsigned char *image_buffer,
   unsigned long int rows,
   unsigned long int cols,
   unsigned long int nChannels,
   unsigned long int bits_stored,
   BOOL bInterleaved
)

aw_j2k_set_input_image_raw_ex(
   aw_j2k_object *j2k_object,
   unsigned char *image_buffer,
   unsigned long int rows,
   unsigned long int cols,
   unsigned long int nChannels,
   unsigned long int bits_allocated,
   unsigned long int bits_stored,
   BOOL bInterleaved
)
```

ActiveX Visual Basic Function Prototype

```
set_input_image_raw(
   image_buffer as Variant(array of bytes)
   rows as long,
   cols as long,
   nChannels as long,
   bits_stored as as long,
   bInterlaved as Boolean
) as long

set_input_image_raw_ex(
   image_buffer as Variant(array of bytes)
   rows as long,
   cols as long,
   nChannels as long,
   bits_allocated as as long,
   bits_stored as as long,
   bInterlaved as Boolean
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object` where the input image will be stored.

`image_buffer` — The input image buffer.

`rows` — The number of rows in the full input image. The maximum allowable value is $2^{32} - 1$.

`columns` — The number of columns in the full input image. The maximum allowable value is $2^{32} - 1$.

`nChannels` — number of color channels in the image.

`bpp_allocated` — total bit depth of the data (inlusive of padding) in each pixel in bits (sum of all channels). Supported pixel depth: up to 16 bits per channel, signed or unsigned. All channels are assumed to have the same bit depth. The difference between this parameter and `bpp_stored` is the number of padding bits per pixel that will be ignored during compression.

`bpp_stored` — total bit depth of the data (exclusive of padding) in each pixel in bits (sum of all channels). Supported pixel depth: up to 16 bits per channel, signed or unsigned. All channels are assumed to have the same bit depth.

`bInterleaved` — This parameter should be set to true if the input image is byte interleaved by channel (ex – R,G,B, R,G,B). Set it to false, if all the bytes from the first channels are in a continuous block, followed immediately by the bytes from the second channel as a continuous block, and so on.

5.1.4 aw_j2k_set_input_image_raw_channel

Description

Reads an additional channel of data from a raw image into the aw_j2k_object.

The size of the original image should be specified using aw_j2k_set_input_raw_information (see page 50). The endianness of the raw data can be specified using aw_j2k_set_input_raw_endianness (see page 53). The bitdepth of the data in this channel can be specified using aw_j2k_set_input_raw_channel_bpp (see page 52). The subsampling of this channel relative to the full image can be specified using aw_j2k_set_input_raw_channel_subsampling (see page 54).

C Function Prototype

```
aw_j2k_set_input_image_raw_channel(
   aw_j2k_object *j2k_object,
   unsigned long int channel,
   unsigned char *channel_buffer,
   unsigned long int buffer_length
)
```

ActiveX Visual Basic Function Prototype

```
set_input_image_raw_channel(
   channel as long,
   channel_buffer as Variant(array of bytes)
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object where the input image will be stored.

channel — The channel index for which data is provided.

channel_buffer — The input channel data buffer.

buffer_length — The length of the data buffer, in bytes.

5.1.5 aw_j2k_set_input_image_raw_region

Reads a region of a raw (i.e. headerless) input image from a memory buffer. This function enables an image to be compressed incrementally into a single, tiled, JPEG-2000 image. The compressed tiles corresponding to the region that was provided by this function can be retrieved using one of aw_j2k_get_output_image (as described on page 96), aw_j2k_get_output_image_type (page 97), aw_j2k_get_output_image_file (page 101), or aw_j2k_get_output_image_file_type (page 102). The function aw_j2k_set_output_j2k_header_option (page 166) is used to control whether the JPEG2000 headers or footers are added to the compressed data.

This function should only be called after the image dimensions are described using the function aw_j2k_set_input_raw_information (page 50).

This function should only be used when the output image type is either J2K or JP2.

C Function Prototype

```
aw_j2k_set_input_image_raw_region(
    aw_j2k_object *j2k_object,
    unsigned long int rows_offset,
    unsigned long int cols_offset,
    unsigned long int rows,
    unsigned long int cols,
    unsigned long int nChannels,
    BYTE *buffer,
    BOOL bInterleaved
)
```

ActiveX Visual Basic Function Prototype

```
set_input_image_raw_region(
    rows_offset as long,
    cols_offset as long,
    rows as long,
    cols as long,
    nChannels as long,
    buffer as Variant (array of bytes)
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

`row_offset` — Specifies the row location of the buffer in the the RAW image.

`col_offset` — Specifies the column location of the buffer in the the RAW image.

`rows` — The number of rows in the buffer.

`cols` — The number of columns in the buffer.

`nChannels` — The number of channels

`buffer` — The input image buffer

`bInterleaved` — This parameter should be set to true if the input buffer is byte interleaved by channel (for example, R,G,B, R,G,B). Set it to false, if all the bytes from the first channels are in a continuous block, followed immediately by the bytes from the second channel as a continuous block, and so on.

Example Code

This example code implements a loop that reads a region of an image, compresses this data, and repeats — that is, the compression of the image is incremental. It is worth noting that if the intention is to compress the image as a whole after feeding it to the library in pieces, both the call to `aw_j2k_get_output_image` *and* the call to `aw_j2k_set_output_j2k_header_option` must be taken out of the loop. In fact, since the image is compressed once in this case, both the main header and the end-of-codestream marker are needed, so the call to `aw_j2k_set_output_j2k_header_option` is redundant.

```
/* the object is created */
aw_j2k_create(&j2k_object);

/* inform the library of the full image dimensions */
aw_j2k_set_input_raw_information(
  j2k_object,
  FULL_IMAGE_ROWS,
  FULL_IMAGE_COLS,
  FULL_IMAGE_CHANNELS,
  FULL_IMAGE_BIT_DEPTH,
  FULL_IMAGE_INTERLEAVING);
/* the last flag, the interleave boolean flag, is actually
 * unnecessary in this scenario, since it is also signalled in the
 * aw_j2k_set_input_image_raw_region call. */

/* set up the default encoding options */

/* the output image type must be a JPEG2000 image */
```

```
aw_j2k_set_output_image_type(
  j2k_object, AW_J2K_FORMAT_J2K);

/* Important: the tile size must be set before the first call to
 * aw_j2k_get_output_image(), or else the entire image will be
 * compressed at once. */
aw_j2k_set_output_j2k_tile_size(
  j2k_object, target_tile_width,
  target_tile_height);

/* other default options can be set here, e.g. the target
 * compression ratio */
aw_j2k_set_output_j2k_ratio(
  j2k_object, target_ratio);

/* now loop over the regions of the image */
do {
  /* obtain data for next region -- this could be a tile or multiple
   * tiles, some number of scan lines, etc. */
  get_next_region(
    &row_location, &col_location,
    &rows_in_region, &cols_in_region,
    &channels_in_region, &region_data_buffer,
    &interleave_flag,
    &first_region_flag, &last_region_flag);

  /* the region data is now loaded into the library */
  aw_j2k_set_input_image_raw_region(
    j2k_object, row_location, col_location,
    rows_in_region, cols_in_region,
    channels_in_region, region_data_buffer,
    interleave_flag);

  /* The first_region_flag indicates whether this region is the
   * first one or not; if it is, the main headers must be prepended
   * to the JPEG2000 data.  Similarly, the last_region_flag
   * indicates whether this region is the last one; the last one
   * requires that the end-of-codestream marker must be appended to
   * the JPEG2000 data.
   */
  aw_j2k_set_output_j2k_header_option(
    j2k_object, first_region_flag,
    last_region_flag);
```

```
  /* Setting the compressed_region_buffer to null ensures that the
   * library allocates a buffer of the right size.  Alternatively, a
   * buffer of the proper size can be allocated;
   * compressed_region_buffer_length must then be set to the size of
   * that buffer. */
  compressed_region_buffer = NULL;
  compressed_region_buffer_length = 0;

  /* the region can now be compressed */
  aw_j2k_get_output_image(
    j2k_object, &compressed_region_buffer,
    &compressed_region_buffer_length);

  /* something is done with the data here */
  do_something_with_compressed_data(
    compressed_region_buffer,
    compressed_region_buffer_length);

  /* the compressed data can now be disposed of */
  aw_j2k_free(
    j2k_object, compressed_region_buffer);
} while (!last_region_flag);

/* finally, the j2k_object can be disposed of */
aw_j2k_destroy(j2k_object);
```

5.1.6 aw_j2k_set_input_image_file

Description

Reads a image from a file into the aw_j2k_object. The type of the image is determined automatically, if possible.

C Function Prototype

```
aw_j2k_set_input_image_file(
   aw_j2k_object *j2k_object,
   char *filename)
```

ActiveX Visual Basic Function Prototype

```
set_input_image_file(filename as String) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object where the input image will be stored.

filename — Name of the image file. The following image types are supported and will be automatically detected: JPEG2000, JPG, PPM, PGM, PGX, BMP, TGA, TIFF, DICOM.

Note that the filename extentions are *not* used to determine the input type.

Comment

If the input image type is not recognized, it returns AW_J2K_ERROR_UNKNOWN_IMAGE_TYPE

5.1.7 aw_j2k_set_input_image_file_type

Description

Reads a image of a specific type from a file into the aw_j2k_object. This function is identical to the previous function (aw_j2k_set_input_image_file, page 46), except that it allows the user to explicitly specify the input type. This function should be used in situations when the image type was not correctly detected using aw_j2k_set_input_image_file.

C Function Prototype

```
aw_j2k_set_input_image_file_type(
   aw_j2k_object *j2k_object,
   char *filename,
   int image_type
)
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object where the input image will be stored.

filename — Name of the image file.

image_type — The image type. The following image types are supported: JPEG-2000, JPG, PPM, PGM, PGX, BMP, TGA, TIFF, DICOM. The following constants are defined in j2k.h:

- AW_J2K_FORMAT_TIF
- AW_J2K_FORMAT_PPM
- AW_J2K_FORMAT_PGX
- AW_J2K_FORMAT_BMP
- AW_J2K_FORMAT_TGA
- AW_J2K_FORMAT_JPG
- AW_J2K_FORMAT_J2K
- AW_J2K_FORMAT_DCM
- AW_J2K_FORMAT_DCMJ2K
- AW_J2K_FORMAT_DCMJPG

TIFF images that incorporate runlength coding and LZW compression are supported. DICOM file format is supported in both uncompressed raw, JPEG compressed format, and JPEG2000 compressed format (the last three constants allow for differentiation between the data formats within the DICOM file). Both 8 and 12 bit DICOM JPEG images are supported.

5.1.8 aw_j2k_set_input_image_file_raw

Description

Reads a raw image into the aw_j2k_object. For images with more than 8 bits per channel, you can specify the endianness of the raw image using the function aw_j2k_set_input_raw_endianness on page 53.

C Function Prototype

```
aw_j2k_set_input_image_file_raw(
   aw_j2k_object *j2k_object,
   char *filename,
   unsigned long int rows,
   unsigned long int cols,
   unsigned long int nChannels,
   unsigned long int bpp,
   BOOL bInterleaved)
```

ActiveX Visual Basic Function Prototype

```
set_input_image_file_raw(
   filename as String,
   rows as long,
   cols as long,
   nChannels as long,
   bpp as long,
   bInterleaved as Boolean
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object where the input image will be stored.

filename — Name of the image file.

rows — The number of rows in the full input image. The maximum allowable value is $2^{32} - 1$.

columns — The number of columns in the full input image. The maximum allowable value is $2^{32} - 1$.

nChannels — number of color channels in the image.

`bpp` — depth of pixel in bits, (sum of all channels). Supported pixel depth: up to 16 bits per channel, signed or unsigned. All channels are assumed to have the same bit depth.

`bInterleaved` — This parameter should be set to true if the input image is byte interleaved by channel (ex – R,G,B, R,G,B). Set it to false, if all the bytes from the first channels are in a continuous block, followed immediately by the bytes from the second channel as a continuous block, and so on.

5.1.9 aw_j2k_set_input_raw_information

Description

Specifies to the library the dimensions of the input image without providing the input image itself. The input image may be provided in a later operation via a call to aw_j2k_set_input_image_raw_channel (see page 41).

The rows and columns provided as arguments to this function describe the size of the full image. The function aw_j2k_set_input_raw_channel_subsampling can be used to indicate channels that are subsampled relative to this full size (see page 50).

The bitdepth argument to this function is an aggregate — the sum of the bitdepths of all the channels of the image. The bitdepth of each channel can be specified separately by using aw_j2k_set_input_raw_channel_bpp (see page 52).

C Function Prototype

```
aw_j2k_set_input_raw_information(
   aw_j2k_object *j2k_object,
   unsigned long int rows,
   unsigned long int cols,
   unsigned long int nChannels,
   long int bpp,
   BOOL bInterleaved
)
```

ActiveX Visual Basic Function Prototype

```
set_input_raw_information(
   rows as long,
   cols as long,
   nChannels as long,
   bpp as long,
   bInterleaved as Boolean
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`rows` — The number of rows in the full image.

`cols` — The number of columns in the full image.

`nChannels` — The number of channels in the full image.

`bpp` — The total bitdepth of the image. See page 52 on how to set the bitdepth of each channel individually, using the `aw_j2k_set_input_raw_channel_bpp` function.

`bInterleaved` — This parameter should be set to true if the input image is byte interleaved by channel (ex – R,G,B, R,G,B). Set it to false, if all the bytes from the first channels are in a continuous block, followed immediately by the bytes from the second channel as a continuous block, and so on.

5.1.10 aw_j2k_set_input_raw_channel_bpp

Description

Specifies to the library the bitdepth of a single channel in the image, as well as whether the data in the channel is signed or unsigned.

C Function Prototype

```
aw_j2k_set_input_raw_channel_bpp(
   aw_j2k_object *j2k_object,
   unsigned long int channel,
   long int bpp
)
```

ActiveX Visual Basic Function Prototype

```
set_input_raw_channel_bpp(
   channel as long,
   bpp as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`channel` — The channel for which the bitdepth is provided.

`bpp` — The bitdepth of the channel. The absolute value of this variable is the actual bitdepth. If the value is positive, the data is unsigned; if negative, it is signed.

5.1.11 aw_j2k_set_input_raw_endianness

Description

Sets the endianness of the input raw image data. This controls how images with more than 8 bits of data per color component are read — i. e. which of the two bytes that comprise each sample is the most significant byte and which the least. Three choices are accepted: big endian (native to Motorola and Sparc processors); little endian (native to Intel processors); and the local machine endianness.

This function must be called before aw_j2k_set_input_image_raw (see page 38) or aw_j2k_set_input_image_file_raw (see page 48) to specify the endianness of an input raw image.

C Function Prototype

```
aw_j2k_set_input_raw_endianness(
   aw_j2k_object *j2k_object,
   unsigned long int fEndianness
)
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`fEndianness` — The endianness of the raw data. The following constants are defined in j2k.h:

- `AW_J2K_ENDIAN_LOCAL_MACHINE`
- `AW_J2K_ENDIAN_SMALL`
- `AW_J2K_ENDIAN_BIG`

5.1.12 aw_j2k_set_input_raw_channel_subsampling

Description

Specifies to the library the subsampling of a single channel in the image relative to the size of the image as a whole.

C Function Prototype

```
aw_j2k_set_input_raw_channel_subsampling(
   aw_j2k_object *j2k_object,
   unsigned long int channel,
   unsigned long int X_subsampling,
   unsigned long int Y_subsampling
)
```

ActiveX Visual Basic Function Prototype

```
set_input_raw_channel_bpp(
   channel as long,
   X_subsampling as long,
   Y_subsampling as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`channel` — The channel for which the bitdepth is provided.

`X_subsampling` — The horizontal subsampling of this channel relative to the original image.

`Y_subsampling` — The vertical subsampling of this channel relative to the original image.

5.1.13 aw_j2k_set_input_dcm_frame_index

Description

DICOM format supports placement of several images into a single file using the multiframe syntax. This parameter specifies to the library the desired single frame number to extract from a multiframe DICOM file. If this argument is omitted, the default value depends on the output format: if the input and output formats are DICOM, the default value is -1, which means that all the frames in the file will be processed. If the output format is not DICOM, the default value is 0, which refers to the first frame. If the frame index value specified is greater than the number of frames available in the input file, an error is returned.

C Function Prototype

```
aw_j2k_set_input_dcm_frame_index(
   aw_j2k_object *j2k_object,
   int frame_index,
)
```

ActiveX Visual Basic Function Prototype

```
set_input_input_dcm_frame_index(
   frame_index as long,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`frame_index` — The desired frame number.

5.2 Input Image Information Functions

The following functions return information about the input image. These functions can be called at any time after calling one of the `aw_j2k_set_input_image...()` functions.

5.2.1 `aw_j2k_get_input_image_info`

Description

Returns the dimensions, bitdepth and number of channels in the image.

The bitdepth returned by this function is the aggregate bitdepth for all of the channels in the image. The bitdepth of an individual channel, as well as whether the data in that channel is signed or unsigned, can be determined using `aw_j2k_get_input_channel_bpp` (see page 57).

C Function Prototype

```
aw_j2k_get_input_image_info(
   aw_j2k_object *j2k_object,
   unsigned long int *cols,
   unsigned long int *rows,
   unsigned long int *bpp,
   unsigned long int *nChannels)
```

ActiveX Visual Basic Function Prototype

```
get_input_image_info(
   cols as long,
   rows as long,
   bpp as long,
   nChannels as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`cols` — The number of columns in the image.

`rows` — The number of rows in the image.

`bpp` — depth of pixels in bits (sum of all channels).

`nChannels` — number of color channels in the image.

5.2.2 aw_j2k_get_input_channel_bpp

Description

Returns the bitdepth of a channel in the image, as well as whether the data in the channel is signed or unsigned.

C Function Prototype

```
aw_j2k_get_input_channel_bpp(
   aw_j2k_object *j2k_object,
   unsigned long int channel,
   long int *bpp
)
```

ActiveX Visual Basic Function Prototype

```
get_input_channel_bpp(
   channel as long,
   bpp as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`channel` — The channel for which the bitdepth will be returned.

`bpp` — The bit depth of pixels in bits. The absolute value is the actual bitdepth. If this number is positive, the data is unsigned. If it is negative, the data is signed.

5.2.3 aw_j2k_get_input_j2k_channel_bpp

Description

Returns the bitdepth of a channel in the JPEG2000 image, as well as whether the data in the channel is signed or unsigned.

This function is deprecated. Use aw_j2k_get_input_channel_bpp (see page 57).

C Function Prototype

```
aw_j2k_get_input_j2k_channel_bpp(
   aw_j2k_object *j2k_object,
   unsigned long int channel,
   long int *bpp
)
```

ActiveX Visual Basic Function Prototype

```
get_input_j2k_channel_bpp(
   channel as long,
   bpp as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`channel` — The channel for which the bitdepth will be returned.

`bpp` — The bit depth of pixels in bits. If this number is positive, the data is unsigned. If it is negative, the data is signed. The absolute value is the actual bitdepth.

5.2.4 aw_j2k_get_input_j2k_channel_subsampling

Description

Returns the subsampling, relative to the reference grid, of a channel in the JPEG2000 image.

C Function Prototype

```
aw_j2k_get_input_j2k_channel_subsampling(
   aw_j2k_object *j2k_object,
   unsigned long int channel,
   unsigned long int *Y_subsampling,
   unsigned long int *X_subsampling
)
```

ActiveX Visual Basic Function Prototype

```
get_input_j2k_channel_subsampling(
   channel as long,
   Y_subsampling as long,
   X_subsampling as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`channel` — The channel for which the subsampling will be returned.

`X_subsampling` — The horizontal subsampling of the channel.

`Y_subsampling` — The vertical subsampling of the channel.

5.2.5 aw_j2k_get_input_j2k_num_comments

Description

Returns the number of comments in a JPEG2000 input image.

C Function Prototype

```
aw_j2k_get_j2k_input_j2k_num_comments(
   aw_j2k_object *j2k_object,
   int *n_comments)
```

ActiveX Visual Basic Function Prototype

```
get_j2k_input_j2k_num_comments(
   n_comments as integer
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`n_comments` — pointer to return the number of comments in the given JPEG2000 image.

5.2.6 aw_j2k_get_input_j2k_comments

Description

Returns a comment string in the input image.

C Function Prototype

```
aw_j2k_get_j2k_input_j2k_comments(
   aw_j2k_object *j2k_object, int comment_index,
   unsigned char **comment_data,
   unsigned long int *length,
   unsigned long int *type,
   unsigned long int *location)
```

ActiveX Visual Basic Function Prototype

```
get_input_j2k_comments(
   comment_index as integer,
   string_data as Variant,
   binary_data as Variant,
   type as enumCommentType,
   location as enumCommentLocation,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`comment_index` — index of the comment to be returned. Set it to 0 to get the first comment in the image. Set it to 1 to get the second comment, and so on. If there is no comment corresponding to the given index, the function returns `AW_J2K_ERROR_INVALID_PARAMETER`.

`comment_data` — Pointer to return the comment data. If the pointer points to NULL, the library will allocate a buffer to hold the comment data, which should be freed using `aw_j2k_free()` (see page 26). If the pointer is not NULL, the library will assume it points to a buffer whose length is specified by the `length` argument, and attempt to use it. If the buffer is too short, an error will be issued. See the discussion on page 23 for more details.

`length` — the pointer to return the length of the comment data in number of bytes

`type` — JPEG2000 allows two types of comments. This pointer return the type of comment, defined by one of the following constants:

- AW_J2K_CO_BINARY
- AW_J2K_CO_ISO_8859

For AW_J2K_CO_BINARY the comment is a binary stream of data. For AW_J2K_CO_ISO_8859 the comment should be interpreted as being text created using IS 8859-15:1999 Latin values.

location — JPEG2000 allows the comments to be included in the main header of a specific tilepart header. This pointer returns the number of the tile whose tilepart header includes the comment. The value AW_J2K_CO_LOCATION_MAIN_HEADER indicates that the comment is located in the main header.

5.2.7 aw_j2k_get_input_j2k_progression_info

Description

Returns the progression order, number of wavelet transform levels, number of quality layers, and number of color channels in a JPEG2000 image. Use only when the input image is a JPEG2000 image.

C Function Prototype

```
aw_j2k_get_input_j2k_progression_info(
   aw_j2k_object *j2k_object,
   unsigned long int *order,
   unsigned long int *progression_levels,
   unsigned long int *transform_levels,
   unsigned long int *layers,
   unsigned long *channels,
   unsigned long int *precincts,
   unsigned long int *prog_available)
```

ActiveX Visual Basic Function Prototype

```
get_input_j2k_progression_info(
   order as long,
   progression_levels as long,
   transform_levels as long,
   layers as long,
   channels as long, precincts as long, prog_available as long,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`order` — Pointer to return the progression order of the input image. The following constants are defined:

- `AW_J2K_PO_LAYER_RESOLUTION_COMPONENT_POSITION` – Ordered in progression of quality
- `AW_J2K_PO_RESOLUTION_LAYER_COMPONENT_POSITION` – Progressive by resolution, with secondary progression by quality
- `AW_J2K_PO_RESOLUTION_POSITION_COMPONENT_LAYER` – Progressive by resolution, with a secondary progression by spatial position

- `AW_J2K_PO_POSITION_COMPONENT_RESOLUTION_LAYER` – Progressive by spatial position
- `AW_J2K_PO_COMPONENT_POSITION_RESOLUTION_LAYER` – Progressive by color channel

`Progression_levels` — number of progression levels. If the progression order is by resolution, for example, it would be identical to transform_levels. If the progression order is by quality, then it would be identical to the number of layers. If the progression order is by color channel, it would be equal to the number of color channels in the image.

`Tranform_levels` — number of wavelet transform levels

`layers` — number of quality layers.

`channels` — number of color channels.

`precints` — not supported at this time.

`prog_available` — indicates how many of the total progression levels are available in a partial codestream. Example: for an image with a total of 5 progression levels, this parameter will return 1 when all progression levels are available. The parameter will return 2 when 4 progression levels are available, 3 when 3 progression levels are available, 4 when 2 progression levels are available, and 5 when 1 progression level is available.

5.2.8 aw_j2k_get_input_j2k_progression_byte_count

Description

Returns the number of bytes needed to decode a certain progression level. Use only when the input image is a JPEG2000 image.

This function is analogous to aw_j2k_set_input_j2k_progression_level (see page 75) in that the latter function specifies to the library what portion of a JPEG-2000 codestream to decode, while this function returns the number of bytes from the codestream that are needed for such an operation.

C Function Prototype

```
aw_j2k_get_input_j2k_progression_byte_count(
   aw_j2k_object *j2k_object,
   int progression_level,
   unsigned long int *progression_bytes
)
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

progression_level — The progression level. This is the same value used in aw_j2k_set_input_j2k_progression_level (see page 75).

Legal values: from 1 (meaning full decompression) to the total number of levels in the image (meaning least decompression). The total number of levels in the image can be determined using aw_j2k_get_input_j2k_progression_info (see page 63).

progression_bytes — Pointer to the progression bytes, the number of bytes needed to decode the progression level specified by the progression_level parameter.

Example Code

This sample code will create an array of progression block sizes for an image already loaded into the library object. These sizes are the number of bytes needed to decompress to a given progression level.

```
/* determine the number of progression levels available in the file */
r = aw_j2k_get_input_j2k_progression_info(
        pj2k_object,
        &progression_order,
        &max_progression_levels,
        &resolution_levels,
```

```
        &quality_levels,
        &color_channels,
        &precincts,
        &progression_levels_available
    );

/* allocate an array for the byte counts */
unsigned long int *byte_counts =
  malloc(sizeof (unsigned long int) * max_progression_levels);

/* read the byte count for each level */
for (level = 1; level <= max_progression_levels; level++) {
  aw_j2k_get_input_j2k_progression_byte_count(
        pj2k_object,
        level,
        &byte_counts[level - 1]
    );
}
```

After making these function calls, `byte_counts[0]` stores the number of bytes that are needed for the full decompression, while `byte_counts[1]` stores the number of bytes (from the beginning of the file) that are needed for all but the last progression level. Trucating after the first `byte_counts[1]` bytes of the image would result in an image at the 2^{nd} highest progression level.

5.2.9 aw_j2k_get_input_j2k_tile_info

Description

Returns the tile sizes and image and tile offsets from a JPEG2000 image. Use only when the input image is a JPEG2000 image.

C Function Prototype

```
aw_j2k_get_input_j2k_tile_info(
   aw_j2k_object *j2k_object,
   int *tile_offset_x,
   int *tile_offset_y,
   int *tile_cols,
   int *tile_rows,
   int *image_offset_x,
   int *image_offset_y)
```

ActiveX Visual Basic Function Prototype

```
get_input_j2k_tile_info(
   tile_offset_x as long,
   tile_offset_y as long,
   tile_cols as long,
   tile_rows as long,
   image_offset_x as long,
   image_offset_y as long,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_offset_x` — pointers to return tile offset in x-direction

`tile_offset_y` — pointers to return tile offset in y-direction

`tile_cols` — Pointer to return number of columns in each tile.

`tile_rows` — Pointer to return number of rows in each tile.

`image_offset_x` — pointers to return image offset in x-direction

`image_offset_y` — pointers to return image offset in y-direction

5.2.10 aw_j2k_get_input_j2k_selected_tile_geometry

Description

Returns the size and offset from the origin of a tile. Use only when the input image is a JPEG2000 image.

C Function Prototype

```
aw_j2k_get_input_j2k_selected_tile_geometry(
   aw_j2k_object *j2k_object,
   unsigned long int tileindex,
   unsigned long int *tile_cols,
   unsigned long int *tile_rows,
   unsigned long int *tile_offset_x,
   unsigned long int *tile_offset_y
);
```

ActiveX Visual Basic Function Prototype

```
get_input_j2k_selected_tile_geometry(
   tileindex as long,
   tile_cols as long,
   tile_rows as long,
   tile_offset_x as long,
   tile_offset_y as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tileindex` — Selects the tile for which the geometry information will be returned.

`tile_cols` — Pointer to return number of columns in this tile.

`tile_rows` — Pointer to return number of rows in this tile.

`tile_offset_x` — pointers to return this tile's offset in x-direction

`tile_offset_y` — pointers to return this tile's offset in y-direction

5.2.11 aw_j2k_get_input_j2k_component_registration

Description

Returns the component registration offsets for the specified component from a JPEG-2000 image. The component registration offset specifies by what fraction each of the component's pixels is offset. This allows for the correct display when overlayed components are subsampled with respect to each other.

The offsets returned are in units of 1/65536 of the pixel spacing. E.g. a horizontal offset value of 32768 means that each pixel should be displayed 1/2 a pixel width to the right.

The function aw_j2k_set_output_j2k_component_registration (see page 160) can be used to set the component registration offsets for a JPEG2000 output image.

C Function Prototype

```
aw_j2k_get_input_j2k_component_registration(
    aw_j2k_object *j2k_object,
    WORD component,
    WORD *Xreg,
    WORD *Yreg
)
```

ActiveX Visual Basic Function Prototype

```
get_input_j2k_component_registration(
    component as long,
    Xreg as long,
    Yreg as long,
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

component — The component for which the offsets will be returned.

Xreg — pointer to return the horizontal registration offset.

Yreg — pointer to return the vertical registration offset.

5.2.12 aw_j2k_get_input_j2k_bytes_read

Description

This function returns the number of bytes actually used from a JPEG 2000 image. This number is equal to the size of the entire image, if complete decompression was performed, or to a smaller number, if partial decompression was performed.

Partial decompression is achieved using any combination of `aw_j2k_set_input_j2k_progression_level` (as described on page 75), `aw_j2k_set_input_j2k_resolution_level` (as described on page 77), `aw_j2k_set_input_j2k_quality_level` (as described on page 78), `aw_j2k_set_input_j2k_color_level` (as described on page 79), `aw_j2k_set_input_j2k_selected_channel` (as described on page 81), or `aw_j2k_set_input_j2k_selected_tile` (as described on page 82),

C Function Prototype

```
aw_j2k_get_input_j2k_bytes_read(
   aw_j2k_object *j2k_object,
   DWORD *bytes
)
```

ActiveX Visual Basic Function Prototype

```
get_input_j2k_bytes_read(
   bytes as long,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`bytes` — pointer to return the byte count.

5.2.13 aw_j2k_get_input_j2k_decoder_status

Description

Allows retrieval of the JPEG2000 decoder's status after decoding a JPEG2000 image (either to another format, or via reformatting to a new JPEG2000 image). The decoder's status will indicate whether the input JPEG2000 image was fully decoded or whether parts of it were not decodable.

The JPEG2000 decoder, when operating in the error-resilient mode (see aw_j2k_set_input_j2k_error_handling, described on page 83, to set the error-resilience mode), will attempt to overcome errors in the JPEG2000 codestream and decode as much as possible from the stream. The decoder status will indicate whether any errors were encoutered.

This function will return meaningful information only if it is called after a call to one of the aw_j2k_get_output_image family of functions, which are described on pages 96 through 104.

C Function Prototype

```
aw_j2k_get_input_j2k_decoder_status(
   aw_j2k_object *j2k_object,
   unsigned long int *status
)
```

ActiveX Visual Basic Function Prototype

```
get_input_j2k_decoder_status(
   status as long
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

status — The JPEG2000 decoder status. This can have one of the following values:

- AW_J2K_DECODER_FULL_DECODE — The image was fully decoded (that is, decoded to the extent required by calls to aw_j2k_set_input_j2k_progression_level, as described on page 75, aw_j2k_set_input_j2k_resolution_level, as described on page 77, aw_j2k_set_input_j2k_quality_level, as described on page 78, aw_j2k_set_input_j2k_color_level, as described on page 79, aw_j2k_set_input_j2k_region_level, as described on page 80, aw_j2k_set_input_j2k_selected_channel, as described on

page 81, or `aw_j2k_set_input_j2k_selected_tile`, as described on page 82).

- `AW_J2K_DECODER_PARTIAL_DECODE_TRUNCATION` — The image was only partially decoded because some data is missing from its end.
- `AW_J2K_DECODER_PARTIAL_DECODE_CORRUPTION` — The image was only partially decoded because corruption was detected.

5.3 JPEG2000 Decompression and Input Reformatting Functions

The functions described in this section are used to set parameters that control the decoding of the JPEG2000 image. The JPEG2000 standard is unique in that a compressed image actually represents a class of possible decoded images. For this reason, to fully support JPEG2000 files as input images, a number of JPEG2000 input parameters and options must be accessible to the user. The functions described in this section provide this capability. For each function, we describe its functionality, the default value, and the allowable range of parameter values.

Usage:
The functions described in this section are used for the following operations:

1. Decompression of JPEG2000 Image.
2. Reformatting of a JPEG2000 Image. These input functions control the decoding process. For example, if you want to extract a grayscale image at a lower resolution from a full size color image, you would set the `color_level` parameter using the function `aw_j2k_set_input_j2k_color_level` and set the `resolution_level` parameter using the function `aw_j2k_set_input_j2k_resolution_level`.

5.3.1 aw_j2k_set_input_j2k_rvalue

! **The use of this function is deprecated. The dequantization offset default value is set to 0.5 and cannot be modified, at this time.**

Description

Sets the dequantization offset value used during dequantization of the wavelet coefficients.

C Function Prototype

```
aw_j2k_set_input_j2k_rvalue(
   aw_j2k_object *j2k_object,
   float rvalue)
```

ActiveX Visual Basic Function Prototype

```
set_input_j2k_rvalue(rvalue as float) as integer
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`rvalue` — The dequantization offset value. During compression, the wavelet-transformed coeffecients are sorted into bins in a process known as quantization. During decompression, the reverse process, dequantization, attempts to restore the coefficient values from the bin number assigned to each one of them. The dequantization offset value specifies the relative location within each bin of the value assigned to the coefficients in that bin. For example, for a bin of coeffients with values ranging from 5 to 10, a value of 0 specifies that all coefficients in the bin be assigned a value of 5, while a value of 1 specifies that all coefficients in the bin be assigned a value of 10.

The default value is 0.44, which often results in the best decompressed image quality. It has been shown that 0.44 is optimal for coefficients with a Laplacian distribution.

Useful values: 0.0 to 1.0. Values outside this range can be used but may result in odd or artistic looking images.

5.3.2 aw_j2k_set_input_j2k_progression_level

Description

Sets the number of progression levels to be decoded from the native progression order of the image.

JPEG2000 images can be organized progressively by:

- Resolution, from a thumbnail to a full size image.
- Quality, from low quality to best quality or lossless.
- Color channel, from grayscale to full 3 channel color.

This function controls the number of progression levels to decode from a JPEG-2000 image which may be organized in any of the 3 progression orders described above.

The native progression order of the image and the number of progression levels in the image can be obtained using the `aw_j2k_get_input_j2k_progression_info` function (see page 63).

If the image is organized progressively, by resolution, this function has the same effect as `aw_j2k_set_input_j2k_resolution_level`. Similarly, if the image is organized progressively by quality, this function has the same effect as `aw_j2k_set_input_j2k_quality_level`. This function is useful for cases where a JPEG2000 image is decoded from a stream, and the user desires to progressively display the image as it arrives. The number of progression levels available in the (possibly) incomplete stream can be determined by `aw_j2k_get_input_j2k_progression_info`, and then this function can be used to partially decompress the image.

C Function Prototype

```
aw_j2k_set_input_j2k_progression_level(
   aw_j2k_object *j2k_object,
   int progression_level)
```

ActiveX Visual Basic Function Prototype

```
set_input_j2k_progression_level(
   progression_level as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`progression_level` — The number of progression levels to decode from the native progression order of the image.

Default value: all the available progression levels.

Legal values: from 1 (meaning full decompression) to the total number of levels in the image (meaning least decompression).

5.3.3 aw_j2k_set_input_j2k_resolution_level

Description

Sets the number of resolution levels to be decoded from a JPEG2000 input image.

C Function Prototype

```
aw_j2k_set_input_j2k_resolution_level(
   aw_j2k_object *j2k_object,
   int resolution_level,
   int full_xform_flag)
```

ActiveX Visual Basic Function Prototype

```
set_input_j2k_resolution_level(
   resolution_level as long,
   full_xform_flag as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`resolution_level` — The number of resolution levels to decode.

This function does not require that the progression order of the JPEG2000 image be organized by resolution. If the input image is organized progressively by resolution, using this function is equivalent to using the function `aw_j2k_set_input_j2k_progression_level`. However, for images that are not organized progressively by resolution, this function will instruct the decoder to parse the file and only decode those parts necessary for the requested resolution levels.

Default value: all the available resolution levels. Legal values are from 1 (meaning full decompression) to the total number of levels in the image (meaning least decompression).

The number of resolution levels in the image can be obtained using the `aw_j2k_get_input_j2k_progression_info` function (see page 63).

`full_xform_flag` Indicates whether a full transform should be carried out when not all the resolution levels are decoded from the file. Setting this flag to true will execute a full transform, which will present a full-size image interpolated from the low resolution data that was decoded from the file. Setting this flag to false will produce a low-resolution image.

5.3.4 aw_j2k_set_input_j2k_quality_level

Description

Sets the number of quality levels to be decoded from a JPEG2000 input image.

C Function Prototype

```
aw_j2k_set_input_j2k_quality_level(
   aw_j2k_object *j2k_object,
   int quality_level)
```

ActiveX Visual Basic Function Prototype

```
set_input_j2k_quality_level(
   quality_level as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`quality_level` – The number of quality levels to decode. Quality levels correspond to encoded bitplanes of the quantized wavelet coefficients. Decoding more quality levels improves the decoded image quality.

This function does not require that the progression order of the JPEG2000 image be organized by quality. If the input image is organized progressively by quality, using this function is equivalent to using the function `aw_j2k_set_input_j2k_progression_level` (see page 75). However, for images that are not organized progressively by quality, this function will instruct the decoder to parse the file and only decode those parts necessary for the requested quality levels.

Default value: all available quality levels. Legal values are from 1 (meaning full decompression) to the total number of levels in the image (meaning least decompression). The number of quality levels in the image can be obtained using the `aw_j2k_get_input_j2k_progression_info` function (see page 63).

5.3.5 aw_j2k_set_input_j2k_color_level

Description

Sets the number of color levels (channels) to be decoded.

C Function Prototype

```
aw_j2k_set_input_j2k_color_level(
   aw_j2k_object *j2k_object,
   int color_channels,
   int color_xform_flag)
```

ActiveX Visual Basic Function Prototype

```
set_input_j2k_color_level(
   color_channels as long,
   color_xform_flag as long,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`color_channels` — The number of color channels to decode.

Default value: all available channels. Legal values are from 1 (meaning full decompression) to the total number of levels in the image (meaning least decompression).

The number of channels in the image can be obtained using the aw_j2k_get_input_j2k_progression_info function (see page 63).

`color_xform_flag` — Indicates whether a color transform should be performed. By default, a color transform will be performed if the original JPEG2000 image indicates that it should be (see `aw_j2k_set_output_color_xform`, page 112). This flag can indicate that the default action should be taken (use `AW_J2K_CT_DEFAULT`), that the color transform should not be applied (use `AW_J2K_CT_SKIP`), or that the color transform should be applied (use `AW_J2K_CT_PERFORM`).

5.3.6 aw_j2k_set_input_j2k_region_level

Description

Defines a rectangular subregion to be decoded from a JPEG2000 image. If the JPEG-2000 image was created using non-default precinct sizes (by calling `aw_j2k_set_output_j2k_channel_precinct_size`, described on page 149), the library will decode only those precincts that intersect with the defined region. This effectively means that an area larger than the defined area may be decoded, although only the defined region is fully decoded.

If default precinct sizes were used, the library will have no choice but to decode the entire image.

Note that the precinct size is an output image parameter, i.e. it must be defined when the JPEG2000 image is created.

C Function Prototype

```
aw_j2k_set_input_j2k_region_level(
   aw_j2k_object *j2k_object,
   unsigned long int x0,
   unsigned long int y0,
   unsigned long int x1,
   unsigned long int y1
)
```

ActiveX Visual Basic Function Prototype

```
set_input_j2k_region_level(
   x0 as long,
   y0 as long,
   x1 as long,
   y1 as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`x0` — The left coordinate of the region.

`x1` — The right coordinate of the region.

`y0` — The top coordinate of the region.

`y1` — The bottom coordinate of the region.

5.3.7 aw_j2k_set_input_j2k_selected_channel

Description

Selects a specific color channel from a JPEG2000 image with multiple channels. After calling this function, any further processing will only be applied to the selected channel in the image. This call is valid only if the input image is a J2K image.

C Function Prototype

```
aw_j2k_set_input_j2k_selected_channel(
   aw_j2k_object *j2k_object,
   unsigned long int channel
)
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`channel` — Index of the channel to decompress. For an RGB image, channel 0 is the red channel, channel 1 is the green channel, and channel 2 is the blue channel. Note that if color rotation was applied to an RGB image before compression (using a call to `aw_j2k_set_output_j2k_color_xform`, page 112), then channel 0 will contain luminance information and channels 1 and 2 will contain chrominance information. The default behavior, which is to decode all the channels in an image, can be obtained by setting this parameter to `AW_J2K_SELECT_ALL_CHANNELS`.

5.3.8 aw_j2k_set_input_j2k_selected_tile

Description

Selects a specific tile from a JPEG2000 image with multiple tiles. After calling this function, any further processing will only be applied to the selected tile in the image. This call is valid only if the input image is a J2K image.

C Function Prototype

```
aw_j2k_set_input_j2k_selected_tile(
   aw_j2k_object *j2k_object,
   int tile
)
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile` — Index of the tile to decompress. The index of each tile is arranged in a raster scan (top-left to bottom-right). For example, in a J2K image with 10 tiles across and 10 tiles down, the index of the tile at top-left is 0. And the index of the tile at the top-right is 9. The tile index at bottom left is 90, and so on. Set this parameter to `AW_J2K_SELECT_ALL_TILES` to select all the tiles in the image.

5.3.9 aw_j2k_set_input_j2k_error_handling

Description

Selects the level of error-resiliency offered by the JPEG2000 decoder. The JPEG2000 decoder may operate in a strict mode, in which any errors in the bitstream will trigger an error and stop image decoding, or in a resilient mode, in which the decoder will try to recover from errors and decode those parts of the image which are not corrupt.

It is possible to query the decoder, when it is operating in the resilient mode, as to whether it encountered errors and recovered from them. This is done via a call to aw_j2k_get_input_j2k_decoder_status, described on page 71.

C Function Prototype

```
aw_j2k_set_input_j2k_error_handling(
   aw_j2k_object *j2k_object,
   int mode
)
```

ActiveX Visual Basic Function Prototype

```
set_input_j2k_error_handling(
   mode as long
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

mode — The selected error handling mode. The allowable values for this option are:

- AW_J2K_ERROR_RESILIENT_MODE – to select the error-resilient mode of operation.
- AW_J2K_ERROR_STRICT_MODE – to select the strict mode of operation. This is the default mode.

5.3.10 aw_j2k_input_reset_options

Description

Resets all the input options described above to their defaults values.

C Function Prototype

```
aw_j2k_input_reset_options(
   aw_j2k_object *j2k_object)
```

ActiveX Visual Basic Function Prototype

```
input_reset_options() as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

5.4 JP2 Input Information Functions

The JPEG2000 standard allows the user to add unstructured data, in the format of a comment, into the J2K codestream (using `aw_j2k_set_output_j2k_add_comment` described on page 116). The JPEG 2000 file format (also known as JP2) expands on this capability, by allowing the embedding of structured data in various types of boxes.

The functions in this section enable the detection and extraction of several types of metadata boxes from a JP2 image. The following types of metadata boxes can be extracted:

- Intellectual Property Box, which contains intellectual property information. The format specification of the data is reserved for ISO. It will be up to the user to interpret the data.
- XML Box, which contains text data in XML format.
- UUID Box, which contains a 16-byte UUID followed by optional data. The format of the UUID is specified by ISO/IEC 11578:1996.
- UUID Information BOX, a superbox which contains two different sub boxes: a UUID List box and a URL box. The URL points to a location where information regarding the UUIDs in the list may be found.

The JP2 format also adds support for including color space information with the image. The color space may be specified in one of two ways: by enumeration, or by inclusion of a Restricted ICC Profile. The SDK allows the extraction of either type of specification from a JP2 image.

To add metadata to a JP2 image, see the function in the JP2 Output Functions section, on page 179.

5.4.1 aw_j2k_get_input_jp2_num_metadata_boxes

Description

Returns the number of metadata boxes of a given type in a JP2 image. Metadata boxes include boxes that contain intellectual property information, XML, UUIDs, and UUID information. To retrieve the actual contents of a metabox use aw_j2k_get_input_jp2_metadata_box, described on page 87.

C Function Prototype

```
aw_j2k_get_input_jp2_num_metadata_boxes(
   aw_j2k_object *j2k_object,
   unsigned int type,
   int *num_metadata_boxes
)
```

ActiveX Visual Basic Function Prototype

```
get_input_jp2_num_metadata_boxes(
   type as enumMetadataType
   num_metadata_boxes as long
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

type — The type of the metadata box, which can be one of the following:

- AW_JP2_MDTYPE_JP2I — for Intellectual Property boxes,
- AW_JP2_MDTYPE_XML — for XML boxes,
- AW_JP2_MDTYPE_UUID — for UUID boxes, and
- AW_JP2_MDTYPE_UUIDINFO — for UUID Information boxes.

num_metadata_box — pointer to return the number of metadata boxes of the given type present in the input JP2 file.

5.4.2 aw_j2k_get_input_jp2_metadata_box

Description

This function can be used to extract single metadata boxes of a given type from an input JP2 image. The count of the number of metadata boxes of this kind can be found using aw_j2k_get_input_jp2_num_metadata_boxes, described on page 86.

C Function Prototype

```
aw_j2k_get_input_jp2_metadata_box(
   aw_j2k_object *j2k_object,
   unsigned long int type,
   unsigned long int index,
   unsigned char **metadata,
   unsigned long int *metadata_length,
   unsigned long int **uuid,
   unsigned long int *uuid_length
)
```

ActiveX Visual Basic Function Prototype

```
get_input_jp2_metadata_box(
   type as enumMetadataType,
   index as long,
   metadata as Variant(array of bytes),
   uuid as Variant(array of bytes)
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

type — The type of the metadata box, which can be one of the following:

- AW_JP2_MDTYPE_JP2I — for Intellectual Property boxes,
- AW_JP2_MDTYPE_XML — for XML boxes,
- AW_JP2_MDTYPE_UUID — for UUID boxes, and
- AW_JP2_MDTYPE_UUIDINFO — for UUID Information boxes.

index — The index of the selected metadata box from which data will be retrieved.

`metadata` — Pointer to receive the metadata. If it points to NULL, the library will allocate a buffer to hold this data, which should be freed using `aw_j2k_free` (see page 26). If the pointer is not NULL, the library will assume that it points to buffer whose length is specified by the `metadata_length` argument. If the specified length is too short, the function will return `AW_J2K_ERROR_BUFFER_TOO_SHORT` and `metadata_Length` will be set to the required length.

`metadata_Length` — pointer to return the length of the metadata in number of bytes.

`UUID` — pointer to return UUID data. If it points to NULL, the library will allocate a buffer to hold UUID data, which should be freed using `aw_j2k_free` (see page 26). If the pointer is not NULL, the library will assume that it points to buffer whose length is specified by `uuid_length`. If the specified length is less than 16, the function will return `AW_J2K_ERROR_BUFFER_TOO_SHORT` and `uuid_length` will be set to the size of the required buffer. This field is used only if `type` is set to `AW_JP2_MDTYPE_UUID`. If it is not, this field will be ignored.

`uuid_length` — pointer to return the length of the UUID. The length will be set to 16. This field is used only if `type` is set to `AW_JP2_MDTYPE_UUID`. If it is not, this field will be ignored.

5.4.3 aw_j2k_get_input_jp2_UUIDInfo_box

Description

This function can be used to extract UUID Information data from an input JP2 image. The function aw_j2k_get_input_jp2_num_metadata_boxes (described on page 86) can be used to determine the number of UUID Information boxes that are present in a JP2 image.

C Function Prototype

```
aw_j2k_get_input_jp2_UUIDInfo_box(
   aw_j2k_object *j2k_object,
   unsigned long int index,
   unsigned char **uuid_data,
   unsigned long int *num_uuid,
   char **url,
   unsigned long int *url_length
)
```

ActiveX Visual Basic Function Prototype

```
get_input_jp2_UUIDInfo_box(
   index as long,
   uuid_data as Variant (array of bytes),
   num_uuid as long,
   url as Variant,
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

index — Index of the selected UUID Information box.

uuid_data — The UUID data buffer. If it initially points to NULL, the library will allocate a buffer to hold the UUID data, which should be freed using aw_j2k_free (described on page 26). If the pointer is not NULL, the library will assume that it points to buffer whose length is 16 times the value of num_uuid. If the length is too short, the function will return AW_J2K_ERROR_BUFFER_TOO_SHORT, and num_uuid will be set to the number of UUIDs present in the box.

num_uuid — Pointer to receive the number of UUIDs returned. Each UUID is 16 bytes long.

`URL` — Pointer to receive the URL. If it points to NULL, the library will allocate a buffer to hold URL, which should be freed using `aw_j2k_free` (described on page 26). If the pointer is not NULL, the library will assume that it points to buffer whose length is specified by `url_length`. If the specified length is less than amount required, the function will return `AW_J2K_ERROR_BUFFER_TOO_SHORT` and `url_length` will be set to size of the required buffer.

`url_length` — pointer to receive length of URL in bytes.

5.4.4 aw_j2k_get_input_jp2_enum_colorspace

Description

This function is used to get the enumurated color space information from the input JP2 image, if any. This function returns AW_J2K_JP2_NO_ENUM_COLORSPACE if the input JP2 image does not contain an enumerated colorspace.

C Function Prototype

```
aw_j2k_get_input_jp2_enum_colorspace(
   aw_j2k_object *j2k_object,
   long int *colorspace
)
```

ActiveX Visual Basic Function Prototype

```
get_input_jp2_enum_colorspace(
   colorspace as enumColorspaceType
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

colorspace — The pointer to return the color space information. Can be one of the following values:

- AW_JP2_CS_GREYSCALE
- AW_JP2_CS_sRGB
- AW_JP2_CS_sYCC

5.4.5 aw_j2k_get_input_jp2_restricted_ICCProfile

Description

This function is used to retrieve the restricted ICC color space profile from an input JP2 image, if any. Returns AW_J2K_JP2_NO_ICCPROFILE if no ICC profile is found in the input JP2 image.

C Function Prototype

```
aw_j2k_get_input_jp2_restricted_ICCProfile(
   aw_j2k_object *j2k_object,
   unsigned char **ICC_profile,
   unsigned long int *length
)
```

ActiveX Visual Basic Function Prototype

```
get_input_jp2_restricted_ICCProfile(
   ICC_profile as Variant (Array of Bytes),
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

ICC_profile — pointer to receive the ICC profile data. If it points to NULL, the library will allocate a buffer to hold the data, which should be freed using aw_j2k_free (described on page 26). If the pointer is not null, the library will assume that it points to buffer whose length is specified by length argument. If the specified length is too short, the function will return AW_J2K_ERROR_BUFFER_TOO_SHORT and length will be set to size of the required buffer.

length — pointer to receive the length of the ICC_profile buffer.

Chapter 6

Output Image Functions

The functions described in this section, as shown in Figure 6.1, operate on the output image of the library. The functions described in this section are organized, based on their functionality, in the following categories:

Image Writing Functions, which are used to create an output image from the object. These functions are described in Section 6.1.

Basic JPEG 2000 Output Functions, which are the basic functions that are used to create a JPEG2000 Output Image. These functions are described in Section 6.2.

Advanced JPEG2000 Compression Functions, which implement the advanced options that are available when creating a JPEG2000 Output Image. These functions are described in Section 6.3.

JPEG2000 Region of Interest Functions, which are used to define a Region of Interest when creating a JPEG2000 Output Image. These functions are described in Section 6.4.

JPEG Output Image Functions, which are used to set the options to create a JPEG Output Image. These functions are described in Section 6.6.

Other Output Image Functions, which are used to set other options when creating an Output Image. These functions are described in Section 6.7.

ActiveX-only Functions, which are only available through the ActiveX interface. These functions are described in Section 6.8.

Multi-component Transformation Functions, which are used to enable wavelet transformation of the image data along the 3^{rd} dimension, effecting volumetric compression. These functions are described in Section 6.9.

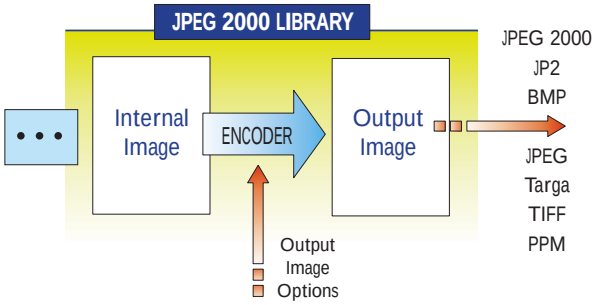

Figure 6.1: The output parameters of the Library

6.1 Image Writing Functions

.

The functions described in this section are used to to create an output image from the object. These functions actually perform all the encoding, decoding, and reformatting that may be necessary to turn the input image into the output image. Accordingly, these functions should be called after the input image and all the input and output parameters are set.

6.1.1 aw_j2k_set_output_type

Description

Sets the output image type.

C Function Prototype

```
aw_j2k_set_output_type(
   aw_j2k_object *j2k_object,
   int image_type
)
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`image_type` — The image type. The following image types are supported: JPEG-2000, JPG, PPM, PGM, PGX, BMP, TGA, TIFF, DICOM. The following constants are define in j2k.h:

- `AW_J2K_FORMAT_TIF`
- `AW_J2K_FORMAT_PPM`
- `AW_J2K_FORMAT_PGX`
- `AW_J2K_FORMAT_BMP`
- `AW_J2K_FORMAT_TGA`
- `AW_J2K_FORMAT_JPG`
- `AW_J2K_FORMAT_J2K`
- `AW_J2K_FORMAT_DCM`
- `AW_J2K_FORMAT_DCMJ2K`
- `AW_J2K_FORMAT_DCMJPG`

Except for JPEG and J2K formats, all the files are expected in uncompressed format (TIFF, for example, allows LZW compression, which is not supported). DICOM file format is supported in both uncompressed raw, JPEG compressed format and J2K compressed format (the last three constants allow for differentiation between the data formats within the DICOM file). Both 8 and 12 bit DICOM JPEG images are supported.

6.1.2 aw_j2k_get_output_image

Description

Returns a new image from the aw_j2k_object to a buffer.

C Function Prototype

```
aw_j2k_get_output_image(
   aw_j2k_object *j2k_object,
   unsigned char **image_buffer,
   unsigned long int *image_buffer_length
)
```

ActiveX Visual Basic Function Prototype

```
get_output_image(
   image_buffer as Variant(array of Bytes)
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

image_buffer — The output image buffer. The image type must be set using the aw_j2k_set_output_type function (described on page 95).

If image_buffer points to NULL, the library will allocate a buffer to hold the image data, which should be freed using aw_j2k_free() (see page 26). If the pointer is not NULL, the library will assume it points to a buffer whose length is specified by the image_buffer_length argument, and attempt to use it. If the buffer is too short, an error will be issued. See the discussion on page 23 for more details.

image_buffer_length — The pointer to return length of the output image.

6.1.3 aw_j2k_get_output_image_type

Description

Returns a new image of a specific type from the object to a buffer. The only difference between this function and aw_j2k_get_output_image (page 96) is that this function allows the user to specify the output image type and retrieve the output image with a single function call.

C Function Prototype

```
aw_j2k_get_output_image_type(
   aw_j2k_object *j2k_object,
   int image_type,
   unsigned char **image_buffer,
   unsigned long int *image_buffer_length
)
```

ActiveX Visual Basic Function Prototype

```
get_output_image_type(
   image_type as enumSupportImageTypes,
   image_buffer as Variant(array of Bytes)
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

image_type — The image type. The following image types are supported: JPEG-2000 codestreams (J2K), JPEG2000 file format (JP2), JPG, PPM, PGM, PGX, BMP, TGA, TIFF, DICOM.

image_buffer — The output image buffer. If image_buffer points to NULL, the library will allocate a buffer to hold the image data, which should be freed using aw_j2k_free() (see page 26). If the pointer is not NULL, the library will assume it points to a buffer whose length is specified by the image_buffer_length argument, and attempt to use it. If the buffer is too short, an error will be issued. See the discussion on page 23 for more details.

image_buffer_length — The pointer to return length of the output image.

6.1.4 aw_j2k_get_output_image_raw aw_j2k_get_output_image_raw_ex

Description

Returns a raw image from the aw_j2k_object to a buffer.

There are two versions to this function. The first, aw_j2k_get_output_image_raw, returns the raw data with each sample (that is, each color sample of each pixel) stored separately, padded to the next byte boundary. For example, an image with 3 color channels and 12-bit data per color channel per pixel will be stored in 2 bytes per sample per pixel for a total of 6 bytes per pixel.

The second version, aw_j2k_get_output_image_raw_ex, allows the caller to specify how to pad the data, by specifying how many bits are allocated per pixel. For the example image above, if the allocated bits parameter is specified as 48 (which is 16 bits per sample per pixel) than the default behavior of the first function is obtained. For the same image, to avoid padding, the caller can specify the allocated bits as 36 (12 bits per sample per pixel), which would cause the data to be packed into 3 bytes per pixel.

To find the stored bit-depth before calling these functions use the aw_j2k_get_input_image_info function, described on page 56.

C Function Prototype

```
aw_j2k_get_output_image_raw(
   aw_j2k_object *j2k_object,
   unsigned char **image_buffer,
   unsigned long int *image_buffer_length,
   unsigned long int *rows,
   unsigned long int *cols,
   unsigned long int *nChannels,
   unsigned long int *bpp_stored,
   BOOL bInterleaved
)
aw_j2k_get_output_image_raw_ex(
   aw_j2k_object *j2k_object,
   unsigned char **image_buffer,
   unsigned long int *image_buffer_length,
   unsigned long int *rows,
   unsigned long int *cols,
   unsigned long int *nChannels,
   long int *bpp_stored,
   long int bpp_allocated,
   BOOL bInterleaved
)
```

ActiveX Visual Basic Function Prototype

```
get_output_image_raw(
   image_buffer as Variant(array of Bytes),
   rows as long,
   cols as long,
   nChannels as long,
   bpp_stored as long,
   bInterleaved as Boolean,
) as long

get_output_image_raw_ex(
   image_buffer as Variant(array of Bytes),
   rows as long,
   cols as long,
   nChannels as long,
   bpp_stored as long,
   bpp_allocated as long,
   bInterleaved as Boolean,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object` where the input image will be stored.

`image_buffer` — The output image buffer. If `image_buffer` points to NULL, the library will allocate a buffer to hold the image data, which should be freed using `aw_j2k_free()` (see page 26). If the pointer is not NULL, the library will assume it points to a buffer whose length is specified by the `image_buffer_length` argument, and attempt to use it. If the buffer is too short, an error will be issued. See the discussion on page 23 for more details.

`image_buffer_length` — The pointer to return the length of the output image buffer.

`rows` — The pointer to return the number of rows in the full input image.

`columns` — The pointer to return the number of columns in the full input image.

`nChannels` — The pointer to return the number of color channels in the image.

`bpp_stored` — The pointer to return the bit depth of the data in the pixels (sum of all channels). All channels are assumed to have the same bit depth. Supported pixel depth: up to 16 bits per channel, signed or unsigned.

`bpp_allocated` — The depth of pixels in bits (sum of all channels). All channels are assumed to have the same bit depth. Supported pixel depth: up to 16 bits per channel, signed or unsigned.

`bInterleaved` — flag indicating whether the channels in the output image should be interleaved.

6.1.5 aw_j2k_get_output_image_file

Description

Writes an image from the aw_j2k_object to a file.

C Function Prototype

```
aw_j2k_get_output_image_file(
   aw_j2k_object *j2k_object,
   char *filename
)
```

ActiveX Visual Basic Function Prototype

```
get_output_image_file(
   filename as String
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

filename — the name of the image file.

Comments

Call aw_j2k_set_output_type (page 95) before calling this function to set the output file type.

6.1.6 aw_j2k_get_output_image_file_type

Description

Writes an image of a specific type from the aw_j2k_object to a file.

C Function Prototype

```
aw_j2k_get_output_image_file_type(
    aw_j2k_object *j2k_object,
    int image_type,
    char *filename
)
```

ActiveX Visual Basic Function Prototype

```
get_output_image_file_type(
   BSTR filename as String,
   image_type as enumSupportImageTypes
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

image_type — The image type. The following image types are supported: JPEG-2000, JPG, PPM, PGM, PGX, BMP, TGA, TIFF, DICOM. The following constants are define in j2k.h:

- AW_J2K_FORMAT_TIF
- AW_J2K_FORMAT_PPM
- AW_J2K_FORMAT_PGX
- AW_J2K_FORMAT_BMP
- AW_J2K_FORMAT_TGA
- AW_J2K_FORMAT_JPG
- AW_J2K_FORMAT_J2K
- AW_J2K_FORMAT_JP2
- AW_J2K_FORMAT_DCM
- AW_J2K_FORMAT_DCMJ2K
- AW_J2K_FORMAT_DCMJPG

Except for JPEG and J2K formats, all the files are expected in uncompressed format (TIFF, for example, allows LZW compression, which is not supported). DICOM file format is supported in both uncompressed raw, JPEG compressed format and J2K compressed format (the last three constants allow for differentiation between the data formats within the DICOM file). Both 8 and 12 bit DICOM JPEG images are supported.

`filename` — the name of the image file.

6.1.7 aw_j2k_get_output_image_file_raw

Description

Writes a RAW image from the aw_j2k_object to a file.

C Function Prototype

```
aw_j2k_get_output_image_file_raw(
   aw_j2k_object *j2k_object,
   char *filename,
   unsigned long int *rows,
   unsigned long int *cols,
   unsigned long int *nChannels,
   unsigned long int *bpp,
   BOOL bInterleaved
)
```

ActiveX Visual Basic Function Prototype

```
get_output_image_file_raw(
   filename as String,
   rows as long,
   cols as long,
   nChannels as long,
   bpp as long,
   bInterleaved as Boolean
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object where the input image will be stored.

filename — The name of the output image file.

rows — The pointer to return the number of rows in the full input image.

columns — The pointer to return the number of columns in the full input image.

nChannels — The pointer to return the number of color channels in the image.

bpp — The pointer to return the depth of pixels in bits (sum of all channels). All channels are assumed to have the same bit depth. Supported pixel depth: up to 16 bits per channel, signed or unsigned.

bInterleaved — flag indicating whether the output image should be interleaved.

6.2 Basic JPEG2000 Compression and Output Reformatting Functions

The functions described in this section are the basic functions used to create a JPEG-2000 Output Image. They are only applicable when the output image type is set to JPEG2000.

Usage:
The functions described in this section are used for the following operations:

1. Compression of JPEG2000 Image. These functions control the main options of a compression operation, set the compression ratio, progression order, etc.

2. Reformatting of a JPEG2000 Image. These output functions set the main options that determine what type of JPEG2000 image will be created as a result of the reformatting operation. For example, to create an image organized progressively by quality, you would set the order parameter using the function `aw_j2k_set_output_j2k_progression_order` (page 113).

6.2.1 aw_j2k_set_output_j2k_bitrate

Description

Sets the output image bitrate in bits per pixel. This function can be used in place of aw_j2k_set_output_j2k_ratio (page 107) and aw_j2k_set_output_j2k_filesize (page 108) to specify the desired output JPEG2000 image size, in bits per pixel.

If more than one of these functions is called, whichever function was called later takes effect.

C Function Prototype

```
aw_j2k_set_output_j2k_bitrate(
   aw_j2k_object *j2k_object,
   float bpp
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_bitrate(
   bpp as float,
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

bpp — Bitrate in bits per pixel for the output image.

Default: bitrate equivalent 10:1 compression ratio.

For lossless compression, set ratio to 0 and select the 5-3 wavelet using aw_j2k_set_output_j2k_xform (see page 123).

6.2.2 aw_j2k_set_output_j2k_ratio

Description

Sets the compression ratio for a JPEG2000 compression process. The compression ratio is defined as the number of bytes in the original image divided by the number of bytes in the output image. This function can be used in place of `aw_j2k_set_output_j2k_bitrate` (page 106) and `aw_j2k_set_output_j2k_filesize` (page 108) to specify the desired output JPEG2000 image size. If more than one of these functions is called, whichever function was called later takes effect.

C Function Prototype

```
aw_j2k_set_output_j2k_ratio(
   aw_j2k_object *j2k_object,
   float ratio
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_ratio(
   ratio as float,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`ratio` — The compression ratio, defined as the number of bytes in the input image over the number of bytes in the compressed image. For input images that are in a compressed image format, such as JPEG and JPEG2000, the compression ratio is based on the total size of the input image after it has been decompressed.

Default value: 10.

For lossless compression, set ratio to 0 and select the 5-3 wavelet using `aw_j2k_set_output_j2k_xform` (see page 123).

6.2.3 aw_j2k_set_output_j2k_filesize

Description

Sets the output file size in bytes. This function can be used in place of aw_j2k_set_output_j2k_bitrate (page 106) and aw_j2k_set_output_j2k_ratio (page 107) to specify the desired output image size. If more than one of these functions is called, whichever function was called later takes effect.

C Function Prototype

```
aw_j2k_set_output_j2k_filesize(
   aw_j2k_object *j2k_object,
   unsigned int cbBytes
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_filesize(
   cbBytes as long
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

cbBytes — The number of bytes in the output image.

Default: filesize equivalent to 10:1 compression ratio.

6.2.4 aw_j2k_set_output_j2k_psnr

Description

Sets the output image quality, measured by peak Signal-to-Noise Ratio (pSNR) in decibels (dB). The library will attempt to match the output image pSNR to the specified value by estimating the distortion introduced in the image during compression. This process is less accurate if a color transform is used on the first 3 color channels in the image. For images with more than one component, the library will attempt to match the pSNR of each channel to the specified pSNR.

To get the most accurate pSNR results using this function, set the coding predictor offset using the function aw_j2k_set_output_j2k_coding_predictor_offset (page 158) to AW_J2K_PREDICTOR_CODE_ALL_DATA.

This function and the layer equivalent function (aw_j2k_set_output_j2k_layer_psnr, page 137) should be used when attempting to create a JPEG2000 output image with a specific quality, in terms of pSNR. For this reason, these functions are alternatives to the following bitrate, filesize and compression ratio targeting functions:

- aw_j2k_set_output_j2k_bitrate (page 106)
- aw_j2k_set_output_j2k_layer_bitrate (page 126)
- aw_j2k_set_output_j2k_ratio (page 107)
- aw_j2k_set_output_j2k_layer_ratio (page 130)
- aw_j2k_set_output_j2k_filesize (page 108)
- aw_j2k_set_output_j2k_layer_size (page 134)

Therefore, this function and it's layer equivalent function should not be used in conjunction with any of the functions listed above.

C Function Prototype

```
aw_j2k_set_output_j2k_psnr(
   aw_j2k_object *j2k_object,
   float psnr
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_psnr(
   psnr as float,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`psnr` — The target pSNR in dB. For images with more than one component, the library will attempt to match the pSNR for each channel to the target pSNR.

Default: none. This functionality will only get activated by calling this function with a specific value.

6.2.5 aw_j2k_set_output_j2k_tolerance

! **The use of this function is deprecated; setting the tolerance of the rate control process is not enabled at this time.**

Description

Sets a tolerance on matching the compression target (ratio, bitrate, or filesize). The library will not always compress an image to the exact file size, bitrate, or compression ratio requested by the user. By default, the library will run the compression a single time and return to the user the resulting file. If a tolerance is set, the library will instead run multiple compression attempts until the requested compression target and tolerance are matched. This allows for better matching of the requests, but causes slower operation.

The tolerance is expressed in percent when a compression ratio or bitrate are targeted. It is expressed in bytes when a file size is targeted.

C Function Prototype

```
aw_j2k_set_output_j2k_tolerance(
   aw_j2k_object *j2k_object,
   float tolerance
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_tolerance(
   tolerance as float,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tolerance` — Tolerance, expressed in percent for compression ratio or bitrate targets, and in bytes for file size targets. A value of 0 (zero) turns the iterator off and implies that no tolerance is applied to the compressor at all. Any other value will turn the compression iterator on, which will produce more predictably-compressed files but will take more time to do so.

6.2.6 aw_j2k_set_output_j2k_color_xform aw_j2k_set_output_j2k_color_xform_ex

Description

Turns the color transform option on and off. The first function sets this option for the entire image while the second function sets the same option for the specified tile. The color transform performs a color rotation from RBG to YUV color spaces. This option should be used when the input image is in RGB format.

C Function Prototype

```
aw_j2k_set_output_j2k_color_xform(
   aw_j2k_object *j2k_object,
   int color_xform
)

aw_j2k_set_output_j2k_color_xform_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   int color_xform
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_color_xform(
   color_xform as enumColorTransformation,
) as long

set_output_j2k_color_xform_ex(
   tile_index as long,
   color_xform as enumColorTransformation
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the option is set. Use `AW_J2K_SELECT_ALL_TILES` to set the option for all tiles.

`color_xform` — flag that sets the color transform on and off.

Default value: 1 (on).

6.2.7 aw_j2k_set_output_j2k_progression_order

Description

Sets the progression order of the compressed image file. JPEG2000 images can be organized progressively by:

1. Resolution, from a thumbnail to a full size image.
2. Quality, from low quality to best quality or lossless.
3. Color channel, from grayscale to full 3 channel color for 3 channel color images, or from the 1st color component to the last for images with more than 3 color components.
4. Spatial position, from top left to bottom right (this is not yet implemented in the library)

This function sets the progression order from one of these choices.

C Function Prototype

```
aw_j2k_set_output_j2k_progression_order(
   aw_j2k_object *j2k_object,
   int order
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_progression_order(
   order as long,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`order` — Progression order. The following constants are defined:

- `AW_J2K_PO_LAYER_RESOLUTION_COMPONENT_POSITION` – Progressive by quality
- `AW_J2K_PO_RESOLUTION_LAYER_COMPONENT_POSITION` – Progressive by resolution with secondary progression by quality
- `AW_J2K_PO_RESOLUTION_POSITION_COMPONENT_LAYER` – Progressive by resolution with secondary progression by spatial position

- `AW_J2K_PO_POSITION_COMPONENT_RESOLUTION_LAYER` – Progressive by spatial position
- `AW_J2K_PO_COMPONENT_POSITION_RESOLUTION_LAYER` – Progressive by color channel
- `AW_J2K_PO_DEFAULT` – Default progression. If the original image is not a JPEG2000 image (i.e. compression is being performed), the default progression is by resolution (with secondary progression by quality). If the original image is a JPEG2000 image (reformatting is being performed) the progression order of the original image is kept for the new image.

6.2.8 aw_j2k_set_output_j2k_tile_size

Description

Sets the tile size for JPEG2000 output (compressed) image. The JPEG2000 standard includes an option to divide the image into rectangular tiles, and compress each tile independently. This function is used to set the tile dimensions.

C Function Prototype

```
aw_j2k_set_output_j2k_tile_size(
   aw_j2k_object *j2k_object,
   unsigned long int width,
   unsigned long int height
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_tile_size(
  width as long,
  height as long,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`width` — width of tile size. Default value: the full image width, single tile. To reset to the default value, set width = 0.

`height` — height of tile size. Default value: the full image heigtht. To reset to the default value, set height = 0.

Notes:

If image offsets are used, the default value will be the full image dimensions minus the image offsets, in each direction.

If tile offsets are also used, the default value will be the full image dimensions minus the tile offsets, in each direction.

6.2.9 aw_j2k_set_output_j2k_add_comment

Description

Adds a comment to the JPEG2000 codestream. JPEG2000 allows the inclusion of comments in the main or tilepart headers. The comments can be either binary or ISO Latin 15 text. This function allows the user to specify the type of comment and the location of the comment in the codestream.

C Function Prototype

```
aw_j2k_set_output_j2k_add_comment(
   aw_j2k_object *j2k_object,
   char *data,
   int type,
   int length,
   int location
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_add_comment_text(
   data as String,
   location as enumCommentLocation,
) as long

set_output_j2k_add_comment_binary(
   data as Variant (array of bytes),
   location as enumCommentLocation
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`data` — The comment data.

`type` — JPEG2000 allows two types of comments. This pointer return the type of comment, defined by one of the following constants:

- `AW_J2K_CO_BINARY`
- `AW_J2K_CO_ISO_8859` For `AW_J2K_CO_BINARY` the comment is a binary stream of data. For `AW_J2K_CO_ISO_8859` the comment should be interpreted as being text created using ISO 8859-15:1999 Latin values.

`length` – The comment length. The maximum comment length is 65531 bytes.

`location` – The location of the comment. Comments can be included in the main header at the beginning of the file, or the in a tilepart header of a specifed tile. Use `AW_J2K_CO_LOCATION_MAIN_HEADER` to include the comment in the main header. Use a specific tile number to include the comment in the tilepart header of that tile. For an image with a single tile, use a value of 0 to specify that the comment should be included in the tilepart header.

- `AW_J2K_CO_LOCATION_TILEPART_HEADER`

6.3 Advanced JPEG2000 Compression and Output Reformatting Functions

The functions described in this section control the advanced options that are available when creating a JPEG2000 Output Image. They are only applicable when the output image type is set to JPEG2000.

Usage:
The functions described in this section are used for the following operations:

1. Compression of JPEG2000 Image. These functions control the advanced options of a compression operation, set the compression ratio, progression order, etc.

2. Reformatting of a JPEG2000 Image. These output functions set the advanced options that determine what type of JPEG2000 image will be created as a result of the reformatting operation. For example, to set the number of quality layers in the image, you would set the layers parameter using the function `aw_j2k_set_output_j2k_layers` (page 125).

6.3.1 aw_j2k_set_output_j2k_image_offset

Description

Sets the horizontal and vertical image offset from upper left pixel in the image. This function can be used to crop rows from the top and columns from the left side of the image. For tiled images, this function can be used to position the tiles in specific locations on the image.

C Function Prototype

```
aw_j2k_set_output_j2k_image_offset(
   aw_j2k_object *j2k_object,
   unsigned long int x,
   unsigned long int y
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_image_offset(
   x as long,
   y as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`x` — horizontal offset, in columns, from the origin of the reference grid to the location of the upper left pixel in the image.

`y` — vertical offset, in rows, from the origin of the reference grid to the location of the upper left pixel in the image.

Example Code

To crop the leftmost 20 columns and uppermost 10 rows of pixels from the image, use x=20 and y=10.

```
return_value = aw_j2k_set_output_j2k_image_offset(j2k_object, 20, 10);
```

6.3.2 aw_j2k_set_output_j2k_tile_offset

Description

Sets the horizontal and vertical tile offset from upper left pixel in the image. This function can be used to set the upper left tile to be a partial tile, which can be useful in positioning or centering tiles on the image.

C Function Prototype

```
aw_j2k_set_output_j2k_tile_offset(
   aw_j2k_object *j2k_object,
   unsigned long int x,
   unsigned long int y
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_tile_offset(
   x as long,
   y as long,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`x` — horizontal offset, in columns, from the upper left pixel in the image to the upper left pixel in the upper left tile. Default value: 0 or the value of the horizontal image offset, if set.

`y` — vertical offset, in rows, from the the upper left pixel in the image to the upper left pixel in the upper left tile. Default value: 0 or the value of the vertical image offset, if set.

Tile offsets must be used in conjunction with image offsets, which are set using the `aw_j2k_set_output_j2k_image_offset` function (page 119). Tile offsets must be smaller than or equal to the image offsets, in each direction.

6.3.3 aw_j2k_set_output_j2k_error_resilience aw_j2k_set_output_j2k_error_resilience_ex

Description

Sets error resilience options: start of packet headers, end of packet headers, and segmentation symbols. The first function sets these options for the entire image while the second function sets the same options for the specified component of the specified tile. These options add redundant information in the JPEG2000 image file. This information can be used by a decoder to reduce the effect of bit errors in the image file. These options should be used in situations when the compressed JPEG2000 image file will be transmitted over a noisy channel or other noise prone environments.

Use of these options will reduce the effective compression ratio due to the additional bits included in the image file.

C Function Prototype

```
aw_j2k_set_output_j2k_error_resilience(
   aw_j2k_object *j2k_object,
   int SOP,
   int EPH,
   int SEG_SYMB
)

aw_j2k_set_output_j2k_error_resilience_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   int channel_index,
   int SOP,
   int EPH,
   int SEG_SYMB
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_error_resilience(
   SOP as long,
   EPH as long,
   SEG_SYMB as long,
) as long

set_output_j2k_error_resilience_ex(
```

```
   tile_index as long,
   channel_index as long,
   SOP as long,
   EPH as long,
   SEG_SYMB as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the options are set. Use `AW_J2K_SELECT_ALL_TILES` to set the options for all tiles.

`channel_index` — Index of the component for which the options are set. Use `AW_J2K_SELECT_ALL_CHANNELS` to set the options for all components.

`SOP` — Add start of packet markers. Values: TRUE or FALSE.

Default: FALSE.

`EPH` — Add end of packet markers. Values: TRUE or FALSE.

Default: FALSE.

`SEG_SYMB` — Add segmentation symbols. Values: TRUE or FALSE.

Default: FALSE.

6.3.4 aw_j2k_set_output_j2k_xform aw_j2k_set_output_j2k_xform_ex

Description

Sets the type of wavelet transform to use and the number of decomposition levels. The first function sets these options for the entire image while the second function sets the same options for the specified component of the specified tile. The JPEG2000 standard allows for the use of 2 wavelet transforms:

- The irreversible 9-7 wavelet transform for lossy compression.
- The reversible 5-3 wavelet transform for lossless or lossy compression.

Of these two wavelet transform choices, the irreversible 9-7 wavelet tends to produce higher quality lossy compressed images, while the reversible 5-3 wavelet allows for lossless compression.

Selecting a wavelet transform type also determines the type of quantization that will be used to compress the image. Selecting the irreversible 9-7 wavelet transform automatically sets the quantization type to 9-7 Expounded quantization, unless the quantization has been explicitly set to Derived quantization. Selecting the reversible 5-3 automatically sets the quantization type to 5-3 Reversible quantization. See aw_j2k_set_output_j2k_channel_quantization (page 140) for details on how to explicitly set the quantization type.

C Function Prototype

```
aw_j2k_set_output_j2k_xform(
   aw_j2k_object *j2k_object,
   int type,
   int levels
)
aw_j2k_set_output_j2k_xform_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   int channel_index,
   int type,
   int levels
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_xform(
   type as enumWaveletTransformTypes,
   levels as long
```

```
) as long

set_output_j2k_xform_ex(
   tile_index as long,
   channel_index as long,
   type as enumWaveletTransformTypes,
   levels as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the options are set. Use `AW_J2K_SELECT_ALL_TILES` to set the options for all tiles.

`channel_index` — Index of the component for which the options are set. Use `AW_J2K_SELECT_ALL_CHANNELS` to set the options for all components.

`type` — wavelet transform selection. I9-7 uses the irreversible 9-7 wavelet filter, R5-3 uses the reversible 5-3 wavelet filter.

The following constants are defined.

- AW_J2K_WV_TYPE_I97
- AW_J2K_WV_TYPE_R53

Default value: I9-7

`levels` — number of decomposition levels to perform in the wavelet transform. The number of decomposition levels in the wavelet transform determines the number of resolution levels that will be available in the resulting JPEG2000 image.

By default, the number of decomposition levels is set so that the smallest dimension of the smallest resolution is between 16 and 31 pixels, with at least 3 decomposition levels at all times.

6.3.5 aw_j2k_set_output_j2k_layers

Description

JPEG2000 images may contain multiple representations of the same images at different quality levels, stored progressively. Each additional quality level is stored as a layer which adds information (and therefore quality) to the previously encoded layers. If all of the layers in a file are decoded, the best image is decoded from the file; if fewer layers are decoded, a lower quality image results.

This function sets the number of quality layers in the compressed file.

C Function Prototype

```
aw_j2k_set_output_j2k_layers(
   aw_j2k_object *j2k_object,
   int layers
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_layers(
   layers as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`layers` — the number of quality layers in the compressed image file.

Default: 1.

Allowable values 1 – 65535.

6.3.6 aw_j2k_set_output_j2k_layer_bitrate aw_j2k_set_output_j2k_layer_bitrate_ex

Description

Sets the layer bitrate, in bits per pixel, for the specified layer in the output JPEG-2000 image. The first function sets this option for the entire image while the second function sets the same option for the specified tile. The layer bitrate for the entire image is defined as:

$$\text{layer bitrate} = B * \frac{Bytes_{layer}}{Bytes_{image}}, \tag{6.1}$$

where B is the original image bitdepth, $Bytes_{layer}$ is the number of bytes in the layers up to and including the specified layer, and $Bytes_{image}$ is the number of bytes in the original input image. For input images that are in a compressed image format, such as JPEG and JPEG2000, the number of bytes in the original image is based on the total size of the input image after it has been decompressed.

Similarly, the layer bitrate for a specific tile is defined as:

$$\text{tile layer bitrate} = B * \frac{Bytes_{layer}}{Bytes_{tile}}, \tag{6.2}$$

where B is the original image bitdepth, $Bytes_{layer}$ is the number of bytes for this tile in the layers up to and including the specified layer, and $Bytes_{tile}$ is the number of bytes in this tile in the original input image. For input images that are in a compressed image format, such as JPEG and JPEG2000, the number of bytes in this tile in the original image is based on the total size of the tile in the input image after it has been decompressed.

This function is similar to `aw_j2k_set_output_j2k_bitrate` (see page 107), which defines the global bitrate for the entire JPEG2000 image. Using the `aw_j2k_set_output_j2k_layer_bitrate` function with the layer index set to `AW_J2K_GLOBAL_LAYER_INDEX` is identical to using the `aw_j2k_set_output_j2k_bitrate` function.

The `aw_j2k_set_output_j2k_layer_bitrate` function can be used in place of `aw_j2k_set_output_j2k_layer_ratio` (see page 130) and `aw_j2k_set_output_j2k_layer_size` (see page 134) to specify layer sizes. If more than one of these functions is called, whichever function was called later takes effect.

The `aw_j2k_set_output_j2k_layer_bitrate` function can be called multiple times to set the bitrates for several layers. See the Examples below for sample code that details this functionality.

For a discussion of layers in a JPEG2000 image see the description of the function `aw_j2k_set_output_j2k_layers` (page 125).

C Function Prototype

```
aw_j2k_set_output_j2k_layer_bitrate(
```

```
   aw_j2k_object *j2k_object,
   int layer_index,
   float bpp
)

aw_j2k_set_output_j2k_layer_bitrate_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   int layer_index,
   float bpp
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_layer_bitrate(
   layer_index as long,
   bpp as float,
) as long

set_output_j2k_layer_bitrate_ex(
   tile_index as long,
   layer_index as long,
   bpp as float
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the option is set. Use `AW_J2K_SELECT_ALL_TILES` to set the option for all tiles.

`layer_index` — The layer index. Use 0 for the first layer, N-1 for the Nth layer. If you use this function with `layer_index` larger than the number of layers in the image (which can be set using `aw_j2k_set_output_j2k_layers`, see page 125), the number of layers in the image will be increased accordingly.

Use `AW_J2K_GLOBAL_LAYER_INDEX` to set the global bitrate using this function.

`bpp` — The layer bitrate, as defined above.

Default: a bitrate equivalent to 50:1 compression ratio for the first layer (layer 0) in the image and equivalent to 10:1 compression ratio for the last layer in the image; intermediate values in between, depending on the number of layers

in the image. If there is only one layer in the image, the default is a bitrate equivalent to a compression ratio of 10:1.

Example 1

This example shows how to use this function to create an image with multiple layers at different bitrates.

```
aw_j2k_set_output_j2k_layer_bitrate(j2k_object, 0, 0.1);
aw_j2k_set_output_j2k_layer_bitrate(j2k_object, 9, 1);
```

The first line of code above sets the bitrate for the first layer to 0.1 bit/pixel. The second line sets the bitrate for layer 9 to 1 bit/pixel. The library would then interpolate linearly between bitrate values of 0.1 and 1 for the intermediate layers, leading to a bitrate of .2 for layer 1, .3 for layer 2, .4 for layer 3, etc, up to .9 for layer 8.

On the other hand, swapping the bitrates in the above code to:

```
aw_j2k_set_output_j2k_layer_bitrate(j2k_object, 0, 1);
aw_j2k_set_output_j2k_layer_bitrate(j2k_object, 9, 0.1);
```

would result in the library setting the bitrate for layer 0 to be 1 bit/pixel and then creating 9 empty quality layers after layer 0.

Example 2

This example defines the precedence rules for using this function with the `aw_j2k_set_output_j2k_layers` and `aw_j2k_set_output_j2k_bitrate` functions.

```
aw_j2k_set_output_j2k_layers(j2k_object, 5);
aw_j2k_set_output_j2k_bitrate(j2k_object, 0.5);
aw_j2k_set_output_j2k_layer_bitrate(j2k_object, 10, 1);
```

The first line of code sets the number of layers in the image to 5. The second line would set the bitrate to 0.5 bit/pixel for the entire image. This line is identical to using:

```
aw_j2k_set_output_j2k_layer_bitrate(j2k_object,
AW_J2K_GLOBAL_LAYER_INDEX , 0.5);
```

The third line overrides both of the layer and bitrate settings of the previous lines. It would set the number of layers to 11 and the bitrate for layer 10 to 1 bit/pixel. The bitrate for layers 0 to 9 would be set by a linear interpolation between the default bitrate for the first layer (which is the bitrate equivalent to a compression ratio of 50:1) and the 1 bit/pixel bitrate specified by this line for layer 10.

Note that the layer bitrates set using `aw_j2k_set_output_j2k_layer_bitrate` always take precedence over global bitrates set using `aw_j2k_set_output_j2k_bitrate`, regardless of the order in which the functions are called.

6.3.7 aw_j2k_set_output_j2k_layer_ratio aw_j2k_set_output_j2k_layer_ratio_ex

Description

Sets the layer compression ratio for the specified layer in the output JPEG2000 image. The first function sets this option for the entire image while the second function sets the same option for the specified tile. The layer compression ratio is defined as the number of bytes in the original image divided by the number of bytes in the layers up to and including the specified layer. For input images that are in a compressed image format, such as JPEG and JPEG2000, the number of bytes in the original image is based on the total size of the input image after it has been decompressed.

Similarly, for a specific tile, the layer compression ratio is defined as the number of bytes in this tile of the original image divided by the number of bytes in the layers up to and including the specified layer, for the specified tile. For input images that are in a compressed image format, such as JPEG and JPEG2000, the number of bytes in the tile of the original image is based on the total size of the tile in the input image after it has been decompressed.

This function is similar to `aw_j2k_set_output_j2k_ratio` (see page 107), which defines the global compression ratio for the entire JPEG2000 image. Using this function with the layer index set to `AW_J2K_GLOBAL_LAYER_INDEX` is identical to using the `aw_j2k_set_output_j2k_ratio` function.

This function can be used in place of `aw_j2k_set_output_j2k_layer_bitrate` (page 126) and `aw_j2k_set_output_j2k_layer_size` (page 134) to specify layer sizes. If more than one of these functions is called, whichever function was called later takes effect.

This function can be used multiple times to set the layer compression ratio for many layers. See the Examples below for sample code that details this functionality.

For a discussion of layers in a JPEG2000 image see the description of the function `aw_j2k_set_output_j2k_layers` (see page 125).

C Function Prototype

```
aw_j2k_set_output_j2k_layer_ratio(
    aw_j2k_object *j2k_object,
    int layer_index,
    float ratio
)

aw_j2k_set_output_j2k_layer_ratio_ex(
    aw_j2k_object *j2k_object,
    int tile_index,
    int layer_index,
    float ratio
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_layer_ratio(
   layer_index as long,
   ratio as float,
) as long

set_output_j2k_layer_ratio_ex(
   tile_index as long,
   layer_index as long,
   ratio as float
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the option is set. Use `AW_J2K_SELECT_ALL_TILES` to set the option for all tiles.

`layer_index` — The layer index. Use 0 for the first layer, N-1 for the Nth layer. If you use this function with `layer_index` larger than the number of layers in the image (which can be set using `aw_j2k_set_output_j2k_layers`, see page 125), the number of layers in the image will be increased accordingly.

Use `AW_J2K_GLOBAL_LAYER_INDEX` to set the global compression ratio using this function.

`ratio` — The layer compression ratio, defined as the number of bytes in the input image divided by the number of bytes in the layers up to and including the layer specified by `layer_index`. For input images that are in a compressed image format, such as JPEG and JPEG2000, the compression ratio is based on the total size of the input image after it has been decompressed.

Default value: 50 for the first layer (layer 0) and 10 for the last layer; intermediate values in between, depending on the number of layers in the image. If there is only one layer in the image, the default compression ratio is 10.

Example 1

This example shows how to use this function to create an image with multiple layers at different compression ratios.

```
aw_j2k_set_output_j2k_layer_ratio(j2k_object, 0, 75);
```

```
aw_j2k_set_output_j2k_layer_ratio(j2k_object, 14, 5);
```

The first line of code above sets the layer compression ratio for layer 0 to 75:1. The second line sets the layer compression ratio for layer 14 to 5:1. The library would then convert these compression ratios to bitrates and interpolate linearly in the bitrate interval between the 75:1 and 5:1 compression ratios for the intermediate layers. See the description of the `aw_j2k_set_output_j2k_layer_bitrate` function for more details on the resulting interpolated values.

On the other hand, swapping the compression ratios in the above code to:

```
aw_j2k_set_output_j2k_layer_ratio(j2k_object, 0, 5);
aw_j2k_set_output_j2k_layer_ratio(j2k_object, 14, 75);
```

would result in the library setting the compression ratio for layer 0 to be 5:1 and then creating 14 empty quality layers after layer 0.

Example 2

This example defines the precedence rules for using this function with the `aw_j2k_set_output_j2k_layers` and `aw_j2k_set_output_j2k_ratio` functions.

```
aw_j2k_set_output_j2k_layers(j2k_object, 5);
aw_j2k_set_output_j2k_ratio(j2k_object, 50);
aw_j2k_set_output_j2k_layer_ratio(j2k_object, 10, 15);
```

The first line of code would set the number of layers in the image to 5. The second line would set the compression ratio to 50:1 for the entire image. This line is identical to using:

```
aw_j2k_set_output_j2k_layer_ratio(j2k_object,
AW_J2K_GLOBAL_LAYER_INDEX , 50);
```

The third line overrides both of the layer and compression ratio settings of the previous lines. It would set the number of layers to 11 and the compression ratio for layer 10 to 15:1. The compression ratio for layers 0 to 9 would be set by a linear interpolation (in the bitrate space) between the bitrates equivalent to the default compression ratio for the first layer (which is 50:1) and the 15:1 compression ratio specified by this line for layer 10.

Note that the layer ratios set using `aw_j2k_set_output_j2k_layer_ratio` always take precedence over global ratio set using `aw_j2k_set_output_j2k_ratio`, regardless of the order in which the functions are called.

6.3.8 aw_j2k_set_output_j2k_layer_size aw_j2k_set_output_j2k_layer_size_ex

Description

Sets the layer size, defined as the number of bytes in the layers up to and including the specified layer. The first function sets this option for the entire image while the second function sets the same option for the specified tile. This function is similar to aw_j2k_set_output_j2k_filesize (see page 108), which defines the global filesize for the entire JPEG2000 image. Using this function with the layer index set to AW_J2K_GLOBAL_LAYER_INDEX is identical to using the aw_j2k_set_output_j2k_filesize function.

This function can be used in place of aw_j2k_set_output_j2k_layer_bitrate (page 126) and aw_j2k_set_output_j2k_layer_ratio (page 130) to specify layer sizes. If more than one of these functions is called, whichever function was called later takes effect.

This function can be used multiple times to set the layer sizes for several layers. See the Examples below for sample code that details this functionality.

For a discussion of layers in a JPEG2000 image see the description of the function aw_j2k_set_output_j2k_layers (see page 125).

C Function Prototype

```
aw_j2k_set_output_j2k_layer_size(
    aw_j2k_object *j2k_object,
    int layer_index,
    unsigned int cbBytes
)
aw_j2k_set_output_j2k_layer_size_ex(
    aw_j2k_object *j2k_object,
    int tile_index,
    int layer_index,
    unsigned int cbBytes
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_layer_size(
    layer_index as long,
    cbBytes as long
) as long

set_output_j2k_layer_size_ex(
    tile_index as long,
```

```
   layer_index as long,
   cbBytes as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the option is set. Use `AW_J2K_SELECT_ALL_TILES` to set the option for all tiles.

`layer_index` — The layer index. Use 0 for the first layer, N-1 for the Nth layer. If you use this function with `layer_index` larger than the number of layers in the image (which can be set using `aw_j2k_set_output_j2k_layers` (see page 125), the number of layers in the image will be increased accordingly.

Use `AW_J2K_GLOBAL_LAYER_INDEX` to set the global filesize using this function.

`cbBytes` — The number of bytes in the layers up to and including the layer specified by `layer_index`.

Default value: a layer size equivalent to 50:1 compression ratio for the first layer (layer 0) and 10:1 compression ratio for the last layer; intermediate values in between, depending on the number of layers in the image. If there is only one layer in the image, the default is a layer size equivalent to a compression ratio of 10:1.

Example 1

This example shows how to use this function to set multiple layer sizes.

```
aw_j2k_set_output_j2k_layer_size(j2k_object, 0, 1000);
aw_j2k_set_output_j2k_layer_size(j2k_object, 9, 10000);
```

The first line of code above sets the layer size for layer 0 to 1000 bytes. The second line sets the layer size for layer 9 to 10000 bytes. The library would then convert these layer sizes into bitrates and interpolate linearly in the bitrate space between the 1000 byte and 10000 byte layer sizes to set the layer sizes for the intermediate layers. See the description of the `aw_j2k_set_output_j2k_layer_bitrate` function for more details on the resulting interpolated values.

On the other hand, swapping the layer sizes in the above code to:

```
aw_j2k_set_output_j2k_layer_size(j2k_object, 0, 10000);
aw_j2k_set_output_j2k_layer_size(j2k_object, 9, 1000);
```

would result in the library setting the layer size for layer 0 to be 10000 bytes and then creating 9 empty quality layers after layer 0.

Example 2

This example defines the precedence rules for using this function with the `aw_j2k_set_output_j2k_layers` and `aw_j2k_set_output_j2k_filesize` functions.

```
aw_j2k_set_output_j2k_layers(j2k_object, 5);
aw_j2k_set_output_j2k_filesize(j2k_object, 5000);
aw_j2k_set_output_j2k_layer_size(j2k_object, 10, 15000);
```

The first line of code would set the number of layers in the image to 5. The second line would set the filesize to 5000 bytes for the entire image. This line is identical to using:

```
aw_j2k_set_output_j2k_layer_size(j2k_object,
AW_J2K_GLOBAL_LAYER_INDEX , 5000)
```

The third line overrides both of the layer and compression ratio settings of the previous lines. It would set the number of layers to 11 and the layer size for layer 10 to 15000. The layer sizes for layers 0 to 9 would be set by a linear interpolation (in the bitrate space) between the bitrates equivalent to the default layer size for the first layer (which is a layer size equivalent to a compression ratio of 50:1) and the layer size of 15000 bytes specified by this line for layer 10.
Note that the layer sizes set using `aw_j2k_set_output_j2k_layer_size` always take precedence over global filesizes set using `aw_j2k_set_output_j2k_filesize`, regardless of the order in which the functions are called.

6.3.9 aw_j2k_set_output_j2k_layer_psnr aw_j2k_set_output_j2k_layer_psnr_ex

Description

Sets the output image quality, measured by the peak Signal-to-Noise Ratio (pSNR) in decibels (dB), for the specified layer in the output JPEG2000 image. The first function sets this option for the entire image while the second function sets the same option for the specified tile. This function can be used multiple times to set the target pSNR values for several layers. If the target pSNR is not specified for all the layers in the image, intermediate layer pSNR values will be linearly interpolated (see Example below for details). The library will attempt to match the output pSNR for the specified layer to the target value by estimating the distortion introduced in the image during compression. This process is less accurate if a color transform is used on the first 3 color channels in the image. For images with more than one component, the library will attempt to match the pSNR for the specified layer of each channel to the specified pSNR.

For a discussion of layers in a JPEG2000 image see the description of the function `aw_j2k_set_output_j2k_layers` (page 125). To set the output image pSNR for an entire JPEG2000 image use the function (`aw_j2k_set_output_j2k_psnr`, page 109).

To get the most accurate pSNR results using this function, set the coding predictor offset using the function `aw_j2k_set_output_j2k_coding_predictor_offset` (page 158) to `AW_J2K_PREDICTOR_CODE_ALL_DATA`.

This function and the image equivalent function (`aw_j2k_set_output_j2k_psnr`, page 109) should be used when attempting to create a JPEG2000 output image with a specific quality, in terms of pSNR. For this reason, these functions are alternatives to the following bitrate, filesize and compression ratio targeting functions:

- `aw_j2k_set_output_j2k_bitrate` (page 106)
- `aw_j2k_set_output_j2k_layer_bitrate` (page 126)
- `aw_j2k_set_output_j2k_ratio` (page 107)
- `aw_j2k_set_output_j2k_layer_ratio` (page 130)
- `aw_j2k_set_output_j2k_filesize` (page 108)
- `aw_j2k_set_output_j2k_layer_size` (page 134)

Therefore, this function and it's layer equivalent function should not be used in conjunction with any of the functions listed above.

C Function Prototype

```
aw_j2k_set_output_j2k_layer_psnr(
   aw_j2k_object *j2k_object,
   int layer_index,
   float psnr
)
aw_j2k_set_output_j2k_layer_psnr_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   int layer_index,
   float psnr
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_layer_psnr(
   layer_index as long,
   psnr as float
) as long

set_output_j2k_layer_psnr_ex(
   tile_index as long,
   layer_index as long,
   psnr as float
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the option is set. Use `AW_J2K_SELECT_ALL_TILES` to set the option for all tiles.

`layer_index` — The layer index. Use 0 for the first layer, N-1 for the Nth layer. If you use this function with `layer_index` larger than the number of layers in the image (which can be set using `aw_j2k_set_output_j2k_layers`, page 125), the number of layers in the image will be increased accordingly.

`psnr` — The target pSNR for the specified layer in dB. For images with more than one component, the library will attempt to match the pSNR for the specified layer of each channel to the target pSNR.

Default: none. This functionality will only get activated by calling this function with a specific value.

Example

This example shows how to use this function to create an image with multiple layers at different pSNR values.

```
aw_j2k_set_output_j2k_layer_psnr(j2k_object, 0, 30);
aw_j2k_set_output_j2k_layer_psnr(j2k_object, 10, 50);
```

The first line of code above sets the target pSNR for the first layer (layer 0) to 30 dB. The second line sets the target pSNR for layer 10 to 50 dB. The library would then interpolate linearly between pSNR values of 30 and 50 for the intermediate layers, leading to a target pSNR of 32 for layer 1, 34 for layer 2, 36 for layer 3, 38 for layer 4, etc., up to 50 for layer 10.

6.3.10 aw_j2k_set_output_j2k_channel_quantization aw_j2k_set_output_j2k_quantization_ex

Description

Sets the output image quantization type for a JPEG2000 output image. The first sets this option for the specific component and the seconf function sets the same option for the specified component of the specified tile.

JPEG2000 is a lossy compression standard, which means that the decompressed image will not be identical to the compressed image. The JPEG2000 library supports two different schemes for quantization (the operation that causes the data loss) for the 9-7 irreversible wavelet transform and one scheme for the 5-3 reversible wavelet transform. The quantization type affects the quality of the compressed image, as well as the resulting compressed file size. This function selects the quantization type for a JPEG2000 output image.

C Function Prototype

```
aw_j2k_set_output_j2k_channel_quantization(
   aw_j2k_object *j2k_object,
   int channel_index,
   int quantization_option
)

aw_j2k_set_output_j2k_quantization_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   int channel_index,
   int quantization_option
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_channel_quantization(
   channel_index as long,
   quant_option as enumQuantizationOptions
) as long

set_output_j2k_quantization_ex(
   tile_index as long,
   channel_index as long,
   quant_option as enumQuantizationOptions
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the option is set. Use `AW_J2K_SELECT_ALL_TILES` to set the option for all tiles.

`channel_index` — Index of the component for which the option is set. Use `AW_J2K_SELECT_ALL_CHANNELS` to set the option for all components.

`quantization_option` — Quantization type.

- 5-3 Reversible Quantization (use `AW_J2K_QUANTIZATION_REVERSIBLE`): The wavelet coefficients are not quantized, compression is achieved via truncation of the encoded data. This option can only be used with the Reversible 5-3 wavelet transform. See `aw_j2k_set_output_j2k_xform` (page 123) for details on how to set the wavelet transform type. If this quantization option is used, the wavelet transform type will be automatically set to the Reversible 5-3 (R53) transform.
- 9-7 Expounded Quantization (use `AW_J2K_QUANTIZATION_EXPOUNDED`): The quantization binwidths are calculated for each subband separately and the image header includes the quantization binwidths for all the subbands. This option can only be used with the Irreversible 9-7 wavelet transform. See `aw_j2k_set_output_j2k_xform` (page 123) for details on how to set the wavelet transform type. If this quantization option is used, the wavelet transform type will be automatically set to the Irreversible 9-7 (I97) transform.

  In general, this option tends to produce slightly higher quality images than the Derived option below, but increases the number of bytes used to signal the quantization parameters in the compressed file. For this reason, this option may actually result in slightly lower quality images for small images compressed at very high compression ratios.
- 9-7 Derived Quantization (use `AW_J2K_QUANTIZATION_DERIVED`): The quantization binwidths are calculated only for the lowest resolution subband and the image header only includes the quantization binwidths for the lowest resolution subband. To decode an image created with this option, the decoder must calculate the quantization parameters for the other subbands using the lowest resolution subband quantization values. This option can only be used with the Irreversible 9-7 wavelet transform. See `aw_j2k_set_output_j2k_xform` (page 123) for details on how to set the wavelet transform type. If this quantization option is used, the wavelet transform type will be automatically set to the Irreversible 9-7 (I97) transform.

In general, this tends to produce slightly lower quality images than the Expounded option described above, but reduces the number of bytes used to signal the quantization parameters in the compressed file. For this reason, this option may actually result in slightly higher quality images for small images compressed at very high compression ratios.

Default value:: 9-7 Expounded Quantization for the irreversible 9-7 wavelet transform and 5-3 Reversible Quantization for the reversible 5-3 wavelet transform.

6.3.11 aw_j2k_set_output_j2k_quant_options

The use of this function is deprecated; use aw_j2k_set_output_j2k_channel_quantization instead (page 140), with a channel_index value of -1 to set the quantization type for all the color channels in the image.

Description

JPEG2000 is a lossy compression standard, which means that the decompressed image will not be identical to the compressed image. The JPEG2000 library supports multiple schemes for quantization (the operation that causes the data loss). This function selects among the various quantization options. The result will affect the quality of the compressed file, as well as the resulting compressed file size. The aw_j2k_set_output_j2k_tolerance function (page 111) can be used to controlled the variability of the compressed file size.

C Function Prototype

```
aw_j2k_set_output_j2k_quant_options(
   aw_j2k_object *j2k_object,
   int quant_option
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_quant_options(
   quant_option as enumQuantizationOptions,
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

quant_option — Quantization type. This option controls the method by which the compressor tries to achieve the target compression ratio. This argument can be set to one of the following values:

- AW_J2K_Q_TYPE_EXPOUNDED – signals that normal quantization is performed on this channel.

- `AW_J2K_Q_TYPE_DERIVED` – signals that derived quantization is performed; i.e. that normal quantization is performed on the lowest resolution and that the quantization parameters for the higher resolution levels are derived from it. This option saves a few bytes from the JPEG2000 headers while sacrificing image quality.

6.3.12 aw_j2k_set_output_j2k_quantization_binwidth_scale aw_j2k_set_output_j2k_quantization_binwidth_scale_ex

Description

Sets the quantization binwidth scaling. The first function sets this option for the entire image while the second function sets the same option for the specified component of the specified tile.

The JPEG2000 compressor has two modes of operation — a reversible mode, and an irreversible mode. In the reversible mode, an integer wavelet transform is performed on the image, and the results are coded. In the irreversible mode, a rational wavelet transform is performed on the data, the results are quantized, and then coded. The compression ratio is controlled by discarding unwanted data.

If the binwidths used in the irreversible mode are too large, the compressor will not be able to create files larger than a certain size (and this size depends on the image), meaning that it cannot meet very low compression ratios. Conversely, if the binwidths are too small, the compressor may encode much more data than is necessary, slowing compression down.

This function scales the binwidths. If low compression ratios are not achievable, a scaling factor smaller than 1 can be used to enable them. A scaling factor larger than 1 can be used to reduce the amount of work the compressor has to do on higher compression ratios.

This function has no effect on the reversible mode. The irreversible mode is selected by calling `aw_j2k_set_output_j2k_xform` with `AW_J2K_WV_TYPE_I97` as the type argument (see page 140).

C Function Prototype

```
aw_j2k_set_output_j2k_quantization_binwidth_scale(
   aw_j2k_object *j2k_object,
   float binwidth_scale
)

aw_j2k_set_output_j2k_quantization_binwidth_scale_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   int channel_index,
   float binwidth_scale
)
```

ActiveX Visual Basic Function Prototype

```
aw_j2k_set_output_j2k_quantization_binwidth_scale(
   binwidth_scale as float
```

```
) as long

aw_j2k_set_output_j2k_quantization_binwidth_scale_ex(
   tile_index as long,
   channel_index as long,
   binwidth_scale as float
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the options are set. Use `AW_J2K_SELECT_ALL_TILES` to set the options for all tiles.

`channel_index` — Index of the component for which the options are set. Use `AW_J2K_SELECT_ALL_CHANNELS` to set the options for all components.

`binwidth_scale` — The scaling factor that is applied to the quantization binwidths.

6.3.13 aw_j2k_set_output_j2k_guard_bits aw_j2k_set_output_j2k_guard_bits_ex

Description

Sets the number of guard bits during the encoding process. The first function sets this options for the entire image while the second function sets the same option for the specified component of the specified tile.

The JPEG2000 compressor encodes images by bitplanes. It uses a heuristic to pick the number of bitplanes to encode. To ensure that no bitplanes are missed, the compressor may be told to examine extra bitplanes. These extra bitplanes are known as "guard bits." This function sets their number. If the compressor detects that bitplanes were missed, it will issue the error AW_J2K_ERROR_GUARD_BITS_TOO_FEW (23). If this error is detected, the number of guard bits should be increased and compression reattempted.

The default value of 2 guard bits should cover most images.

C Function Prototype

```
aw_j2k_set_output_j2k_guard_bits(
   aw_j2k_object *j2k_object,
   int guard_bits
)

aw_j2k_set_output_j2k_guard_bits_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   int channel_index,
   int guard_bits
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_guard_bits(
  guard_bits as long
) as long

set_output_j2k_guard_bits_ex(
  tile_index as long,
  channel_index as long,
  guard_bits as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the options are set. Use `AW_J2K_SELECT_ALL_TILES` to set the options for all tiles.

`channel_index` — Index of the component for which the options are set. Use `AW_J2K_SELECT_ALL_CHANNELS` to set the options for all components.

`guard_bits` — The number of guard bits

Range: 0 to 7.

Default: 2.

6.3.14 aw_j2k_set_output_j2k_channel_precinct_size aw_j2k_set_output_j2k_precinct_size_ex

Description

Sets the dimensions of the precincts. The first function sets this option for the specified component of the entire image while the second function sets the same option for the for the specified component of the specified tile.

The wavelet transform converts the pixels of the image into coefficients, which are divided into different frequency subbands. Each subband is then quantized and divided into precincts. Dividing the subbands into precincts allows subregions of the subbands to be decoded independently of the rest of the subband. This allows for extraction of regions of interest on decode (see aw_j2k_set_input_j2k_region_level on page 80).

If precinct sizes are not specified (and this function is not called), the default size of the precincts, the maximal size, 32768 by 32768, is used at each resolution level.

If precinct sizes are specified for every resolution level in a channel, those sizes are used unmodified.

If precinct sizes are not specified for each resolution level in a channel, the unspecified sizes are extrapolated from the specified sizes by setting each unspecified size to double the size of the one-lower resolution level. If the lowest resolution level is not specified, it is set to half the size of the next higher resolution level. An example may illustrate this more clearly (and note that the values in the function calls are the base-2 logarithms of the actual precinct sizes).

```
aw_j2k_set_output_j2k_xform(j2k_object, AW_J2K_WV_TYPE_I97, 6);
aw_j2k_set_output_j2k_channel_precinct_size(j2k_object, 0, 2, 4, 4);
aw_j2k_set_output_j2k_channel_precinct_size(j2k_object, 0, 4, 8, 9);
```

The first function call sets the transform depth to 6, which means that there are 7 resolution levels. The second call sets the precinct size of resolution level 2 to be 2^4 by 2^4 or 16 by 16. The third call sets the precinct size of resolution level 4 to be 2^8 by 2^9 or 256 by 512. Left unspecified were the precinct sizes of resolution levels 0, 1, 3, 5, and 6. Level 6 will be double the size of level 5, which will be double the size of level 4. So level five precincts will be 512 by 1024, and level 6 will by 1024 by 2048. Level 3 precincts will be double that of level 2, i.e. 32 by 32. Since level 0 precinct sizes were unspecified, they will be half of level 1 sizes, which by same token be half of level 2. Level one sizes will therefore be 8 by 8, and level zero 4 by 4.

C Function Prototype

```
aw_j2k_set_output_j2k_channel_precinct_size(
   aw_j2k_object *j2k_object,
   int channel_index,
   int resolution_level,
```

```
   int precinct_height,
   int precinct_width
)

aw_j2k_set_output_j2k_channel_precinct_size_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   int channel_index,
   int resolution_level,
   int precinct_height,
   int precinct_width
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_channel_precinct_size(
   channel_index as long,
   resolution_level as long,
   precinct_height as long,
   precinct_width as long,
) as long

set_output_j2k_channel_precinct_size_ex(
   tile_index as long,
   channel_index as long,
   resolution_level as long,
   precinct_height as long,
   precinct_width as long,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the precinct size applies. Use `AW_J2K_SELECT_ALL_TILES` to set the precinct size for all tiles.

`channel_index` — The channel for which the precinct size applies. Use `AW_J2K_SELECT_ALL_CHANNELS` to indicate that the precinct size applies to all the color channels.

`resolution_level` — The resolution level for which the precinct size applies. The number of resolution levels is one more than the number of transform levels,

as set by `aw_j2k_set_output_j2k_xform` (see page 123). The lowest resolution level is level 0.

`precinct_height` — $\log_2$ of the precinct height.

Allowable values: 0–15 (precinct height of 1 to 32768). Value of 0 is only allowed for the lowest resolution level (i.e. level 0).

Default: 15 (precinct height of 32768)

`precinct_width` — $\log_2$ of the precinct width.

Allowable values: 0–15 (precinct width of 1 to 32768). Value of 0 is only allowed for the lowest resolution level (i.e. level 0).

Default: 15 (precinct width of 32768)

6.3.15 aw_j2k_set_output_j2k_codeblock_size aw_j2k_set_output_j2k_codeblock_size_ex

Description

Sets the dimensions of the codeblocks. The first function sets this option for the entire image while the second function sets the same option for the specified component of the specified tile.

The wavelet transform converts the pixels of the image into coeffcients, which are divided into different frequency subbands. Each subband is then quantized and divided into precincts. Each precinct is then divided into rectangular codeblocks, each of which is encoded separately. This function is used to set the width and height of the codeblocks.

The dimensions of the codeblocks are restricted to powers of two; consequently, this function accepts the exponent for each dimension (i.e. the $\log_2$ of each dimension).

Note that the actual sizes of the codeblocks are limited by the sizes of the tiles (set by `aw_j2k_set_output_j2k_tile_size`, described on page 115), by the number of transform levels (set by `aw_j2k_set_output_j2k_xform`, described on page 123), and by the sizes of the precincts (set by `aw_j2k_set_output_j2k_channel_precinct_size`, described on page 149).

C Function Prototype

```
aw_j2k_set_output_j2k_codeblock_size(
    aw_j2k_object *j2k_object,
    int height,
    int width
)

aw_j2k_set_output_j2k_codeblock_size_ex(
    aw_j2k_object *j2k_object,
    int tile_index,
    int channel_index,
    int height,
    int width
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_codeblock_size(
    height as long,
    width as long
) as long
```

```
set_output_j2k_codeblock_size_ex(
   tile_index as long,
   channel_index as long,
   height as long,
   width as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the option is set. Use `AW_J2K_SELECT_ALL_TILES` to set the option for all tiles.

`channel_index` — Index of the component for which the option is set. Use `AW_J2K_SELECT_ALL_CHANNELS` to set the option for all components.

`height` — $\log_2$ of the codeblock height.

Allowable values: 2–10 (codeblock height of 4 to 1024).

Default: 6 (codeblock height of 64)

`width` — $\log_2$ of the codeblock width.

Allowable values: 2–10 (codeblock width of 4 to 1024).

Default: 6 (codeblock width of 64)

Sum of height and width cannot exceed 12.

6.3.16 aw_j2k_set_output_j2k_arithmetic_coding_option aw_j2k_set_output_j2k_arithmetic_coding_option_ex

Description

Sets the arithmetic coding options. The first function sets these options for the entire image while the second function sets the same options for the specified component of the specified tile.

An arithmetic encoder is used in JPEG2000 to achieve compression. The JPEG-2000 standard allows various options in the setup and usage of the arithmetic encoder. This function is used to set the following options:

Arithmetic Context Reset — The arithmetic encoder builds a dynamic model of the data it encodes. This model, known as the context, may be reset at the end of every coding pass of each codeblock or only at the end of each codeblock. Resetting the context at the end of each coding pass increases the compressed file size while increasing error resilience.

Vertically Causal Context Formation — The arithmetic encoder scans the pixels in strips of 4 rows. The context it uses to encode each pixel depends on all of the pixel's neighbors. For pixels at the bottom of the strip this normally includes pixels in the next strip. In the vertically causal context formation style the pixels at the bottom of the strip do not depend on the pixels in the next strip, which can allow certain operations to occur in parallel. Note that the JPEG2000 codec does not do these operations in parallel; using this option merely allows the recipient of the image do so.

Arithmetic Termination — The arithmetic encoder's internal state may be reset to a known state at the end of every coding pass of each codeblock or only at the end of each codeblock. Terminating the arithmetic encoder more frequently (i. e. at the end of every coding pass) increases the compressed file size while increasing error resilience.

Predictable Termination — Resetting the arithmetic encoder's internal state may be achieved by several means. Choosing the Predictable Termination option allows a decoder to more easily detect certain errors in the bitstreams (errors that may have been introduced during transmission). Using this option will create smaller files if the arithmetic encoder is terminated at the end of each coding pass, and slightly larger ones otherwise.

C Function Prototype

```
aw_j2k_set_output_j2k_arithmetic_coding_option(
   aw_j2k_object *j2k_object,
   int option,
   int value
```

```
)
aw_j2k_set_output_j2k_arithmetic_coding_option_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   int channel_index,
   int option,
   int value
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_arithmetic_coding_option(
   option as long,
   value as long
) as long

set_output_j2k_arithmetic_coding_option_ex(
   tile_index as long,
   channel_index as long,
   option as long,
   value as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the options are set. Use `AW_J2K_SELECT_ALL_TILES` to set the options for all tiles.

`channel_index` — Index of the component for which the options are set. Use `AW_J2K_SELECT_ALL_CHANNELS` to set the options for all components.

`option` — Selects the arithmetic option to modify. The following constants, defined in j2k.h, are available:

- `AW_J2K_ARITH_OPT_CONTEXT_RESET` – used for setting the arithmetic context reset option.

  Allowable values for this option are:

  - `AW_J2K_ACR_AT_EACH_CODEPASS_END` – for resetting context after each coding pass.
  - `AW_J2K_ACR_ONLY_AT_CODEBLOCK_END` – for resetting context at the end of each codeblock only.

- AW_J2K_ARITH_OPT_VERTICALLY_CAUSAL_CONTEXT – used for selecting the vertically causal context formation style.

  Allowable values for this option are:

  - AW_J2K_VC_NO_VERTICALLY_CAUSAL_CONTEXT – for selecting the normal context formation style.
  - AW_J2K_VC_USE_VERTICALLY_CAUSAL_CONTEXT – for using the vertically causal style for context formation.

- AW_J2K_ARITH_OPT_ARITHMETIC_TERMINATION – used for setting the arithmetic termination option.

  Allowable values for this option are:

  - AW_J2K_AT_AT_EACH_CODEPASS_END – for terminating at the end of each coding pass.
  - AW_J2K_AT_ONLY_AT_CODEBLOCK_END – for terminating at the end of each codeblock only.

- AW_J2K_ARITH_OPT_PREDICTABLE_TERMINATION – used for setting the predictable termination option.

  - AW_J2K_PT_USE_PREDICTABLE_TERMINATION – to use predictable termination.
  - AW_J2K_PT_NO_PREDICTABLE_TERMINATION – to use normal termination.

value — Specifies the new value for the selected option. The allowable values for each option are listed above.

6.3.17 aw_j2k_set_output_j2k_coding_style

Description

Sets the location of the arithmetic coding reset.

The use of this function is deprecated; see aw_j2k_set_output_j2k_arithmetic_coding_option (page 154).

C Function Prototype

```
aw_j2k_set_output_j2k_coding_style(
   aw_j2k_object *j2k_object,
   int coding_style
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_coding_style(
   coding_style as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`coding_style` — Sets location at which end of the encoded section occurs. For more error resilience set the value to `AW_J2K_ACR_AT_EACH_CODEPASS_END` at the expense of larger file sizes. For smaller sizes and less resilience set the value to `AW_J2K_ACR_ONLY_AT_CODEBLOCK_END`.

Default Value: `AW_J2K_ACR_ONLY_AT_CODEBLOCK_END`

6.3.18 aw_j2k_set_output_j2k_coding_predictor_offset aw_j2k_set_output_j2k_coding_predictor_offset_ex

Description

Sets the predictor offset value during the compression process. The first function sets this option for the entire image while the second function sets the same option for the specified tile.

During the JPEG2000 compression process, all of the bitplanes of the wavelet coefficients are encoded using the arithmetic encoder. Depending on the desired compression ratio, a portion of the encoded data is discarded and the remaining data is used to create the compressed image. To make the compression process faster, we try to avoid coding data that will subsequently be discarded. To do this, we must predict how much data will be actually used to create the compressed image and stop coding shortly after we reach that point.

This function sets the offset from the predicted point at which we will stop coding. A smaller number will result in faster compression, though with a possible reduction in quality. A larger number will cause the library to code more data and possibly increase the quality.

C Function Prototype

```
aw_j2k_set_output_j2k_coding_predictor_offset(
   aw_j2k_object *j2k_object,
   int predictor_offset
)

aw_j2k_set_output_j2k_coding_predictor_offset_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   int predictor_offset
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_coding_predictor_offset(
   predictor_offset as long
) as long

set_output_j2k_coding_predictor_offset_ex(
   tile_index as long,
   predictor_offset as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the option is set. Use `AW_J2K_SELECT_ALL_TILES` to set the option for all tiles.

`predictor_offset` — The offset from the predicted point at which the library will stop coding. The default value is 2; a value of 1 will speed up compression and reduce quality; 3 will reduce speed and increase quality. Use `AW_J2K_PREDICTOR_CODE_ALL_DATA` to force the library to code everything, resulting in the lowest speed but best quality. Values larger than 3 will typically be equivalent to using `AW_J2K_PREDICTOR_CODE_ALL_DATA`.

6.3.19 aw_j2k_set_output_j2k_component_registration

Description

Sets the component registration offsets for the named component in a JPEG2000 image. The component registration offset specifies by what fraction each of the component's pixels is offset. This allows for the correct display when overlayed components are subsampled with respect to each other.

The offsets are specified in units of 1/65536 of the pixel spacing. E.g. a horizontal offset value of 32768 means that each pixel should be displayed 1/2 a pixel width to the right.

The function aw_j2k_get_input_j2k_component_registration (see page 69) can be used to retrieve the component registration offsets.

C Function Prototype

```
aw_j2k_set_output_j2k_component_registration(
    aw_j2k_object *j2k_object,
    WORD component,
    WORD Xreg,
    WORD Yreg
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_component_registration(
    component as long,
    Xreg as long,
    Yreg as long,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`component` — The component for which the offsets are set.

`Xreg` — Horizontal registration offset value.

`Yreg` — Vertical registration offset value.

6.3.20 aw_j2k_set_output_j2k_tile_toc

Description

This function instructs the library to add to a JPEG2000 file a Tile Length Marker (TLM) segment. TLM segments contain the lengths of each tile in the JPEG2000 image, which can speed up decoding of individual tiles from a multi-tile image.

C Function Prototype

```
aw_j2k_set_output_j2k_tile_toc(
   aw_j2k_object *j2k_object
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_tile_toc(
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

6.3.21 aw_j2k_set_output_j2k_tile_data_toc aw_j2k_set_output_j2k_tile_data_toc_ex

Description

This function instructs the library to add to a JPEG2000 file a Packet Length, Tile-part Header Marker (PLT) segment. PLT segments contain the lengths of each packet in each tile of the JPEG2000 image, which can speed up decoding of subregions. The first function sets this options for the entire image while the second function sets the same option for the specified tile.

C Function Prototype

```
aw_j2k_set_output_j2k_tile_data_toc(
   aw_j2k_object *j2k_object
)
aw_j2k_set_output_j2k_tile_data_toc_ex(
   aw_j2k_object *j2k_object,
   int tile_index,
   BOOL inclusion_flag
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_tile_toc(
) as long

set_output_j2k_tile_toc_ex(
  tile_index as long,
  inclusion_flag as Boolean
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`tile_index` — Index of the tile for which the option is set. Use `AW_J2K_SELECT_ALL_TILES` to set the option for all tiles.

`inclusion_flag` — This parameter should be set to true to enable this option for the specified tile, or set to false to disable this option for the specified tile.

6.3.22 aw_j2k_set_output_j2k_clear_comments

Description

Clears all comments.

C Function Prototype

```
aw_j2k_set_output_j2k_clear_comments(
   aw_j2k_object *j2k_object
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_clear_comments(
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

6.3.23 aw_j2k_get_output_j2k_layer_psnr_estimate

Description

This function returns the estimated JPEG2000 output image quality, measured by the peak Signal-to-Noise Ratio (pSNR) in decibels (dB), for the specified layer in the output JPEG2000 image. This function can be used multiple times to get the estimated pSNR values for several layers. The estimated pSNR values are less accurate if a color transform is used on the first 3 color channels in the image. For images with more than one component, the estimated pSNR is the average pSNR of the specified layer, over all color channels.

To get the most accurate pSNR estimates using this function, set the coding predictor offset using the function aw_j2k_set_output_j2k_coding_predictor_offset (see page 158) to AW_J2K_PREDICTOR_CODE_ALL_DATA.

Use of this function requires that an output image in JPEG2000 or JP2 format has been created by the library. Therefore, this function should be called only after calling either:

1. aw_j2k_set_output_type (page 95) with JPEG2000, JP2, or DICOM J2K as the image type, followed by aw_j2k_get_output_image (page 96) or aw_j2k_get_output_image_file (page 101)

2. aw_j2k_get_output_image_type (page 97) or aw_j2k_get_output_image_file_type (page 102) with JPEG2000, JP2, or DICOM J2K as the image type.

To set the output image pSNR for the entire image use the function aw_j2k_set_output_j2k_psnr (page 109). To set the output image pSNR for a specific layer use the function aw_j2k_set_output_j2k_layer_psnr (page 137).

For a discussion of layers in a JPEG2000 image see the description of the function aw_j2k_set_output_j2k_layers (page 125).

C Function Prototype

```
aw_j2k_get_output_j2k_layer_psnr_estimate(
   aw_j2k_object *j2k_object,
   int layer_index,
   float *psnr
)
```

ActiveX Visual Basic Function Prototype

```
get_output_j2k_psnr_estimate(
   layer_index as long,
```

```
   psnr as float,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`layer_index` — The layer index. Use 0 for the first layer, N-1 for the Nth layer. Use `AW_J2K_GLOBAL_LAYER_INDEX` to get the estimated pSNR for the entire image. For images with more than one component, the estimated pSNR is the average pSNR of the specified layer, over all color channels.

`psnr` — Pointer to return the estimated pSNR value.

6.3.24 aw_j2k_set_output_j2k_header_option

This function allows the user to specify whether the main image header markers and end of codestream (EOC) marker will be included in the J2K codestream. This is needed for the incremental creation of large JPEG2000 output images, where the function `aw_j2k_set_input_image_raw_region` is used to input regions of the source image into the library. For more details on incremental creation, as well as an example of the usage of this function, please see the discussion on page 42.

C Function Prototype

```
aw_j2k_set_output_j2k_header_option(
   aw_j2k_object *j2k_object,
   BOOL header,
   BOOL end_marker
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_header_option(
   header as long,
   end_marker as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`header` — If true, the subsequent output J2K codestream will have header at the beginning. This value is true by default. Set it to false to create a code stream without header.

`end_marker` — If true, the subsequent output J2K codestream will have j2k end of stream marker at the end. This value is true by default. Set it to false to create a code stream without end of stream marker.

6.3.25 aw_j2k_output_reset_options

Description

Resets all the JPEG2000 output options described above to their defaults values.

C Function Prototype

```
aw_j2k_output_reset_options(
   aw_j2k_object *j2k_object
)
```

ActiveX Visual Basic Function Prototype

```
output_reset_options(
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

6.3.26 aw_j2k_set_output_j2k_lambda

Description

The use of this function is deprecated.

C Function Prototype

```
aw_j2k_set_output_j2k_lambda(
   aw_j2k_object *j2k_object,
   float lambda
)
```

6.4 JPEG2000 Region of Interest Functions

The functions described in this section are used to define a Region of Interest when creating a JPEG2000 Output Image. They are only applicable when the output image type is set to JPEG2000.

The ROI is compressed at a higher fidelity than the rest of the image. The following functions allow the user to define an ROI using either a user supplied image or through geometric descriptions. The default for all these functions is NONE, i.e. no ROI is defined by default.

Usage:
The functions described in this section are used for the following operations:

1. Compression of JPEG2000 Image. These functions are used to define Region of Interest in the compressed JPEG2000 image.

2. Reformatting of a JPEG2000 Image. These output functions can be used to add a Region of Interest to the JPEG2000 image created as a result of the reformatting operation.

6.4.1 aw_j2k_set_output_j2k_roi_from_image

Description

Sets the ROI from a second image, supplied by the user.

C Function Prototype

```
aw_j2k_set_output_j2k_roi_from_image(
   aw_j2k_object *j2k_object,
   char *image,
   unsigned long image_length
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_roi_from_image(
   image as Variant (array of bytes)
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`image` — Pointer to the ROI image buffer.

> This image must be the same size as the image to be compressed. This image can be in any of the supported files formats (JPEG2000, JPG, PPM, PGX, PGM, BMP, TGA, TIFF). The function automatically detects the file format. All non-zero pixels in the first color channel of the image are considered to be part of the ROI (zero valued pixels are assumed to non-ROI). If there are multiple channels in the image, the first channel is used as ROI.

`image_length` — Size of the ROI image buffer.

6.4.2 aw_j2k_set_output_j2k_roi_from_image_type

Description

Sets the ROI from a second bitmap image of a specific type. This function is identical to the previous function except that it allows the user to explicitly specify the input type. This function should be used in situations when the image type was not correctly detected using aw_j2k_set_output_j2k_roi_from_image (page 170).

C Function Prototype

```
aw_j2k_set_output_j2k_roi_from_image_type(
   aw_j2k_object *j2k_object,
   char *image,
   unsigned long image_length,
   int type
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_roi_from_image_type(
   image as Variant (array of bytes),
   type as enumSupportImageTypes
) as long
```

Function Parameters

j2k_object — Pointer to the aw_j2k_object.

image — Pointer to the ROI image buffer. This image must be the same size as the image to be compressed.

image_length — Size of the ROI image buffer.

type — The type of the ROI image. The following image types are supported: JPEG-2000, JPG, PPM, PGM, PGX, BMP, TGA, TIFF. The following constants are define in j2k.h:

- AW_J2K_FORMAT_TIF
- AW_J2K_FORMAT_PPM
- AW_J2K_FORMAT_PGX
- AW_J2K_FORMAT_BMP
- AW_J2K_FORMAT_TGA

- `AW_J2K_FORMAT_JPG`
- `AW_J2K_FORMAT_J2K`

If there are multiple channels in the image, the first channel is used as ROI.

6.4.3 aw_j2k_set_output_j2k_roi_from_image_raw

Description

Sets the ROI from a second image in RAW format.

C Function Prototype

```
aw_j2k_set_output_j2k_roi_from_image_raw(
   aw_j2k_object *j2k_object,
   char *image,
   unsigned long int rows,
   unsigned long int cols,
   unsigned long int nChannels,
   unsigned long int bpp,
   BOOL bInterleaved
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_roi_from_image_raw(
   image as Variant (array of bytes),
   Rows as long,
   Cols as long,
   Bpp as long,
   bInterlaced as Boolean,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`image` — The ROI image buffer.

`rows` — The number of rows in the ROI image.

`columns` — The number of columns ROI image.

`nChannels` — number of color channels in the image. If there are multiple channels in the image, the first channel is used as ROI.

`bpp` — depth of pixels in bits (sum of all channels). Supported pixel depth: 1-16 bits per channel, packed in 1 or 2 bytes.

`bInterleaved` — This parameter should be set to true if the input image is byte interleaved by channel (ex – R,G,B, R,G,B) Set it to false, if all the bytes from the first channels are in a continuous block, followed immediately by the bytes from the second channel as a continuous block, and so on.

6.4.4 aw_j2k_set_output_j2k_roi_from_file

Description

Sets the ROI from a second bitmap image file.

C Function Prototype

```
aw_j2k_set_output_j2k_roi_from_file(
   aw_j2k_object *j2k_object,
   char *filename
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_roi_from_file(
   filename as String,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`filename` — The ROI image filename.

If there are multiple channels in the image, the first channel is used as ROI.

6.4.5 aw_j2k_set_output_j2k_roi_from_file_raw

Description

Sets the ROI from an image file in RAW format.

C Function Prototype

```
aw_j2k_set_output_j2k_roi_from_file_raw(
   aw_j2k_object *j2k_object,
   char *filename,
   unsigned long int rows,
   unsigned long int cols,
   unsigned long int nChannels,
   unsigned long int bpp,
   BOOL bInterleaved
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_roi_from_file_raw(
   filename as String,
   rows as long,
   cols as long,
   bpp as long,
   bInterleaved as Boolean,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`filename` — The ROI image filename.

`rows` — The number of rows in the ROI image.

`columns` — The number of columns ROI image.

`nChannels` — number of color channels in the ROI image. If there are multiple channels in the image, the first channel is used as ROI.

`bpp` — depth of pixels in bits (sum of all channels). Supported pixel depth: 1-16 bits per channel, packed in 1 or 2 bytes.

`bInterleaved` — flag indicating whether the ROI image is interleaved.

6.4.6 aw_j2k_set_output_j2k_roi_add_shape

Description

Adds a geometric shape to the region of interest.

C Function Prototype

```
aw_j2k_set_output_j2k_roi_add_shape(
   aw_j2k_object *j2k_object,
   int shape,
   int x0,
   int y0,
   int width,
   int height
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_roi_add_shape(
   shape as enumROIShapeDefinitions,
   x as long,
   y as long,
   width as long,
   height as long,
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`shape` — The ROI shape, defined as:

- `AW_ROISHAPE_RECTANGLE` – 0
- `AW_ROISHAPE_ELLIPSE` – 1

`x0` — The column coordinate of the upper left pixel of the ROI bounding box.

`y0` — The row coordinate of the upper left pixel of the ROI bounding box.

`width` — The width of the ROI bounding box.

`height` — The height of the ROI bounding box.

6.4.7 aw_j2k_set_output_j2k_clear_roi

Description

Clears all ROI information set using functions described above.

C Function Prototype

```
aw_j2k_set_output_j2k_clear_roi(
   aw_j2k_object *j2k_object
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_clear_roi(
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

6.5 JP2 Output Functions

The JPEG2000 standard allows the user to add unstructured data, in the format of a comment, into the J2K codestream (using `aw_j2k_set_output_j2k_add_comment` described on page 116). The JPEG 2000 file format (also known as JP2) expands on this capability, by allowing the embedding of structured data in various types of boxes.

The functions in this section enable the addition of several types of metadata boxes into a JP2 image. The following types of metadata boxes can be encapsulated into JP2:

- Intellectual Property Box, which contains intellectual property information. The format specification of the data is reserved for ISO. It will be up to the user to format the data and the SDK will not do format checking.
- XML Box, which contains text data in XML format. It will be up to the user to format the data and the SDK will not do format checking.
- UUID Box, which contains a 16-byte UUID followed by optional data. The format of the UUID is specified by ISO/IEC 11578:1996. The SDK will check that UUID is 16 bytes long, but formating or format checking beyond that would be impractical.
- UUID Information BOX, a superbox which contains two different sub boxes: a UUID List box and a URL box. The URL points to a location where information regarding the UUIDs in the list may be found.

The JP2 format also adds support for including color space information with the image. The color space may be specified in one of two ways: by enumeration, or by inclusion of a Restricted ICC Profile. Either type of specification may be used during the creation of a JP2 image; only a single color space will be added.

If no colorspace information is specified, the image is assumed to be grayscale if the image has single channel. Otherwise the image is assumed to be in the sRGB color space.

To extract metadata from a JP2 image, see the function in the JP2 Input Information Functions section, on page 85.

6.5.1 aw_j2k_set_output_enum_colorspace

Description

This function is used to set the color space of the output image. The color space is selected from an enumerated list.

If the input image has color space information (as would a JP2 image), then the output image will be transformed to the color space given to this function. If the input image does not have color space information, no color space transformation will take place.

This function also embeds the color space information into the output image, if the output image format is capable of storing it. Only a single color space can be embedded into an image.

C Function Prototype

```
aw_j2k_set_output_enum_colorspace(
   aw_j2k_object *j2k_object,
   long int colorspace
)
```

ActiveX Visual Basic Function Prototype

```
set_output_enum_colorspace(
   colorspace as enumColorspaceType
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`colorspace` — Color space designation. The input image, if it is in a different color space, will be transformed to this color space. This color space will also be embedded in the output image. It can be one of the following values:

- `AW_JP2_CS_GREYSCALE` — Single channel luminence data, which should be interpreted relative to the reference conditions in Section 2 of IEC 61966-2-1.
- `AW_JP2_CS_sRGB` — sRGB as defined by IEC 61966-2-1.
- `AW_JP2_CS_sYCC` — sYCC as defined by IEC 61966-2-1 Amendment 1.
- `AW_JP2_CS_DEFAULT` — sRGB if the image has multiple channels, and Greyscale if the image has single channel.

6.5.2 aw_j2k_set_output_jp2_restricted_ICCProfile

Description

This function is used to insert a restricted ICC color space profile into the output JP2 image. Unlike `aw_j2k_set_output_enum_colorspace` (see page 180), this function does not perform a color rotation on the input image.

C Function Prototype

```
aw_j2k_set_output_jp2_restricted_ICCProfile(
   aw_j2k_object *j2k_object,
   unsigned char *ICC_buffer,
   unsigned long int icc_buffer_length
)
```

ActiveX Visual Basic Function Prototype

```
set_output_jp2_restricted_ICCProfile(
   ICC_buffer as Variant(array of bytes),
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`ICC_buffer` — Pointer to the Restricted ICC color space profile to embed. The user is responsible for properly formatting this data. The library will not check that it constitutes a valid Restricted ICC Profile. The format of the data is specified in the ICC Profile Format Specification, version ICC.1:1998-09.

`bufferlength` — The length of `ICC_buffer` in bytes.

6.5.3 aw_j2k_set_output_jp2_add_metadata_box

Description

This is the primary function used to insert metadata into a JP2 file. This function can be used with the following box types:

- Intellectual Property Boxes, for adding intellectual property rights information;
- XML Boxes, for adding arbitrary information in XML format; and
- UUID Boxes, for adding arbitrary information in other formats.

For Intellectual Property boxes and XML boxes, only the box type and the data are required, as the format of the data is implied by the type of the box. For UUID Boxes, both the data and a UUID are required; the format of the UUID is specified by ISO/IEC 11578:1996, and its value specifies the format of the data.

C Function Prototype

```
aw_j2k_set_output_jp2_add_metadata_box(
   aw_j2k_object *j2k_object,
   unsigned long int box_type,
   char *data,
   unsigned long int data_length,
   unsigned char *uuid,
   unsigned long int uuid_length
)
```

ActiveX Visual Basic Function Prototype

```
set_output_jp2_add_metadata_box(
   box_type as enumMetadataType,
   data as Variant(Array of Bytes),
   uuid as Variant(Array of Bytes)
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`type` — The type of Metadata to be added, which can be one of the following:

- `AW_JP2_MDTYPE_JP2I` – For Intellectual Property boxes.
- `AW_JP2_MDTYPE_XML` – For XML boxes.

- `AW_JP2_MDTYPE_UUID` – For UUID boxes.

`data` — Buffer to the data to be inserted into the JP2 file

`data_length` — Length of the data in number of bytes

`uuid` — Pointer to a 16 byte UUID. This field is used only if the type is `AW_JP2_MDTYPE_UUID`.

`uuid_length` — Length of the UUID buffer, which must be 16 bytes. This field is used only if the type is `AW_JP2_MDTYPE_UUID`.

6.5.4 aw_j2k_set_output_jp2_add_UUIDInfo_box

Description

This function is used to insert an UUID Info box that includes a list of UUIDs and an URL. The URL may be used by an application to find additional information associated with the UUIDs in the list.

The format of the UUID is specified by ISO/IEC 11578:1996. The format of the URL is defined by RFC 1738.

C Function Prototype

```
aw_j2k_set_output_jp2_add_UUIDInfo_box(
   aw_j2k_object *j2k_object,
   unsigned char *uuid,
   int num_uuid,
   char *url,
   unsigned long int url_length
)
```

ActiveX Visual Basic Function Prototype

```
set_output_jp2_add_UUIDInfo_box(
   uuid as Variant(array of bytes),
   url as Variant(array of bytes)
) as long
```

`j2k_object` — Pointer to the `aw_j2k_object`.

`uuid` — Pointer to a series of UUIDs. Each UUID is 16 bytes in length. No format checking beyond enforcing the length requirement is performed to validate this data.

`num_uuid` — Number of UUIDs to add. This value may range between 1 and 65535, inclusively. The library will read 16 bytes per UUID from the uuid buffer. The length of the `uuid` buffer should be 16 × `num_uuid`.

`url` — A null terminated string of UTF-8 characters containing a URL. No format checking is performed to determine if the given string is a valid URL.

`url_length` — Length of the URL in bytes.

6.6 JPEG Image Output (Compression) Functions

The function described in this section sets certain options when the output image is a JPEG image.

6.6.1 aw_j2k_set_output_jpg_options

Description

If the output format is JPEG, then use this function to set the parameters for the output jpeg file.

C Function Prototype

```
aw_j2k_set_output_jpg_options (
   aw_j2k_object *j2k_object,
   long int quality
)
```

ActiveX Visual Basic Function Prototype

```
set_output_jpg_options (
   Quality as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`Quality` — Visual quality of the output JPEG file. The higher this number, the higher the quality and the larger the size of the output image. This value ranges from one to 100, and the default is 75.

As a special case, if the quality value is set to "-1", compression will be performed using the Lossless JPEG algorithm.

Many applications do not support JPEG compression for files with more that 8 bits per pixel bit-depth or lossless JPEG.

6.7 Other Image Output Functions

The functions described in this section are used to rotate the output image and describe raw data output with more control than is possible with `aw_j2k_get_output_image_raw` (see page 98) and `aw_j2k_get_output_image_file_raw` (see page 104).

6.7.1 aw_j2k_set_output_com_geometry_rotation

Rotates the output image by an increment of 90 degrees. During a compression operation (for example, from RAW format to JPEG2000 format), the image is rotated before it is encoded using JPEG2000. During a decompression operation (for example, from JPEG2000 format to W), the image is rotated after it has been decoded into RAW data.

C Function Prototype

```
aw_j2k_set_output_com_geometry_rotation(
    aw_j2k_object *j2k_object,
    int angle
)
```

ActiveX Visual Basic Function Prototype

```
set_output_com_geometry_rotation(
   angle as long
)
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`angle` — The angle of rotation in degrees. Only increments of 90 degrees are allowed.

6.7.2 aw_j2k_set_output_raw_endianness

Sets the endianness of the output raw image data. This controls how images with more than 8 bits of data per color component are written — i. e. which of the two bytes that comprise each sample is the most significant byte and which the least. Three choices are accepted: big endian (native to Motorolla and Sparc processors); little endian (native to Intel processors); and the local machine endianness.

This function must be called before `aw_j2k_get_output_image_raw` **(see page 98) or** `aw_j2k_get_output_image_file_raw` **(see page 104) to set the endianness of an output raw image.**

C Function Prototype

```
aw_j2k_set_output_raw_endianness(
   aw_j2k_object *j2k_object,
   unsigned long int fEndianness
)
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`fEndianness` — The endianness of the raw data. The following constants are defined in j2k.h:

- `AW_J2K_ENDIAN_LOCAL_MACHINE`
- `AW_J2K_ENDIAN_SMALL`
- `AW_J2K_ENDIAN_BIG`

6.8 ActiveX Only Functions

Most of the ActiveX interface functions directly correspond to their equivalents in the C interface. The functions described in this section are unique to the ActiveX interface.

6.8.1 `get_output_vb_picture`

Description

Returns the output image as a picture object that can be used under Visual Basic.

ActiveX Function Prototype

```
get_output_vb_picture(
   Picture as IPictureDisp
) as long
```

Function Parameters

`Picture` — Output parameter to return the Visual Basic picture object

6.9 JPEG 2000 Part 2 Extensions

Part 2 of the JPEG2000 standard defines extensions to the core technology of Part 1. These extensions include generalized DC shift, tile overlapping, generalized wavelet transform kernels and decompositions, trellis coded quantization and multi-component transformations.

For images with multiple components, the multi-component transformations in Part 2 extend the simple RBG to YUV color space transformation (set by `aw_j2k_set_color_transform`, as described on page 112) to exploit the correlation of the imagery in the component direction. These transforms are applied independently to the values of each pixel in the component direction. Wavelet transforms, linear transforms and dependency transforms (such as DPCM techniques) are all allowed.

The JPEG2000 Codec by Aware includes the capability to apply the wavelet based multi-component transform, for imagery with multiple components. When the multi-component transform is invoked, the appropriate codestreams markers are included in the codestream to signal the use of the multi-component transform, as defined in Part 2. The Aware JPEG2000 Decoder is also capable of recognizing these markers and decoding the Part 2 codestream, using an inverse multi-component transform.

A single new function, `aw_j2k_set_output_j2k_wavelet_mct` allows the user to set the type of wavelet transform, and the relevant optional parameters. This function is described in detail below.

6.9.1 aw_j2k_set_output_j2k_wavelet_mct

Description

This function allows the user to set the wavelet based multi-component transform and the number of decomposition levels to be performed in the component direction. In addition, Part 2 allows for the grouping of components in collections, so the wavelet transform is applied independently to the pixel values in each component collection. This function also sets the number of components in each component collection.

The following wavelet transforms are allowed:

- The irreversible 9-7 wavelet transform for lossy compression, using a floating point implementation.
- The reversible 5-3 wavelet transform for lossless or lossy compression.

C Function Prototype

```
aw_j2k_set_output_j2k_wavelet_mct(
   aw_j2k_object *j2k_object,
   int type,
   int levels,
   int collection_size
)
```

ActiveX Visual Basic Function Prototype

```
set_output_j2k_wavelet_mct(
   type as enumWaveletTransformTypes,
   levels as long,
   collection_size as long
) as long
```

Function Parameters

`j2k_object` — Pointer to the `aw_j2k_object`.

`type` — wavelet transform selection. I9-7 uses the irreversible 9-7 wavelet filter, R5-3 uses the reversible 5-3 wavelet filter. The following constants are defined.

- `AW_J2K_WV_TYPE_I97`
- `AW_J2K_WV_TYPE_R53`

`levels` — number of decomposition levels to perform in the component direction.

By default, the number of decomposition levels is set so that the smallest resolution (along the component axis) is between 16 and 31 components, with at least 3 decomposition levels at all times. To get the default number of levels use the constant `AW_J2K_WV_DEPTH_DEFAULT`.

`collection_size` — number of components in each component collection.

By default the collection size will be the total number of components in the image (resulting in a single component collection). To get the default collection size use `AW_J2K_SELECT_ALL_CHANNELS`.

Examples

The following command line reads in and compresses a 512x512, 13 bit signed, 200 image, raw file. It sets the compression type to lossless and uses a multi-component transform with 4 levels of wavelet transform and component collections of 50 components.

```
j2kdriver --input-file-raw 200-images.raw 512 512 200 -2600 -w R53 -R 0
-t j2k -mct R53 4 50 -o mct.j2k
```

The equivalent API call for the multi-component transform is:

```
aw_j2k_set_output_j2k_wavelet_mct(j2k_object, AW_J2K_WV_TYPE_R53, 4,
50);
```

Alternatively, to use the default settings for number of levels and component collections, those fields can be omitted:

```
j2kdriver --input-file-raw 200-images.raw 512 512 200 -2600  -w R53 -R 0
-t j2k -mct R53 -o mct.j2k
```

This will result in a single component collection with 200 components and 3 levels of wavelet transform, in the component direction.

The equivalent API call for the multi-component transform is:

```
aw_j2k_set_output_j2k_wavelet_mct(j2k_object, AW_J2K_WV_TYPE_R53,
AW_J2K_WV_DEPTH_DEFAULT, AW_J2K_SELECT_ALL_CHANNELS);
```

To get the default number of levels and a collection size of 100, use:

```
j2kdriver --input-file-raw 200-images.raw 512 512 200 -2600  -w R53 -R 0
-t j2k -mct R53 -1 100 -o mct.j2k
```

The equivalent API call for the multi-component transform is:

```
aw_j2k_set_output_j2k_wavelet_mct(j2k_object, AW_J2K_WV_TYPE_R53,
AW_J2K_WV_DEPTH_DEFAULT, 100);
```

Chapter 7

Windows Tool Tutorial

This tutorial is designed to introduce the use of the complete range of capabilities of the J2K Tool and the compression and decompression capabilities of the underlying library. The user will be introduced to the tool via the applicable input imagery and then a basic compression and decompression session. This description will cover all functionality of the tool.

The source code and project file of the tool is provided under `<InstallDir>/Example Files/J2K Tool` directory.

7.1 Starting the J2K Tool

To activate the demonstration program select `Start->Programs->Aware JPEG2000 SDK->J2K Tool` The executable itself is in `<InstallDir>\Bin\J2K_Tool.exe`.

7.2 Input Images

Several image formats are available as input to the J2K Tool. To open an image file select the menu item `File->Open`. This will bring up a traditional windows file dialog box (Figure 7.1). The user may navigate to their drive, directory, and image of choice. Via the dialog pull down box Files of type the user may select from BMP, JPEG, RAW, TIF, PNM, JPEG2000 Files, All Image Files, or All Files. Each selection will modify the available files presented to the user. Select the one item that is most applicable to the current needs — each format is described, with respect to this program, in the following section.

Once an image file has been selected click on the Open button. This will open the image for use and display it in the left side image box under the Original Image title.

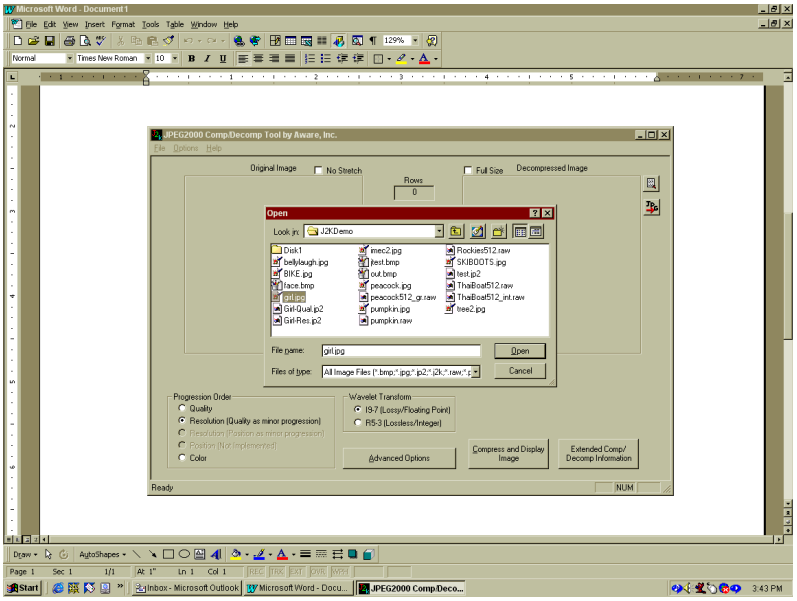

Figure 7.1: The File Open dialog box

7.2.1 BMP

The bitmap (BMP) format is a supported input for the J2K Tool, provided that the image is 24-bit RGB or 8-bit grayscale.

7.2.2 JPEG

The JPEG standard (JPEG2000 will be discussed in a separate section) is supported in 8-bit grayscale, 12-bit grayscale and 24-bit RGB color formats. When a JPEG image is used as an original image, the compression ratio is computed based on the image size after the JPEG image has been decompressed. Lossless JPEG format is also supported.

7.2.3 PNM

This primarily UNIX supported format is available as an input to the J2K Tool. All PNM formats are supported in both raw and ASCII format.

7.2.4 TGA

Targa files are supported provided that the image is 24-bit RGB or 8-bit grayscale.

7.2.5 TIF

Uncompressed TIF files are supported by the J2K Tool.

7.2.6 Raw

The user may input a RAW image under the condition that they know the number of rows, columns and channels of the input image along with the bits per pixel and the band interleaving. The user may also identify the number of header bytes within the image. Once a RAW image is selected the user is presented with a dialog box where these parameters are input.

7.2.7 JPEG2000

The JPEG2000 standard is fully supported with this program. JPEG 2000 images with a .j2k extension can be used as input image to the tool. In addition, images that use the optional JP2 file format, with a .jp2 extension can be used as input images. Note that repetitive compressions on the same image may lead to undesired image artifacts.

Although this tool does allow the user to input JPEG2000 images and recompress them, this version of the tool does not perform reformatting of JPEG2000 images. JPEG2000 images are decompressed completely when loaded into the application and subsequently recompressed when the Compress and Display Image button is clicked. As for JPEG images, the compression ratio is computed based on the image size after the image has been decompressed.

7.3 Open Image

Once the user has selected the directory, path, and image filename via the `File->Open` dialog box the Open button must be clicked. This action reads the image file from disk and displays it in the left side image box under the Original Image title (Figure 7.2). The image will be stretched along its longest side to fit within the 256x256-display area. Other display options will be discussed in later sections.

The opened input image's Rows, Columns, Channels and Bits/Pixel will be displayed down the center of the dialog box for reference. These are read only values and may not be modified. The image is now available for compression given the selected compression parameter set.

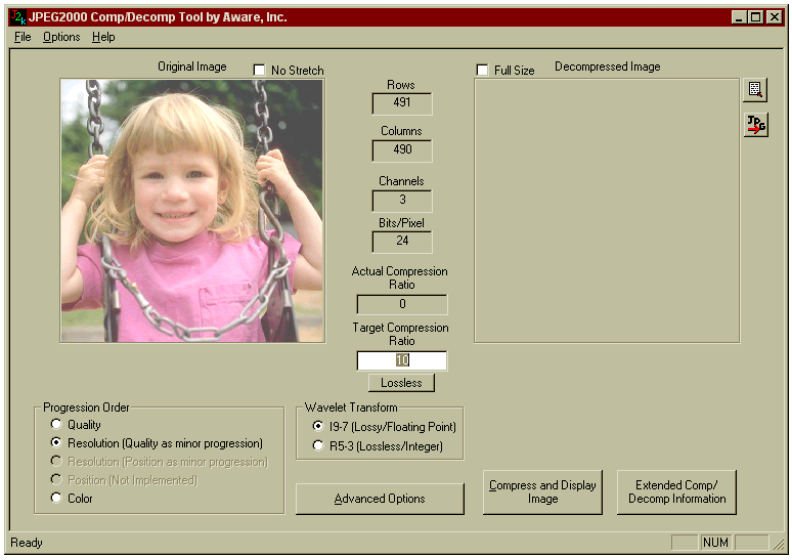

Figure 7.2: Main program window with an uncompressed image loaded from disk

7.4 Basic Compression and Decompression

The basic compression and decompression session begins with the identification of the input image. As described, this will result in the display of the image in the left side image frame. At this time the user will then accept all compression defaults. A full discussion of all available compression parameters is available in later sections.

The compression of the image is accomplished via clicking the `Compress and Display Image` button. The program then compresses the image, i.e. converts it to the JPEG2000 format with the current parameter set, and then decompresses and displays it in the right side image frame (Figure 7.3). The user may then change the displayed level of the image. Finally, the user may then save the JPEG2000 image to a file or save the current decompressed version as a bitmap. Each of these processes will now be discussed in detail.

7.4.1 Default Compression Settings

The default compression settings for the J2K Tool are designed for a balance of compression and quality, as well as for usability of the selected progression. The main window includes three compression parameters: Target Compression Ratio, Progression Order, and Wavelet Transform. Further compression options are available via the Advanced Options button.

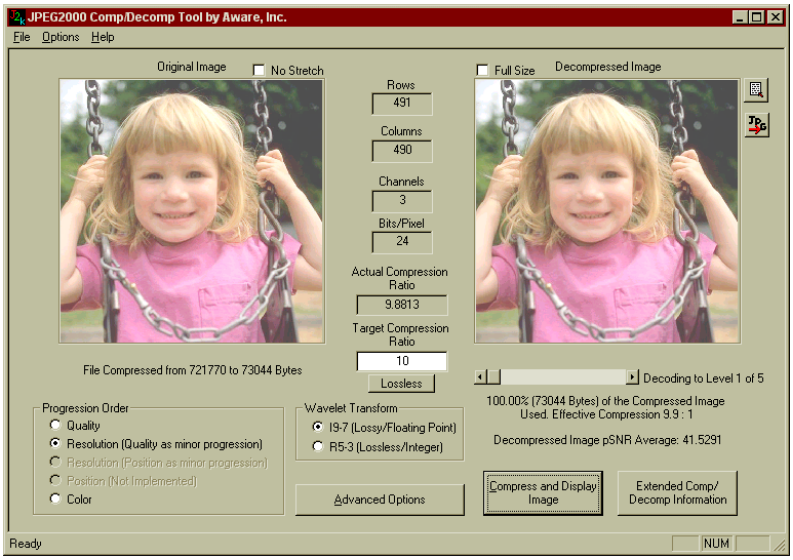

Figure 7.3: Main program window with original image and the decompressed image. Note the additional displayed information.

7.4.2 Compression

The conversion from input image to the JPEG2000 format will be generally discussed as a compression process. This process can be achieved with the Compress and Display Image button. This button performs compression of the image followed by a decompression to create the image shown on the right panel. The J2K Tool also calculates several error metrics that serve to demonstrate the differences between the original image and the compressed/decompressed version.

As the compression is completed the user is informed of the current status and the time required for compression. Finally, the compressed image is displayed in the right panel beneath the Decompressed Image title. Likewise, the compressed image is stored internally and can now be viewed with respect to all of its levels and stored in an external file.

7.4.3 Decompressed Image Representation

Since the compression process can result in information loss, the decompressed image may not be identical to the original input image. The amount of loss is driven by the choice of the compression ratio. The higher the ratio the higher the potential for loss. Therefore, the decompressed image is represented in the right side frame. The user may then judge the quality of the compression and adjust the appropriate parameters. There are several options with respect to the display of both the original

and decompressed images as discussed in the next section.

7.4.4 Display Options

There are several options for the display of the J2K Tool images. The first is the default for the application: Stretch to Fit. The second displays the image within the main window and does not apply a stretch to the image. The last display method uses unstretched images in external windows. Finally, the user may generate a JPEG image in an external window for comparison to the current JPEG2000 decompressed image.

Stretch to Fit

The default display option is the stretching of the input image and the compressed/-decompressed image to fit within the 256x256 frames provided. Although not ideal for image comparison this default setting allows for a comparison of the entire image area.

No Stretch

The No Stretch check box will display both images within the main window without any aspect ratio stretch applied (Figure 7.4). When clicked, scroll bars will become visible and the user may move around the image area. When the scroll bars are moved for one image the result is applied to the second.

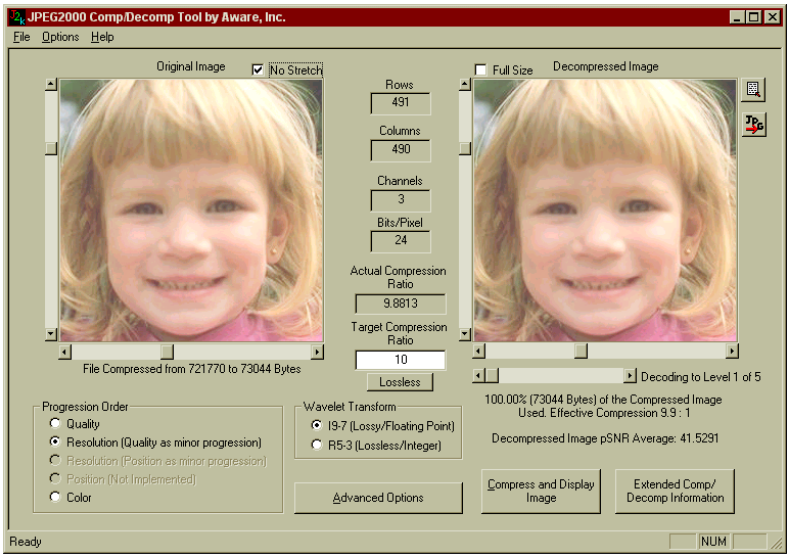

Figure 7.4: Main program window with the No Stretch option applied.

External Display

The menu item `Options->External Display` will spawn an external window for both the original image and the decompressed image (if it exists). If no images are currently available then the setting will become active with the first image opened.

Each image will be identified by its title (Figure 7.5). Other supporting information may also be provided in the title. Each image will also be presented with no zoom or stretch applied. These windows will remain intact until the user closes them. Once the original image has been closed the user re-click on the `Options->External Display` menu item to re-display the original image. Finally, as the user cycles though the compression levels, a new window will be generated for each accessed level. This is made available for side by side comparisons of several levels of compression.

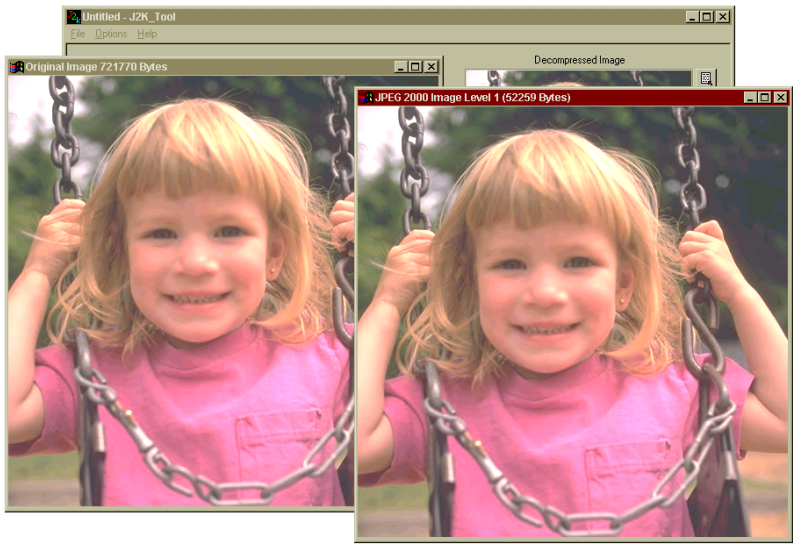

Figure 7.5: Main program window with original and decompressed images using "External Display"

7.4.5 Compare to JPEG

The JPG button to the right of the decompressed J2K image is designed to present a comparison between the currently decompressed image and a comparable JPEG image. The JPEG image is generated via a loop that finds the closest match in file size of the JPEG image to the current J2K decompressed image. Due to the design of the loop the JPEG image will always be larger than the J2K image. The resulting JPEG will be spawned to an external window (Figure 7.6). If the user is currently viewing the original and decompressed images within the main frame and not externally then the decompressed

image will also be spawned externally.

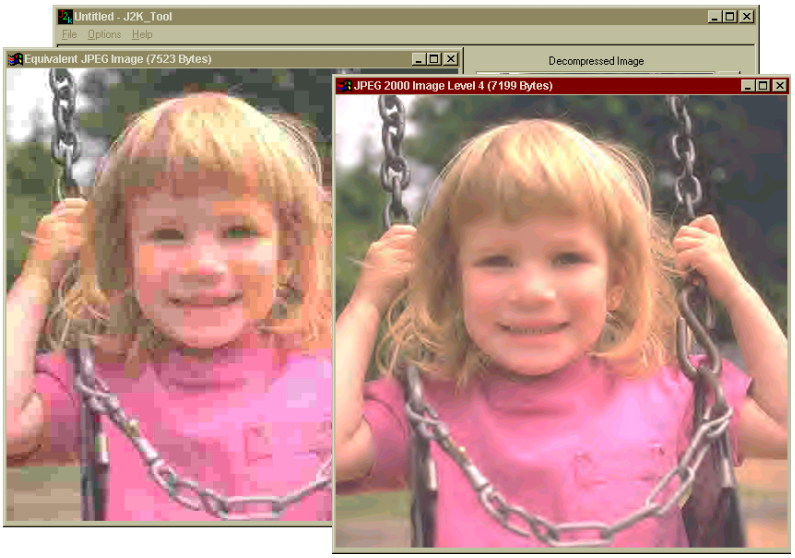

Figure 7.6: "External" display of the decompressed JPEG2000 (right) image and the corresponding JPEG image (left)

7.4.6 Full Size

The full size option applies to the decompressed JPEG2000 image only. Selecting this check box will force a reduced resolution decompressed image to be the same size as the original. The upsampling is carried out in the wavelet domain during the inverse wavelet transform portion of the decompression.

7.4.7 Compression Ratio and Decompressed Bytes

Once the compressed image has been generated, decompressed, and displayed the user will be presented with additional compression information. Beneath the original the user is presented with the original image size (rows x columns x channels x bits) and the compressed image size. This text will resemble the statement: "File Compressed from 786432 to 232984 Bytes." Likewise, the user will be presented with the Actual Compression Ratio. This can be compared to the Target Compression Ratio input parameter. Modifications to this display can be made via `Options->Target Compression Ratio` and `Options->Target Compression File Size`. By toggling these parameters the user can see or set the actual or target compression as a ratio or as the size of the output file in bytes.

7.4.8 Levels

The JPEG2000 standard supports the storage of multiple levels of image information within the file. The concept of levels expresses itself in the ability of the user to decompress a portion of the compressed image and be presented with different sized or different quality versions of the original image. This enables transmission and display of images that utilize less bandwidth and fewer bytes than the compressed file itself. The specific definition of the progression order will be covered in a later section, titled Progression Order. The number of levels within the current image is represented with a new scroll bar below the decompressed image. This scroll bar allows the user to toggle through the available levels. The current and total levels are presented to the right of the scroll bar. As the user progresses through the levels the decompressed image is updated to reflect the current decompressed version of the compressed image. Below the levels scroll bar the application indicates the number of bytes of the compressed image that were used to construct the current decompressed image. This amount of utilized/transmitted data can also be expressed as the effective compression ratio when compared to the number of bytes in the original image. Effective compression is presented below the levels scroll bar.

7.4.9 pSNR

The J2K Tool provides several error metrics for a simple comparison between the current decompressed image and the original image. The peak signal to noise ratio is presented in the statement "Decompressed Image pSNR Average: 52.3587." If the image is RGB then this is the average pSNR for all three colors. If the image is grayscale then value is the pSNR for the entire image.

7.4.10 Extended Compression/Decompression Information

To access additional information regarding the current compression/ decompression the user may press the Extended Comp/Decomp Information button. This button will bring up a new window that provides further information (Figure 7.7). The first set of data, Buffer Sizes, displays the size of the original, compressed, and decompressed buffers. The Processing Times section identifies the time (in whole seconds) required for compressing and decompressing the image. The Compression Ratios are a repeat of the overall and effective compression ratio for the current decompression level. The Compression Levels section reiterates the total number of levels for the current image. Quality Metrics express, on a per channel basis, the peak Signal to Noise Ratio (pSNR), Mean Square Error (MSE), and the percentage of the two image's pixels that are identical (% Same). For each of these metrics, an average is presented. If the input image is grayscale then the values are presented in the Red/Gray column.

The user may exit this window with the Close button.

7.4.11 Comments

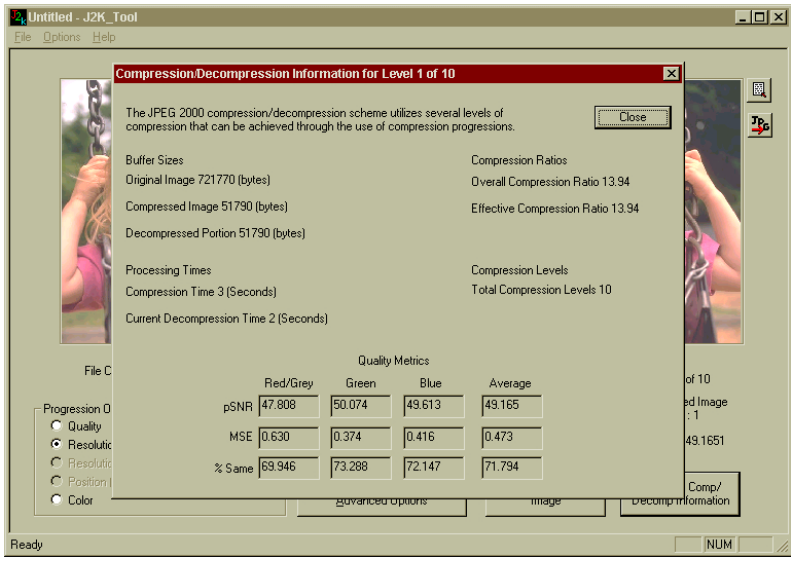

Figure 7.7: The Compression Decompression Information window

The JPEG2000 standard allows for the inclusion of ASCII or binary comments. The comment button will open a new window that provides access to comments if they are available within the image (Figure 7.8).

If the comment is of ASCII format it will be displayed in the edit box. If the comment is binary then the edit box will display a message to that affect. Finally, If there is no comment available the user will also be informed of this. The user may save the comments to a file via the Save Comment button. This button will bring up the standard Save As dialog. The user may then identify the drive, directory and filename they wish to save the comments in. In the JPEG 2000 In Line Comment window click OK to exit. Refer to the Advanced Options: Comments section for further information.

7.4.12 JPEG2000 File Save

Once an input image has been compressed with a particular parameter set the user may save it via `File->Save Compressed Image`. The output filename will automatically be appended with a j2k extension.

7.4.13 Decompressed Image Save

In addition to saving the compressed JPEG2000 image, the user has the ability to save the decompressed version in BMP format. This allows for the storage and comparison

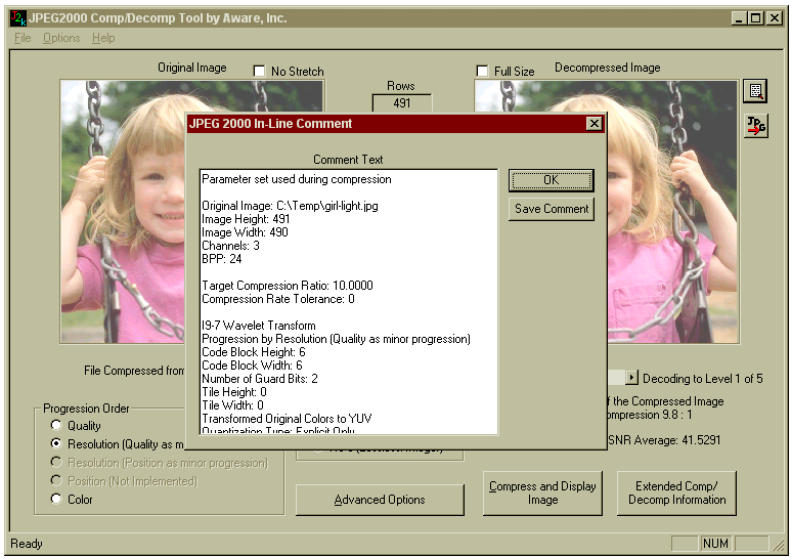

Figure 7.8: The In-Line Comment dialog box

of the decompressed version in the event that a JPEG2000 image ready program is not available. Via `File->SaveDecompressed Image as Bitmap` the user can save this file.

7.5 Basic Compression Parameters

There are many parameters that drive and shape the JPEG2000 compression process. For the purposes of the J2K Tool they are broken into two sections: basic and advanced. The basic parameter set is presented in the main J2K Tool window. These four parameters (target compression, progression order, wavelet transform, and ROI) will have the most significant impact on the compression process. For the most part the default settings for these will suit most compression needs, however they may be changed to any valid value to meet specific compression requirements.

Developers Note: Each parameter discussed will be identified with respect to the corresponding API function call.

7.5.1 Target Compression Ratio/File Size

The target compression ratio and file size provides the compression utility with a goal for compression. Upon completion the actual compression ratio should approximate the target ratio. The user may select either a target via ratio or via file size. The control of this setting is accomplished with the `Options->Target Compression Ratio`

and `Options->Target Compressed File Size` menu items. These two options act as toggles in that only one can be selected at any given time. As the user flips between the two, the Actual and Target compression boxes change value to reflect the setting. Regardless of the type of setting, the Aware, Inc. JPEG2000 Compression/Decompression Library will attempt to match the target. Its ability to match the target will be affected by the advanced option Quantization Type setting (see Advanced Options section).

Note: Lossless compression is available with the R5-3 transform and a target compression ratio of 0.0.

Developers Note: For functions that control the compression ratio, file size and bitrate, please see the following functions:

- `aw_j2k_set_output_j2k_ratio` (page 107)
- `aw_j2k_set_output_j2k_filesize` (page 108)
- `aw_j2k_set_output_j2k_bitrate` (page 106)
- `aw_j2k_set_output_j2k_tolerance` (page 111)

7.5.2 Progression Order

The JPEG2000 file structure is designed so that it can be progressively decoded, or truncated, at specific file locations and still produce a valid image output. The organization of the file that allows for this ability is termed the progression order. Specifically, sections of data are stored sequentially within the file and each portion contributes to the whole image. The progression type determines the nature of that contribution to the whole. In turn the decompressor must read only a portion of the compressed image to reduce both processing and memory requirements. The tradeoff is a reduction in image quality. The specific progression orders are as follows:

- The **Quality**-major orders store the image data in order of error. The more data sections that are read the lower the decompressed image error.
- The **Resolution**-major orders achieve higher compression ratios by leaving only lower resolution versions of the image in the compressed file.
- The **Color**-major orders store the compressed image by color channel. Here the lowest decompressed image level for an RGB image will be grayscale if the Transform to YUV option is used, else the output will be only the red channel.

Note: the progressions involving Position are not currently implemented and are disabled in this application.

Developers Note: The progression order is set by `aw_j2k_set_output_j2k_progression_order` (page 113).

7.5.3 Wavelet Transform

The Wavelet Transform option specifies the wavelet filter that is applied to the image. The irreversible 9-7 (I9-7) option provides high quality lossy compressed images at the expense of more processing needed to generate the compressed file. The reversible 5-3 (R5-3) option can provide lossless compression by setting the target compression ratio to 0.0 (the Lossless button will automatically configure the system for lossless encoding). Lossy compression can also be performed using the R5-3, by setting the compression ratio to the desired value.

Developers Note: The wavelet transform options are set by `aw_j2k_set_output_j2k_xform` (page 123).

7.5.4 Region of Interest Mask

The ability to define a region of interest (ROI) is a unique JPEG2000 feature. The definition of an ROI instructs the library to compress the area within the ROI with higher fidelity than the data outside the ROI. The menu item `Options->Generate ROI` spawns a new viewer with the original image and the tools to draw an ROI (Figure 7.9). The user may click on either the rectangle tool or the ellipse tool and click and drag small ROI on the image. These areas will be compressed a higher quality levels than those shaded in red. The Clear button will remove the current ROI and the user may then start the definition process over. OK will complete the session while Cancel will close the session without making changes to the ROI.

Once the ROI is generated, a new ROI check box will appear above the original image. Click this box to display the current ROI (Figure 7.10). If the user wishes to remove the ROI then select `Options->Generate ROI`, `Clear`, and `OK` to delete the image ROI.

Developers Note: The ROI can be set using the functions described in Section 6.4 titled JPEG2000 Region of Interest Functions.

7.6 Advanced Compression Options

The Advanced Options button provides access to several options that control various specific aspects of the compression process. Each of these parameters will be discussed in the following sections. The access is provided via a dialog box, shown in Figure 7.11, that is accessible through the `Advanced Options` button in the main window.

7.6.1 Transform to YUV

The Transform Image to YUV option will convert an RGB image to a YUV space prior to compression. Due to its nature, the YUV color space yields greater compression ratios with higher image quality. By default this option is turned on. This option is ignored for grayscale images.

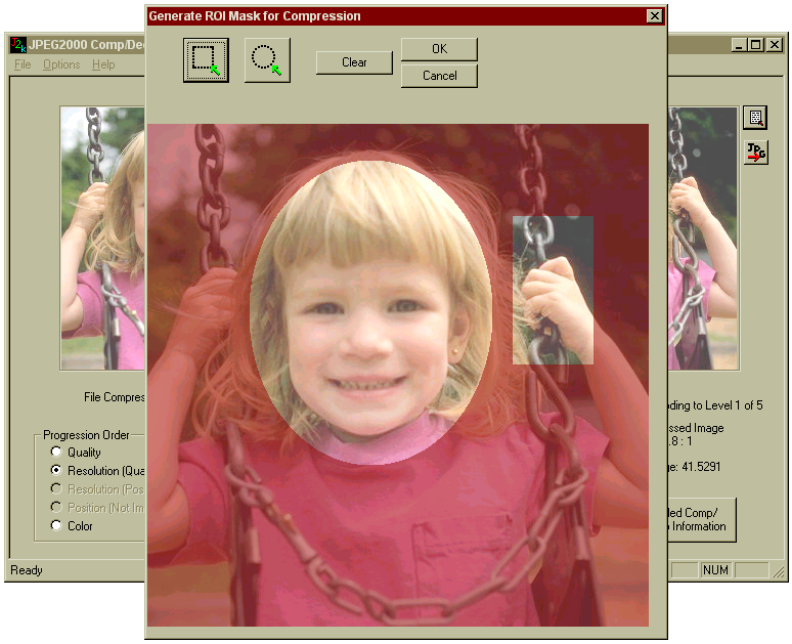

Figure 7.9: The Generate ROI Mask dialog window with two areas (face, hand) hilighted as an ROI

Developers Note: The color transform flag is set by `aw_j2k_set_output_j2k_color_xform` (page 112).

7.6.2 Use Tiles (Tile Height and Width)

The Use Tiles option instructs the library to divide the image into tiles that will each be separately compressed. This will significantly reduce the memory usage while running the program but may also introduce artifacts at the edge of the tiles, especially as the size of the tiles is decreased. However, using small tiles can introduce tiling artifacts at the boundaries of the tiles.

Clicking on the Use Tiles option will allow access to the Output Tile Height and Output Tile Width values. This option is turned off by default. Set the tile size to 0 to disable this option

Developers Note: The tile dimensions are set by `aw_j2k_set_output_j2k_tile_size` (page 115).

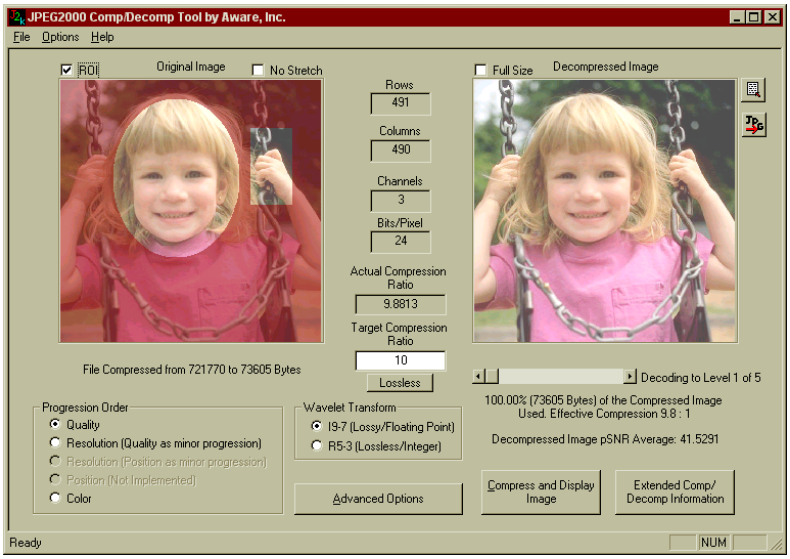

Figure 7.10: The uncompressed input image with an active ROI

7.6.3 Automatic Level Selection (Transform Levels)

The Transform Levels option specifies the number of resolution levels that the compressed file will have. Each resolution levels is one half the size in the X and Y directions with one quarter the number of pixels from its predecessor level, i.e. the image size is down sampled by a factor of two. The number of transform levels is equivalent to the number of subresolution level images available in the resulting JPEG2000 image. For example, setting the number of resolution levels to 2 would create a JPEG2000 image with 3 resolution level: full resolution, half resolution and quarter resolution.

Selecting the Automatic Level Selection allows the library to select the appropriate number of transform levels. Automatic Level Selection is the current default setting. Set this value to -1 for automatic calculation of the number of levels.

Developers Note: The number of levels is set by `aw_j2k_set_output_j2k_xform` (page 123).

7.6.4 Guard Bits

The Guard Bits option specifies the number of extra bits to consider when encoding the image. This option prevents coding overflow. The default value is 2. In the event of a compression error "23" this value should be increased. Accepted values for this parameter are 0–7. For more details on this option, see page 147.

Developers Note: The number of guard bits is set using `aw_j2k_set_output_j2k_-`

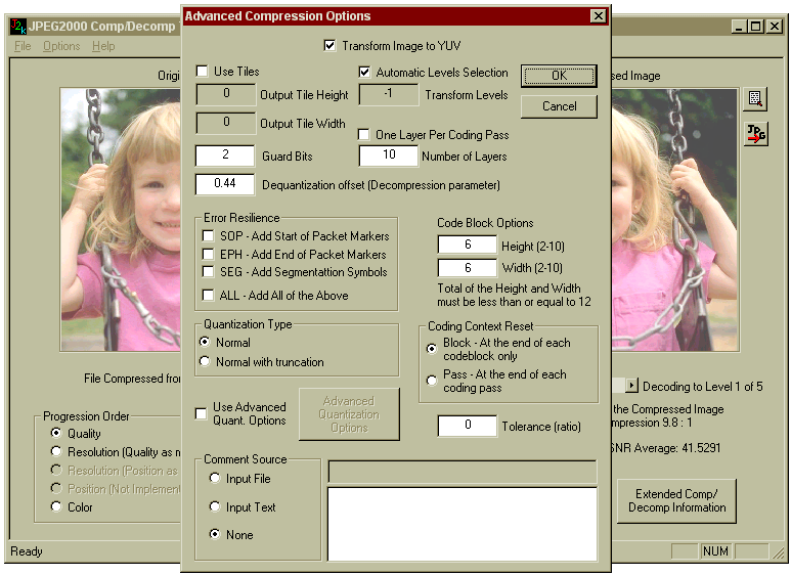

Figure 7.11: The Advanced Options dialog box

`guard_bits` (page 147).

7.6.5 Layers per Coding Pass

The Number of Layers option controls the number of quality levels that are present in the file. When used with quality-major progression the user can directly select the overall number of layers. Higher values will lead to slightly larger file sizes. The One Layer Per Coding Pass option will automatically determine the appropriate amount for other progressions. The default value for this parameter is 10. Use a value of 1 with a Quality progression for the largest compression ratio.

Developers Note: The number of layers is set by `aw_j2k_set_output_j2k_layers` (page 125).

7.6.6 Error Resilience

The Error Resilience options specify the addition of redundant data to the code stream. At the expense of making the compressed file larger, markers are added at specific places (Start of Packet, End of Packet Header, etc) to aid in detection of file corruption during transmission. All markers are turned off as the default.

Developers Note: The error resilience options are set by `aw_j2k_set_output_j2k_error_resilience` (page 121).

7.6.7 Code Block Height and Width

These options control the size of the post wavelet-transform code blocks. After transformation, each block is encoded separately. Larger code blocks result in a smaller file size at the cost of higher memory consumption. Values may range from 2 to 10 and the sum cannot exceed 12. The default values are 6 and 6.

Developers Note: The code block dimensions are set by `aw_j2k_set_output_j2k_codeblock_size` (page 152).

7.6.8 Coding Context Reset

Transmission errors in encoded data will generally make data unreadable from the error to the end of the encoded section. The Coding Context Reset option controls where the end of the encoded section occurs, and therefore how much data will potentially be lost due to transmission errors. The reset can happen at the end of each code block, which makes for smaller compressed files that are more susceptible to errors, and at the end of each coding pass of a code block, which makes for larger files that are less susceptible to errors.

Developers Note: The coding context reset options are set by `aw_j2k_set_output_j2k_coding_style` (page 157).

7.6.9 Quantization Type

The Quantization Type option controls the method by which the compressor tries to achieve the target compression ratio.

Normal : The image is compressed, but the compressed file is not truncated. Produces the highest quality images, with the highest variability of rate-control (i.e. how close is the achieved compression ratio to the target compression ratio).

Normal With Truncation : The image is compressed, and if necessary truncated to produce compressed files closer to the target compression ratio. Tends to produce lower quality image than the previous option.

Developers Note: The quantization options are set by `aw_j2k_set_output_j2k_quant_options` (page 143).

7.6.10 Use Advanced Quantization Options

The Advanced Quantization Options dialog box (seen in figure 7.12) is enabled when the Use Advanced Quant. Options check box is selected and the Advanced Quantization Options button is pressed. This new series of quantization options can be selected on a per-channel basis. Selections made for channel 1 will be automatically propagated to channels 2 and 3. Care should be taken when mixing options across the three channels. For more details on these options, see the description of `aw_j2k_set_output_j2k_quant_options` (page 143).

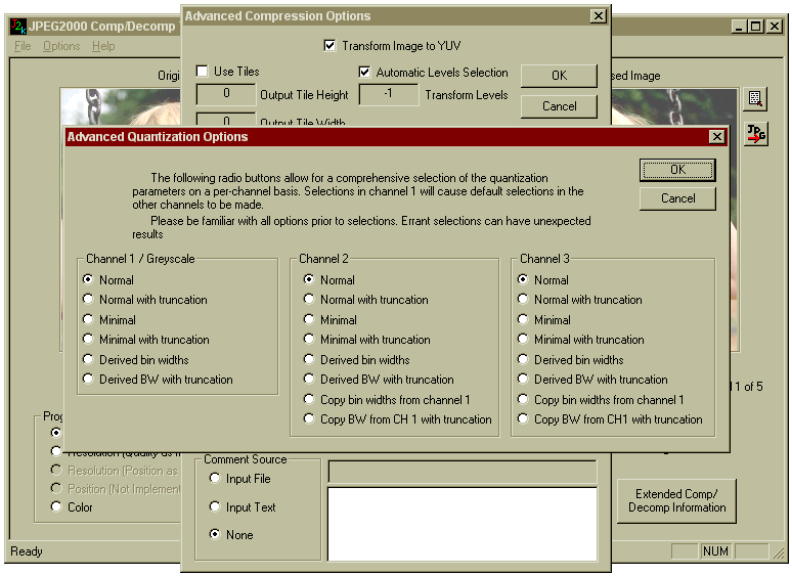

Figure 7.12: The Advanced Quantization Options dialog box

Developers Note: The quantization options are set by `aw_j2k_set_output_j2k_quant_options` (page 143).

7.6.11 Comments

The JPEG2000 standard allows for the inclusion of comment data into the compressed file. There are two selections for including comments: ASCII and binary. The Comment Source option group provides for these selections.

The Input Text option allows the user to enter text into a large edit box. By default this edit box is populated with information about the parameters used during the compression process.

Note: To get the most complete and up to date parameter set the user should exit and reenter the Advanced Options dialog then set the comments to Input Text.

The Input File option grants access to a file open dialog in which the user will identify a binary, or ASCII file to be inserted into the compressed file. The filename will be reflected in the small edit box.

Developers Note: Comments are set by `aw_j2k_set_output_j2k_add_comment` (page 116).

Chapter 8

Windows J2K Viewer

This section describes the use and capabilities of the J2K Viewer Tool. The J2K Viewer is designed to demonstrate the capabilities of the Aware JPEG2000 SDK for compressing, decompressing and interactively viewing large images. In addition, the J2K Viewer demonstrates the SDK's ability to handle input and output images that are larger than the size of the system's physical memory.

The source code and project file of the tool is provided under `<InstallDir>/Example Files/J2KViewer` directory.

8.1 Starting the J2K Viewer

To activate the J2K Viewer program select `Start->Programs->Aware JPEG2000 SDK->J2K Viewer`. The executable itself is in `<InstallDir>\Bin\J2KViewer.exe`.

8.2 Using the J2K Viewer Tool

The J2K Viewer can read input images in the following formats: J2K, JP2, uncompressed TIF and BMP. The J2K Viewer is designed to demonstrate how the Aware JPEG2000 SDK can be used to interactively view very large JPEG2000 images. For this reason, if the input image is either TIF or BMP, it is first compressed into J2K format. For maximum efficiency and usability in this application, the JPEG2000 image should be compressed using tiles and multiple resolution levels and quality layers. For more details on the compression operation, see Sec. 8.2.5. After compressing the image, the compressed file is saved to disk and then loaded into the J2K Viewer for decompression and viewing. If the input image is originally in J2K or JP2 format, then the application will immediately load the image for decompression and viewing.

8.2.1 Decompression and Viewing

To open a J2K file select the menu item `File->Open J2K File`. This will bring up a traditional windows file dialog box . The user may navigate to the drive, directory, and image of choice.

When the application first opens a J2K file, three windows are opened:

1. Main Window - displays the image. When the image is first loaded, the application displays the smallest resolution level that will fit inside the viewing window, at the lowest available quality level.
2. Navigation Window - displays a sub-resolution version of the entire image. It includes a red rectangle outlining the subsection of the image that is currently displayed in the Main Window.
3. Quality Control Window - controls the number of quality layers that are being decoded for display in the Main Window.

To view additional information about the J2K image being viewed select the menu item `File->Properties`.

8.2.2 Panning

There are 3 ways to pan and view different sections of the image:

1. Press and hold down the middle mouse button to activate "Pan" mode on the main window.
2. Use scroll bars available on the main window.
3. Use the navigation window. Left clicking on the navigation window will cause the main window image display to re-center to the location of the mouse click.

Note that the current zoom level and quality level are displayed on the left side of the lower window border of the Main Window. In addition, the number of tiles that were decoded during the most recent decoding operation and the decoding time (in seconds) is displayed in the middle of the lower border of the Main Window.

Developers Note: Panning is enabled using the region decoding capabilities of the library. For more details see the `aw_j2k_set_input_j2k_region_level` function (page 80).

8.2.3 Zooming

Zooming is performed using the left and right mouse buttons, as follows:

- To Zoom In, left click on the main window.
- To Zoom Out, right click on the main window.

Note that the application uses the resolutions available in the JPEG2000 image to zoom in and out of the image. No pixel interpolation or decimation is performed. Therefore, the maximum and minimum zoom levels are dictated by the image.

Developers Note: The decompression resolution level is set by the `aw_j2k_set_output_j2k_resolution_level` function (page 77).

8.2.4 Quality Control

The decoded image quality level is controlled using the buttons on the Quality Control Window. Clicking on the `+1` and `+7` buttons increases the number of quality layers that are being decoded by 1 and 7 layers. Similarly, the `-1` and `-7` buttons reduce the quality layers by 1 and 7 layers.

Developers Note: The decompression quality level is set by the `aw_j2k_set_input_j2k_quality_level` function (page 78).

8.2.5 Compression

To open a TIF or BMP image, select the menu item `File->Open TIF/BMP File`. This will bring up a traditional windows file dialog box. The user may navigate to their drive, directory, and image of choice. Once a TIF or BMP image has been selected, click on the Open button. The application will then open the Compression Parameters dialog box, shown in Fig. 8.1.

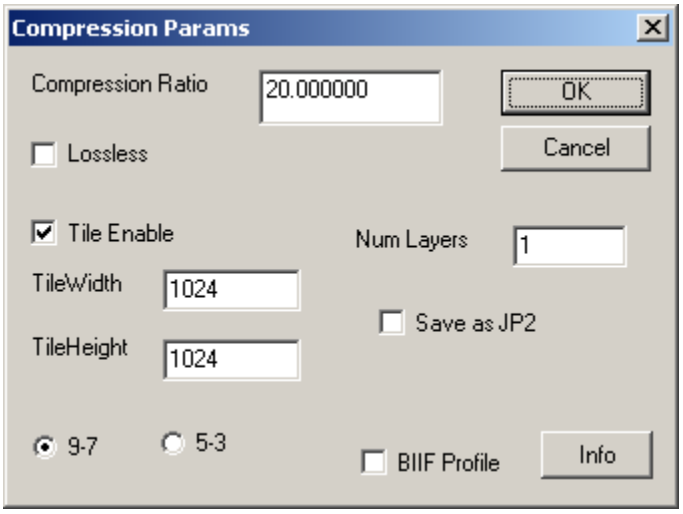

Figure 8.1: The compression parameters dialog box.

The default compression settings for the J2K Viewer are designed to create a JPEG2000 image that is appropriate for viewing in this application. The user is allowed to modify the following compression parameters: compression ratio, tile size, wavelet transform, number of quality layers. In addition, there are buttons that

enable Lossless Compression, JP2 file-format output, and the Basic Image Interchange Format (BIIF) profile. For more details on the BIIF profile, click on the "Info" Button next to the BIIF profile check box or see the detailed description below.

After selecting the compression parameters, clicking on the `OK` button will open a standard windows `Save As` dialog box for the compressed image. The user may then identify the drive, directory and filename for the compressed JPEG2000 image. After selecting the filename for the compressed image and clicking on the `Save` button, the application will load the original image, compress it using JPEG2000 with the compression parameters selected above, save the compressed file to disk, and load the compressed JPEG2000 into the application for viewing. At this point, the application behaves identically to the situation where a JPEG2000 image has been loaded, see Sec. 8.2.1 for details.

BIIF Profile

The BIIF JPEG2000 profile is a set of encoding options for JPEG2000 as defined in BIIF Profile for JPEG2000, Version 01.00, ISO/IEC Working Draft 1.6. Specifically, the profile defines the following parameter values:

- 1024×1024 tiles
- Progressive by quality
- 19 quality layers
- 5 levels of wavelet transform
- No precincts
- Codeblock size of 64×64
- One tile part per tile
- PLT and TLM markers

Developers Note: The BIIF Profile is set using the following set of library calls:

```
/* set tile size */
aw_j2k_set_output_j2k_tile_size(j2k_object,1024, 1024);

/* set Progression order */
aw_j2k_set_output_j2k_progression_order(j2k_object,
AW_J2K_PO_LAYER_RESOLUTION_COMPONENT_POSITION);

/* Set number of layers */
```

```
aw_j2k_set_output_j2k_layers(j2k_object, 19);

/* set number of transform levels */
aw_j2k_set_output_j2k_xform(j2k_object, xform_type, 5);

/* add PLT */
aw_j2k_set_output_j2k_tile_data_toc(j2k_object);

/* add TLM */
aw_j2k_set_output_j2k_tile_toc(j2k_object);
```

Note that the default value for codeblock size is 64×64, so it does not need to be set explicitely. The same is true for precincts, by default the library will not use precincts.

Chapter 9

Command Line Tools

The `j2kdriver` program is included with the SDK. This program provides a command-line interface to the entire set of SDK functions. The complete source code to this program is included as well. For its location see the Installation Chapter, starting on page 5.

9.1 Command Line Program Usage

The command line program accepts the options list listed below. The API function associated with each option is executed as soon as the option is encountered. For example, as soon as the `--input-file-name` option is encountered, the file it names is read into the library object.

The command line switches generally take the form of the ActiveX Visual Basic function name — that is, the leading `aw_j2k_` is removed from the C-function name — with a pair of leading dashes and every underscore replaced by a dash. Every API function has a command line switch to invoke it. There are some API functions that have more than one such switch. There is also a help facility, invoked by the -h switch. Any options that follow the help option will not be executed; instead, a blurb explaining their usage will be printed.

As an example, to read a file named "girl.bmp" and save it as a JPEG 2000 image in the file named "girl.j2k," the following command line should be used:

```
j2kdriver -i girl.bmp -t j2k -o girl.j2k
```

In the following list of the supported command line switches, each command line switch and its arguments are set in a `typewriter` font. Arguments enclosed in square brackets ("`[`" and "`]`") are optional, and may be omitted.

`--help` Turns on the help system. A description of each option listed on the command line following this option will be printed.

`-h` Turns on the help system. A description of each option listed on the command line following this option will be printed.

`--get-library-version [<type>]` Prints the current library version. The optional argument `<type>` can be:

`NUMERIC` print the only the numeric version

`STRING` print the full version string

The full version string is printed by default.

This option calls the `aw_j2k_get_library_version` function, described on page 27.

`--reset-all-options` Resets the J2K library object to its initial state.

This option calls the `aw_j2k_reset_all_options` function, described on page 29.

`--fixed-point` Selects the usage of fixed-point arithmetic for irreversible processing (i.e. when using the I97 wavelet). Using the `--fixed-point` flag is equivalent to using "`--set-internal-option arithmetic fixed`."

This option calls the `aw_j2k_set_internal_option` function, described on page 30.

`--floating-point` Selects the usage of fixed-point arithmetic for irreversible processing (i.e. when using the I97 wavelet). Using the `--floating-point` flag is equivalent to using "`--set-internal-option arithmetic float`."

This option calls the `aw_j2k_set_internal_option` function, described on page 30.

`--set-internal-option <option> <value>` Sets the internal option `<option>` to `<value>`. The options are:

`ARITHMETIC` sets the type of arithmetic operations that will be used for irreversible processing (i.e. when using the I97 wavelet)

For the arithmetic option, the values are:

`FIXED` for fixed-point operation

`FLOAT` for floating-point operation

Using the `--fixed-point` flag is equivalent to using "`--set-internal-option arithmetic fixed`," and using the `--floating-point` flag is equivalent to "`--set-internal-option arithmetic float`."

This option calls the `aw_j2k_set_internal_option` function, described on page 30.

`--set-input-image <file name>` Reads an input image from the file `<file name>`, and inputs it into the library using the memory-buffer interface. The image type is deduced from the image itself.

This option calls the `aw_j2k_set_input_image` function, described on page 35.

`-i <file name>` Reads an input image from the file `<file name>`, and inputs it into the library using the memory-buffer interface. The image type is deduced from the image itself.

This option calls the `aw_j2k_set_input_image` function, described on page 35.

`--input-file-name <file name>` Reads an input image from the file `<file name>`, and inputs it into the library using the memory-buffer interface. The image type is deduced from the image itself.

This option calls the `aw_j2k_set_input_image` function, described on page 35.

`--set-input-image-type <image type> <file name>` Reads an input image from the file `<file name>`, and inputs it into the library using the memory-buffer interface. The image type is specified by `<image type>`, which must be one of:

`J2K` JPEG2000 codestreams

`JP2` JPEG2000 file format

`TIF, TIFF` TIFF

`PNM, PBM, PGM, PPM` Any of the portable bit/gray/pix-map formats

`DCM, DCM_RAW` DICOM (with embedded raw pixels)

`DCM_J2K` DICOM (with embedded JPEG2000 codestreams)

`DCM_JPG` DICOM (with embedded JPEG codestreams)

`BMP` Windows bitmap

`TGA, TARGA` TARGA image format

`JPG, JPEG` Original JPEG format

`PGX` Extended portable graymap format

This option calls the `aw_j2k_set_input_image_type` function, described on page 36.

`--set-input-image-raw <file name> <rows> <columns> <channels> <total bit depth> [<interleaved>]` Reads a raw input image from the file `<file name>`, and inputs it into the library using the memory-buffer interface. The size of the image must be specified using `<rows>`, `<columns>`, `<channels>`, and `<total bit depth>`. The optional `<interleaved>` flag indicates whether the channel data is interleaved or not (the default if not specified). `<interleaved>` may be "`yes`" or "`no`."

This option calls the `aw_j2k_set_input_image_raw` function, described on page 38.

`--input-file-raw <file name> <rows> <columns> <channels> <total bit depth> [<interleaved>]` Reads a raw input image from the file `<file name>`, and inputs it into the library using the memory-buffer interface. The size of the image must be specified using `<rows>`, `<columns>`, `<channels>`, and `<total bit depth>`. The optional `<interleaved>` flag indicates whether the channel data is interleaved or not (the default if not specified). `<interleaved>` may be "`yes`" or "`no`."

This option calls the `aw_j2k_set_input_image_raw` function, described on page 38.

`--set-input-image-raw-ex <file name> <rows> <columns> <channels> <bits allocated> <bits stored> [<interleaved>]` Reads a raw input image from the file `<file name>`, and inputs it into the library using the memory-buffer interface. The size of the image must be specified using `<rows>`, `<columns>`, `<channels>`, `<bits allocated>`, and `<bits stored>`. The optional `<interleaved>` flag indicates whether the channel data is interleaved or not (the default if not specified). `<interleaved>` may be "`yes`" or "`no`."

This option calls the `aw_j2k_set_input_image_raw_ex` function, described on page 38.

`--set-input-image-raw-channel <channel index> <channel data file name>` Loads the data for channel number `<channel index>` from `<channel data file name>`. The data is input into the library using the memory-buffer interface. The dimensions of the image can be provided via a call to `--set-input-raw-information`.

This option calls the `aw_j2k_set_input_image_raw_channel` function, described on page 41.

`--set-input-image-raw-region <region file name> <row offset> <col offset> <rows> <cols> <interleaved>` Loads the data for give region from `<channel data file name>`. The data is input into the library using the memory-buffer interface. The dimensions of the image should be provided via a call to `--set-input-raw-information`.

This option calls the `aw_j2k_set_input_image_raw_region` function, described on page 42.

`--set-input-image-file <file name>` Reads an input image from the file `<file name>`, and inputs it into the library using the file interface. The image type is deduced from the image itself.

This option calls the `aw_j2k_set_input_image_file` function, described on page 46.

`--input-file <file name>` Reads an input image from the file `<file name>`, and inputs it into the library using the file interface. The image type is deduced from the image itself.

This option calls the `aw_j2k_set_input_image_file` function, described on page 46.

`--set-input-image-file-type <file name> <image type>` Reads an input image from the file `<file name>`, and inputs it into the library using the memory-buffer interface. The image type is specified by `<image type>`, which must be one of:

`J2K` JPEG2000 codestreams

`JP2` JPEG2000 file format

`TIF, TIFF` TIFF

`PNM, PBM, PGM, PPM` Any of the portable bit/gray/pix-map formats

`DCM, DCM_RAW` DICOM (with embedded raw pixels)

`DCM_J2K` DICOM (with embedded J2K codestreams)

`BMP` Windows bitmap

`TGA, TARGA` TARGA image format

`JPG, JPEG` Original JPEG format

`PGX` Extended portable graymap format

This option calls the `aw_j2k_set_input_image_file_type` function, described on page 47.

`--set-input-image-file-raw <file name> <rows> <columns> <channels> <total bit depth> [<interleaved>]` Reads a raw input image from the file `<file name>`, and inputs it into the library using the file interface. The size of the image must be specified using `<rows>`, `<columns>`, `<channels>`, and `<total bit depth>`. The optional `<interleaved>` flag indicates whether the channel data is interleaved or not (the default if not specified). `<interleaved>` may be "`yes`" or "`no`."

This option calls the `aw_j2k_set_input_image_file_raw` function, described on page 48.

`--input-image-file-raw <file name> <rows> <columns> <channels> <total bit depth> [<interleaved>]` Reads a raw input image from the file `<file name>`, and inputs it into the library using the file interface. The size of the image must be specified using `<rows>`, `<columns>`, `<channels>`, and `<total bit depth>`. The optional `<interleaved>` flag indicates whether the channel data is interleaved or not (the default if not specified). `<interleaved>` may be "`yes`" or "`no`."

This option calls the `aw_j2k_set_input_image_file_raw` function, described on page 48.

`--set-input-raw-information <rows> <columns> <channels> <total bit depth> [<interleaved>]` Sets the dimensions of an image, without providing the image itself. The image data can be provided later using `--set-input-image-raw-channel`. If the data is signed, the bit depth should be negative; it should be positive for unsigned data. `<interleaved>` may be "`yes`" or "`no`."

This option calls the `aw_j2k_set_input_raw_information` function, described on page 50.

`--set-input-raw-channel-bpp <channel index> <channel bit depth>` Sets the bit depth of the data in the specified channel. If the data is signed, the bit depth should be negative; it should be positive for unsigned data.

This option calls the `aw_j2k_set_input_raw_channel_bpp` function, described on page 52.

`--set-input-raw-endianness <endianness>` Sets the endianness of multi-byte-per-channel raw images. The value of `<endianness>` should be "`big`" or "`little`."

This option calls the `aw_j2k_set_input_raw_endianness` function, described on page 53.

`--set-input-raw-channel-subsampling <channel index> <horizontal subsampling> <vertical subsampling>` Specifies the subsampling of a specific channel relative to the full image dimensions.

This option calls the `aw_j2k_set_input_raw_channel_subsampling` function, described on page 54.

`--get-input-image-info` Prints the dimensions, total bit depth, and number of channels in the image. The bit depth returned here is the sum of the bit depths of all the channels, and does not indicate whether the data is signed or not. Use `--get-input-channel-bpp` to determine the bit depth of each channel, as well as whether the data in the channel is signed.

This option calls the `aw_j2k_get_input_image_info` function, described on page 56.

`--get-input-channel-bpp <channel index>` Returns the bit depth of the specified channel in the image, as well as whether the data in the channel is signed or not.

This option calls the `aw_j2k_get_input_raw_channel_bpp` function, described on page 52.

`--get-input-j2k-channel-subsampling <channel index>` Prints the subsampling of a channel in a JPEG2000 image relative to the full image dimensions.

This option calls the `aw_j2k_get_input_j2k_channel_subsampling` function, described on page 59.

`--get-input-j2k-num-comments` Returns the number of comments that are embedded in a JPEG2000 image.

This option calls the `aw_j2k_get_input_j2k_num_comments` function, described on page 60.

`--get-input-j2k-comments <comment index>` Returns a comment from a JPEG-2000 image. The number of comments can be found using `--get-input-j2k-num-comments`.

This option calls the `aw_j2k_get_input_j2k_comments` function, described on page 61.

`--get-comments <comment index>` Returns a comment from a JPEG2000 image. The number of comments can be found using `--get-input-j2k-num-comments`.

This option calls the `aw_j2k_get_input_j2k_comments` function, described on page 61.

`--get-input-j2k-progression-info` Prints the progression order, number of wavelet transform levels, number of quality layers, and number of color channels in a JPEG 2000 image.

This option calls the `aw_j2k_get_input_j2k_progression_info` function, described on page 63.

`--progression-info` Prints the progression order, number of wavelet transform levels, number of quality layers, and number of color channels in a JPEG 2000 image.

This option calls the `aw_j2k_get_input_j2k_progression_info` function, described on page 63.

`--image-progression-info` Prints the progression order, number of wavelet transform levels, number of quality layers, and number of color channels in a JPEG 2000 image.

This option calls the `aw_j2k_get_input_j2k_progression_info` function, described on page 63.

`--get-input-j2k-progression-byte-count <progression level index>` Prints the number of bytes needed to decode a given progression level. The type of progression level is determined by the progression order of the JPEG2000 image.

This option calls the `aw_j2k_get_input_j2k_progression_byte_count` function, described on page 65.

`--get-input-j2k-tile-info` Prints the tile size and image and tile offsets in a JPEG2000 image.

This option calls the `aw_j2k_get_input_j2k_tile_info` function, described on page 67.

`--image-tiling-info` Prints the tile size and image and tile offsets in a JPEG2000 image.

This option calls the `aw_j2k_get_input_j2k_tile_info` function, described on page 67.

`--get-input-j2k-selected-tile-geometry <tile index>` Returns the size of a tile and its offset from the image origin.

This option calls the `aw_j2k_get_input_j2k_selected_tile_geometry` function, described on page 68.

`--get-input-j2k-component-registration <channel index>` Returns a specified channel's registration offset. The registration offsets indicate how much to offset each color channel relative to the image grid for display. The offsets are given in units of 1/65536 of a pixel.

This option calls the `aw_j2k_get_input_j2k_component_registration` function, described on page 69.

`--get-component-registration <channel index>` Returns a specified channel's registration offset. The registration offsets indicate how much to offset each color channel relative to the image grid for display. The offsets are given in units of 1/65536 of a pixel.

This option calls the `aw_j2k_get_input_j2k_component_registration` function, described on page 69.

`--get-input-j2k-bytes-read` Prints the number of bytes actually used from a JPEG2000 image. If the full image was decompressed, this will return the size of the original image. If a partial decompression was performed (e.g. only a single tile or a reduced resolution was decoded) then this number may be smaller than the full image.

This option calls the `aw_j2k_get_input_j2k_bytes_read` function, described on page 70.

`--get-bytecount` Prints the number of bytes actually used from a JPEG2000 image. If the full image was decompressed, this will return the size of the original image. If a partial decompression was performed (e.g. only a single tile or a reduced resolution was decoded) then this number may be smaller than the full image.

This option calls the `aw_j2k_get_input_j2k_bytes_read` function, described on page 70.

`--set-input-j2k-progression-level <progression level>` Sets the number of progression levels that will be decoded form the JPEG2000 image. The progression type is determined by the progression order of the image.

This option calls the `aw_j2k_set_input_j2k_progression_level` function, described on page 75.

`-pl <progression level>` Sets the number of progression levels that will be decoded form the JPEG2000 image. The progression type is determined by the progression order of the image.

This option calls the `aw_j2k_set_input_j2k_progression_level` function, described on page 75.

`--progression-level <progression level>` Sets the number of progression levels that will be decoded form the JPEG2000 image. The progression type is determined by the progression order of the image.

This option calls the `aw_j2k_set_input_j2k_progression_level` function, described on page 75.

`--set-input-j2k-resolution-level <resolution level> [<full transform>]` Sets the number of resolution levels to decode from the JPEG2000 image. The full number of resolution levels is always one more than the number of wavelet transform levels in the JPEG2000 image. To fully decode the image, set the resolution level to 1. For quarter resolution, use 2. The optional full transform flag, which should be "`YES`" or "`NO`," indicates whether the full wavelet transform should be performed (resulting in a full size image of lower quality) or not (resulting in a reduced size image).

This option calls the `aw_j2k_set_input_j2k_resolution_level` function, described on page 77.

`-rl <resolution level> [<full transform>]` Sets the number of resolution levels to decode from the JPEG2000 image. The full number of resolution levels is always one more than the number of wavelet transform levels in the JPEG-2000 image. To fully decode the image, set the resolution level to 1. For quarter resolution, use 2. The optional full transform flag, which should be "`YES`" or "`NO`," indicates whether the full wavelet transform should be performed (resulting in a full size image of lower quality) or not (resulting in a reduced size image).

This option calls the `aw_j2k_set_input_j2k_resolution_level` function, described on page 77.

`--resolution-level <resolution level> [<full transform>]` Sets the number of resolution levels to decode from the JPEG2000 image. The full number of resolution levels is always one more than the number of wavelet transform levels in the JPEG2000 image. To fully decode the image, set the resolution level to 1. For quarter resolution, use 2. The optional full transform flag, which should be "`YES`" or "`NO`," indicates whether the full wavelet transform should be performed (resulting in a full size image of lower quality) or not (resulting in a reduced size image).

This option calls the `aw_j2k_set_input_j2k_resolution_level` function, described on page 77.

`--set-input-j2k-quality-level <quality level>` Sets the number of quality layers to decode from a JPEG2000 image. Setting this value to 1 decodes all the quality layers; setting it to 2 drops one layer.

This option calls the `aw_j2k_set_input_j2k_quality_level` function, described on page 78.

`-ql <quality level>` Sets the number of quality layers to decode from a JPEG2000 image. Setting this value to 1 decodes all the quality layers; setting it to 2 drops one layer.

This option calls the `aw_j2k_set_input_j2k_quality_level` function, described on page 78.

`--quality-level <quality level>` Sets the number of quality layers to decode from a JPEG2000 image. Setting this value to 1 decodes all the quality layers; setting it to 2 drops one layer.

This option calls the `aw_j2k_set_input_j2k_quality_level` function, described on page 78.

`--set-input-j2k-color-level <color level> [<color transform flag>]` Sets the number of color channels to decode from a JPEG2000 image. Setting `<color level>` to 1 decodes all the color channels. Setting it to 2 drops the last color channel. The optional `<color transform flag>` indicates whether the color transform will be performed or not. Use "`YES`" to force the transform to be performed if possible; "`NO`" to disallow the color transform; and "`DEFAULT`" to let the transform be performed if so specified by the image.

This option calls the `aw_j2k_set_input_j2k_color_level` function, described on page 79.

`-cl <color level> [<color transform flag>]` Sets the number of color channels to decode from a JPEG2000 image. Setting `<color level>` to 1 decodes all the color channels. Setting it to 2 drops the last color channel. The optional `<color transform flag>` indicates whether the color transform will be performed or not. Use "`YES`" to force the transform to be performed if possible; "`NO`" to disallow the color transform; and "`DEFAULT`" to let the transform be performed if so specified by the image.

This option calls the `aw_j2k_set_input_j2k_color_level` function, described on page 79.

`--color-level <color level> [<color transform flag>]` Sets the number of color channels to decode from a JPEG2000 image. Setting `<color level>` to 1 decodes all the color channels. Setting it to 2 drops the last color channel. The optional `<color transform flag>` indicates whether the color transform will be performed or not. Use "`YES`" to force the transform to be performed if possible;

"NO" to disallow the color transform; and "DEFAULT" to let the transform be performed if so specified by the image.

This option calls the aw_j2k_set_input_j2k_color_level function, described on page 79.

--set-input-j2k-region-level <left> <top> <right> <bottom> Defines a rectangular subregion to be decoded from a JPEG2000 image. Only those precincts that intersect with the defined region will be decoded. Precincts are set by calling --set-output-j2k-channel-precinct-size

This option calls the aw_j2k_set_input_j2k_region_level function, described on page 80.

-wl <left> <top> <right> <bottom> Defines a rectangular subregion to be decoded from a JPEG2000 image. Only those precincts that intersect with the defined region will be decoded. Precincts are set by calling --set-output-j2k-channel-precinct-size

This option calls the aw_j2k_set_input_j2k_region_level function, described on page 80.

--decode-region <left> <top> <right> <bottom> Defines a rectangular subregion to be decoded from a JPEG2000 image. Only those precincts that intersect with the defined region will be decoded. Precincts are set by calling --set-output-j2k-channel-precinct-size

This option calls the aw_j2k_set_input_j2k_region_level function, described on page 80.

--set-input-j2k-selected-channel <channel index> Selects a specific single color channel to decode from a JPEG2000 image. The color channels are indexed starting at 0. Any color rotations specified in the JPEG2000 image will not be performed on this channel.

This option calls the aw_j2k_set_input_j2k_selected_channel function, described on page 81.

-sc <channel index> Selects a specific single color channel to decode from a JPEG-2000 image. The color channels are indexed starting at 0. Any color rotations specified in the JPEG2000 image will not be performed on this channel.

This option calls the aw_j2k_set_input_j2k_selected_channel function, described on page 81.

--selected-channel <channel index> Selects a specific single color channel to decode from a JPEG2000 image. The color channels are indexed starting at 0. Any color rotations specified in the JPEG2000 image will not be performed on this channel.

This option calls the aw_j2k_set_input_j2k_selected_channel function, described on page 81.

`--set-input-j2k-selected-tile <tile index>` Selects a specific single tile to decode from a JPEG2000 image.

This option calls the `aw_j2k_set_input_j2k_selected_tile` function, described on page 82.

`-sl <tile index>` Selects a specific single tile to decode from a JPEG2000 image.

This option calls the `aw_j2k_set_input_j2k_selected_tile` function, described on page 82.

`--selected-tile <tile index>` Selects a specific single tile to decode from a JPEG-2000 image.

This option calls the `aw_j2k_set_input_j2k_selected_tile` function, described on page 82.

`--set-input-dcm-frame-index <frame index>` This parameter specifies the desired single frame number to extract from a multiframe DICOM file. If the input and output formats are DICOM, the default value is -1, which means that all the frames in the file will be processed. If the output format is not DICOM, the default value is 0, which refers to the first frame. If the frame index value specified is greater than the number of frames available in the input file, an error is returned.

This option calls the `aw_j2k_set_input_dcm_frame_index` function, described on page 55.

`--input-reset-options` Resets all the input options to their default values.

This option calls the `aw_j2k_input_reset_options` function, described on page 84.

`--set-output-type <output type>` Selects the type for the output image. The output type is one of:

`J2K` JPEG2000 codestreams

`JP2` JPEG2000 file format

`TIF, TIFF` TIFF

`PNM, PBM, PGM, PPM` Any of the portable bit/gray/pix-map formats

`DCM, DCM_RAW` DICOM (with embedded raw pixels)

`DCM_J2K` DICOM (with embedded J2K codestreams)

`BMP` Windows bitmap

`TGA, TARGA` TARGA image format

`JPG, JPEG` Original JPEG format

`PGX` Extended portable graymap format

`RAW` Unformatted raw data

This option calls the `aw_j2k_set_output_type` function, described on page 95.

`-t <output type>` Selects the type for the output image. The output type is one of:

`J2K` JPEG2000 codestreams

`JP2` JPEG2000 file format

`TIF, TIFF` TIFF

`PNM, PBM, PGM, PPM` Any of the portable bit/gray/pix-map formats

`DCM, DCM_RAW` DICOM (with embedded raw pixels)

`DCM_J2K` DICOM (with embedded J2K codestreams)

`BMP` Windows bitmap

`TGA, TARGA` TARGA image format

`JPG, JPEG` Original JPEG format

`PGX` Extended portable graymap format

`RAW` Unformatted raw data

This option calls the `aw_j2k_set_output_type` function, described on page 95.

`--output-file-type <output type>` Selects the type for the output image. The output type is one of:

`J2K` JPEG2000 codestreams

`JP2` JPEG2000 file format

`TIF, TIFF` TIFF

`PNM, PBM, PGM, PPM` Any of the portable bit/gray/pix-map formats

`DCM, DCM_RAW` DICOM (with embedded raw pixels)

`DCM_J2K` DICOM (with embedded J2K codestreams)

`BMP` Windows bitmap

`TGA, TARGA` TARGA image format

`JPG, JPEG` Original JPEG format

`PGX` Extended portable graymap format

`RAW` Unformatted raw data

This option calls the `aw_j2k_set_output_type` function, described on page 95.

`--get-output-image <output image name>` Writes a new image to `<output image name>` using the memory-buffer interface. The output image type must already have been specified using `--set-output-type`

This option calls the `aw_j2k_get_output_image` function, described on page 96.

`-o <output image name>` Writes a new image to `<output image name>` using the memory-buffer interface. The output image type must already have been specified using `--set-output-type`

This option calls the `aw_j2k_get_output_image` function, described on page 96.

`--output-file-name <output image name>` Writes a new image to `<output image name>` using the memory-buffer interface. The output image type must already have been specified using `--set-output-type`

This option calls the `aw_j2k_get_output_image` function, described on page 96.

`--get-output-image-type <output type> <output image name>` Writes a new image to `<output image name>` using the memory-buffer interface. The output type is one of:

`J2K` JPEG2000 codestreams

`JP2` JPEG2000 file format

`TIF, TIFF` TIFF

`PNM, PBM, PGM, PPM` Any of the portable bit/gray/pix-map formats

`DCM, DCM_RAW` DICOM (with embedded raw pixels)

`DCM_J2K` DICOM (with embedded J2K codestreams)

`BMP` Windows bitmap

`TGA, TARGA` TARGA image format

`JPG, JPEG` Original JPEG format

`PGX` Extended portable graymap format

`RAW` Unformatted raw data

This option calls the `aw_j2k_get_output_image_type` function, described on page 97.

`--get-output-image-raw <output image name> [<interleaved>]` Writes a new raw image to `<output image name>` using the memory-buffer interface. The size of the image will be printed if verbose operation was selected. The optional `<interleaved>` flag, which may be "`YES`" or "`NO`", indicates whether the color channels should be interleaved or not.

This option calls the `aw_j2k_get_output_image_raw` function, described on page 98.

`--get-output-image-raw-ex <output image name> [<bits allocated>] [<interleaved>]` Writes a new raw image to `<output image name>` using the memory-buffer interface. The size of the image will be printed if verbose operation was selected. The optional `<bits allocated>` flag controls the padding

of the pixel data. The optional <interleaved> flag, which may be "YES" or "NO", indicates whether the color channels should be interleaved or not.

This option calls the aw_j2k_get_output_image_raw_ex function, described on page 98.

--get-output-image-file <output image name> Writes a new image to <output image name> using the file interface. The output image type must have been specified using --set-output-type

This option calls the aw_j2k_get_output_image_file function, described on page 101.

--get-output-image-file-type <output type> <output image name> Writes a new image to <output image name> using the file interface. The output type is one of:

J2K JPEG2000 codestreams

JP2 JPEG2000 file format

TIF, TIFF TIFF

PNM, PBM, PGM, PPM Any of the portable bit/gray/pix-map formats

DCM, DCM_RAW DICOM (with embedded raw pixels)

DCM_J2K DICOM (with embedded J2K codestreams)

BMP Windows bitmap

TGA, TARGA TARGA image format

JPG, JPEG Original JPEG format

PGX Extended portable graymap format

RAW Unformatted raw data

This option calls the aw_j2k_get_output_image_file_type function, described on page 102.

--get-output-image-file-raw <output image name> [<interleaved>] Writes a new raw image to <output image name> using the memory-buffer interface. The size of the image will be printed if verbose operation was selected. The optional <interleaved> flag, which may be "YES" or "NO", indicates whether the color channels should be interleaved or not.

This option calls the aw_j2k_get_output_image_file_raw function, described on page 104.

--set-output-j2k-bitrate <bitrate> Sets the output image bitrate, in bits per pixel. A bitrate of 0 indicates that all the quantized data should be included in the image. This creates lossless images if the R53 wavelet is chosen using --set-output-j2k-xform.

This option calls the `aw_j2k_set_output_j2k_bitrate` function, described on page 106.

`-B <bitrate>` Sets the output image bitrate, in bits per pixel. A bitrate of 0 indicates that all the quantized data should be included in the image. This creates lossless images if the R53 wavelet is chosen using `--set-output-j2k-xform`.

This option calls the `aw_j2k_set_output_j2k_bitrate` function, described on page 106.

`--bitrate <bitrate>` Sets the output image bitrate, in bits per pixel. A bitrate of 0 indicates that all the quantized data should be included in the image. This creates lossless images if the R53 wavelet is chosen using `--set-output-j2k-xform`.

This option calls the `aw_j2k_set_output_j2k_bitrate` function, described on page 106.

`--set-output-j2k-ratio <ratio>` Sets the compression ratio for the output image. A ratio of 0 indicates that all the quantized data should be included in the image. This creates lossless images if the R53 wavelet is chosen using `--set-output-j2k-xform`.

This option calls the `aw_j2k_set_output_j2k_ratio` function, described on page 107.

`-R <ratio>` Sets the compression ratio for the output image. A ratio of 0 indicates that all the quantized data should be included in the image. This creates lossless images if the R53 wavelet is chosen using `--set-output-j2k-xform`.

This option calls the `aw_j2k_set_output_j2k_ratio` function, described on page 107.

`--ratio <ratio>` Sets the compression ratio for the output image. A ratio of 0 indicates that all the quantized data should be included in the image. This creates lossless images if the R53 wavelet is chosen using `--set-output-j2k-xform`.

This option calls the `aw_j2k_set_output_j2k_ratio` function, described on page 107.

`--set-output-j2k-filesize <file size>` Sets the output image size, in bytes.

This option calls the `aw_j2k_set_output_j2k_filesize` function, described on page 108.

`-F <file size>` Sets the output image size, in bytes.

This option calls the `aw_j2k_set_output_j2k_filesize` function, described on page 108.

`--file-size <file size>` Sets the output image size, in bytes.

This option calls the `aw_j2k_set_output_j2k_filesize` function, described on page 108.

`--set-output-j2k-psnr <psnr>` Sets the output image pSNR, in dB.

This option calls the `aw_j2k_set_output_j2k_psnr` function, described on page 109.

`--set-output-j2k-color-xform <color transform flag>` Sets whether the color transform will be performed. The value of `<color transform flag>` should be "`YES`" to perform and "`NO`" to skip the transform. The color transform can only be performed only if there are 3 or more channels and the first three channels have the same bit depth, data type (signed or unsigned), and subsampling.

This option calls the `aw_j2k_set_output_j2k_color_xform` function, described on page 112.

`--color-transform <color transform flag>` Sets whether the color transform will be performed. The value of `<color transform flag>` should be "`YES`" to perform and "`NO`" to skip the transform. The color transform can only be performed only if there are 3 or more channels and the first three channels have the same bit depth, data type (signed or unsigned), and subsampling.

This option calls the `aw_j2k_set_output_j2k_color_xform` function, described on page 112.

`--set-output-j2k-color-xform-ex <tile index> <transform flag>` Sets whether the color transform will be performed in tile `<tile index>`. The value of `<transform flag>` should be `yes` to perform and `no` to skip the transform. The color transform can only be performed only if there are 3 or more channels and the first three channels have the same bit depth, data type (signed or unsigned), and subsampling.

`--set-output-j2k-progression-order <progression order>` Sets the progression order of the compressed image file. The value of `<progression order>` is one of the following:

`LRCP` Layer, Resolution, Component, Position

`RLCP` Resolution, Layer, Component, Position

`RPCL` Resolution, Position, Component, Layer

`PCRL` Position, Component, Resolution, Layer

`CPRL` Component, Position, Resolution, Layer

This option calls the `aw_j2k_set_output_j2k_progression_order` function, described on page 113.

`-p <progression order>` Sets the progression order of the compressed image file. The value of `<progression order>` is one of the following:

`LRCP` Layer, Resolution, Component, Position

`RLCP` Resolution, Layer, Component, Position

`RPCL` Resolution, Position, Component, Layer

`PCRL` Position, Component, Resolution, Layer

`CPRL` Component, Position, Resolution, Layer

This option calls the `aw_j2k_set_output_j2k_progression_order` function, described on page 113.

`--progression-order <progression order>` Sets the progression order of the compressed image file. The value of `<progression order>` is one of the following:

`LRCP` Layer, Resolution, Component, Position

`RLCP` Resolution, Layer, Component, Position

`RPCL` Resolution, Position, Component, Layer

`PCRL` Position, Component, Resolution, Layer

`CPRL` Component, Position, Resolution, Layer

This option calls the `aw_j2k_set_output_j2k_progression_order` function, described on page 113.

`--set-output-j2k-tile-size <tile width> <tile height>` Sets the tile size for JPEG2000 output. All tiles, except those at the edges of the image, will be of this size.

This option calls the `aw_j2k_set_output_j2k_tile_size` function, described on page 115.

`--tile-size <tile width> <tile height>` Sets the tile size for JPEG2000 output. All tiles, except those at the edges of the image, will be of this size.

This option calls the `aw_j2k_set_output_j2k_tile_size` function, described on page 115.

`--set-output-j2k-add-comment <comment string> [<comment type> [<comment location>]]` Adds a comment to the JPEG2000 image. The value of `<comment string>` is used verbatim for the comment. The optional `<comment type>` flag, whose values may be "`BINARY`" or "`TEXT`," denotes whether the comment is binary data or Latin-1 (ISO-8859-15) text. Text is the default type. The optional `<comment location>` field may be "`MAIN`" to denote that the comment will be stored in the main header, or an index to denote that the comment will be stored in the header of the indexed tile. The default location is the main header.

This option calls the `aw_j2k_set_output_j2k_add_comment` function, described on page 116.

`--add-comment <comment string> [<comment type> [<comment location>]]` Adds a comment to the JPEG2000 image. The value of `<comment string>` is used verbatim for the comment. The optional `<comment type>` flag, whose values may be "`BINARY`" or "`TEXT`," denotes whether the comment is binary data or Latin-1 (ISO-8859-15) text. Text is the default type. The optional `<comment location>` field may be "`MAIN`" to denote that the comment will be stored in the main header, or an index to denote that the comment will be stored in the header of the indexed tile. The default location is the main header.

This option calls the `aw_j2k_set_output_j2k_add_comment` function, described on page 116.

`--add-comment-from-file <comment file> [<comment type> [<comment location>]]` Adds a comment to the JPEG2000 image. The value of `<comment string>` is used verbatim for the comment. The optional `<comment type>` flag, whose values may be "`BINARY`" or "`TEXT`," denotes whether the comment is binary data or Latin-1 (ISO-8859-15) text. Text is the default type. The optional `<comment location>` field may be "`MAIN`" to denote that the comment will be stored in the main header, or an index to denote that the comment will be stored in the header of the indexed tile. The default location is the main header.

This option calls the `aw_j2k_set_output_j2k_add_comment` function, described on page 116.

`--set-output-j2k-image-offset <x offset> <y offset>` Offsets the image boundary from the top-left corner of the image grid, thereby cropping the image.

This option calls the `aw_j2k_set_output_j2k_image_offset` function, described on page 119.

`--image-offset <x offset> <y offset>` Offsets the image boundary from the top-left corner of the image grid, thereby cropping the image.

This option calls the `aw_j2k_set_output_j2k_image_offset` function, described on page 119.

`--set-output-j2k-tile-offset <x offset> <y offset>` Offsets the tile boundary from the top-left corner of the image grid. The tile offset must be within the rectangle defined by the origin and the image offset.

This option calls the `aw_j2k_set_output_j2k_tile_offset` function, described on page 120.

`--tile-offset <x offset> <y offset>` Offsets the tile boundary from the top-left corner of the image grid. The tile offset must be within the rectangle defined by the origin and the image offset.

This option calls the `aw_j2k_set_output_j2k_tile_offset` function, described on page 120.

`--set-output-j2k-error-resilience <options list>` Enables and disables the use of the error resilience features of JPEG2000. Allows Start of Packet and End of Packet Header markers to be added to the codestream, as well as Segmentation Symbols to be embedded in it. The error resilience options are:

`SOP` to enable Start of Packet markers

`EPH` to enable End of Packet Header markers

`SEG` to enable segmentation symbols

`ALL` to enable all of the above

`NONE` to disable all of the options

Multiple options may be listed, separated by commas.

This option calls the `aw_j2k_set_output_j2k_error_resilience` function, described on page 121.

`-E <error resilience options list>` Enables and disables the use of the error resilience features of JPEG2000. Allows Start of Packet and End of Packet Header markers to be added to the codestream, as well as Segmentation Symbols to be embedded in it. The error resilience options are:

`SOP` to enable Start of Packet markers

`EPH` to enable End of Packet Header markers

`SEG` to enable segmentation symbols

`ALL` to enable all of the above

`NONE` to disable all of the options

Multiple options may be listed, separated by commas.

This option calls the `aw_j2k_set_output_j2k_error_resilience` function, described on page 121.

`--error-resilience <error resilience options list>` Enables and disables the use of the error resilience features of JPEG2000. Allows Start of Packet and End of Packet Header markers to be added to the codestream, as well as Segmentation Symbols to be embedded in it. The error resilience options are:

`SOP` to enable Start of Packet markers

`EPH` to enable End of Packet Header markers

`SEG` to enable segmentation symbols

`ALL` to enable all of the above

`NONE` to disable all of the options

Multiple options may be listed, separated by commas.

This option calls the `aw_j2k_set_output_j2k_error_resilience` function, described on page 121.

`--set-output-j2k-error-resilience-ex <tile index> <channel index> <error resilience options list>` Enables and disables the use of the error resilience features of JPEG2000 for the given channel of a tile. Allows Start of Packet and End of Packet Header markers to be added to the codestream, as well as Segmentation Symbols to be embedded in it. SOP and EPH markers cannot be set individually by channel; they must be enabled or disabled for all channels of a tile. The error resilience options are:

`SOP` to enable Start of Packet markers

`EPH` to enable End of Packet Header markers

`SEG` to enable segmentation symbols

`ALL` to enable all of the above

`NONE` to disable all of the options

Multiple options may be listed, separated by commas.

This option calls the `aw_j2k_set_output_j2k_error_resilience_ex` function, described on page 121.

`-w <transform type> [<transform depth>]` Selects the type of wavelet and (optionally) the transform depth to use for compression. The wavelet choices are:

`I97` irreversible 9-7 wavelet

`R53` reversible 5-3 wavelet

This option calls the `aw_j2k_set_output_j2k_xform` function, described on page 123.

`--wavelet-transform <transform type> [<transform depth>]` Selects the type of wavelet and (optionally) the transform depth to use for compression. The wavelet choices are:

`I97` irreversible 9-7 wavelet

`R53` reversible 5-3 wavelet

This option calls the `aw_j2k_set_output_j2k_xform` function, described on page 123.

`--set-output-j2k-xform <transform type> [<transform depth>]` Selects the type of wavelet and (optionally) the transform depth to use for compression. The wavelet choices are:

`I97` irreversible 9-7 wavelet

`R53` reversible 5-3 wavelet

This option calls the `aw_j2k_set_output_j2k_xform` function, described on page 123.

`--set-output-j2k-xform-ex <tile index> <channel index> <type> [<depth>]` Selects the type of wavelet and (optionally) the transform depth to use for compression of the given component of a tile. The wavelet choices are:

`I97` irreversible 9-7 wavelet

`R53` reversible 5-3 wavelet

This option calls the `aw_j2k_set_output_j2k_xform_ex` function, described on page 123.

`--set-output-j2k-wavelet-mct <type> [<depth>] [<number of components>]` Selects the type of multi-component wavelet transform, (optionally) the transform depth to use for compression and (optionally) the number of components for each component collection. The wavelet choices are:

`I97` irreversible 9-7 wavelet

`R53` reversible 5-3 wavelet

This option calls the `aw_j2k_set_output_j2k_wavelet_mct` function, described on page 190.

`-mct <transform type> [<transform depth>] [<number of components>]` Selects the type of multi-component wavelet transform, (optionally) the transform depth to use for compression and (optionally) the number of components for each component collection. The wavelet choices are:

`I97` irreversible 9-7 wavelet

`R53` reversible 5-3 wavelet

This option calls the `aw_j2k_set_output_j2k_wavelet_mct` function, described on page 190.

`--set-output-j2k-layers <quality layers>` Sets the number of quality layers to be embedded in a JPEG2000 image.

This option calls the `aw_j2k_set_output_j2k_layers` function, described on page 125.

`-y <quality layers>` Sets the number of quality layers to be embedded in a JPEG-2000 image.

This option calls the `aw_j2k_set_output_j2k_layers` function, described on page 125.

`--layers <quality layers>` Sets the number of quality layers to be embedded in a JPEG2000 image.

This option calls the `aw_j2k_set_output_j2k_layers` function, described on page 125.

`--set-output-j2k-layer-bitrate <layer index> <bitrate>` Sets the target bitrate, in bits per pixel, for the selected quality layer.

This option calls the `aw_j2k_set_output_j2k_layer_bitrate` function, described on page 126.

`-yB <layer index> <bitrate>` Sets the target bitrate, in bits per pixel, for the selected quality layer.

This option calls the `aw_j2k_set_output_j2k_layer_bitrate` function, described on page 126.

`--layer-bitrate <layer index> <bitrate>` Sets the target bitrate, in bits per pixel, for the selected quality layer.

This option calls the `aw_j2k_set_output_j2k_layer_bitrate` function, described on page 126.

`--set-output-j2k-layer-bitrate-ex <tile index> <layer index> <bitrate>` Sets the target bitrate, in bits per pixel, for the selected quality layer of the selected tile.

This option calls the `aw_j2k_set_output_j2k_layer_bitrate_ex` function, described on page 126.

`--set-output-j2k-layer-ratio <layer index> <ratio>` Sets the target compression ratio for the selected quality layer.

This option calls the `aw_j2k_set_output_j2k_layer_ratio` function, described on page 130.

`-yR <layer index> <ratio>` Sets the target compression ratio for the selected quality layer.

This option calls the `aw_j2k_set_output_j2k_layer_ratio` function, described on page 130.

`--layer-ratio <layer index> <ratio>` Sets the target compression ratio for the selected quality layer.

This option calls the `aw_j2k_set_output_j2k_layer_ratio` function, described on page 130.

`--set-output-j2k-layer-ratio-ex <tile index> <layer index> <ratio>` Sets the target compression ratio for the selected quality layer of the selected tile.

This option calls the `aw_j2k_set_output_j2k_layer_ratio_ex` function, described on page 130.

`--set-output-j2k-layer-size <layer index> <byte count>` Sets the target size, in bytes, of the selected quality layer.

This option calls the `aw_j2k_set_output_j2k_layer_size` function, described on page 134.

`-yF <layer index> <byte count>` Sets the target size, in bytes, of the selected quality layer.

This option calls the `aw_j2k_set_output_j2k_layer_size` function, described on page 134.

`--layer-size <layer index> <byte count>` Sets the target size, in bytes, of the selected quality layer.

This option calls the `aw_j2k_set_output_j2k_layer_size` function, described on page 134.

`--set-output-j2k-layer-size-ex <tile index> <layer index> <byte count>` Sets the target size, in bytes, of the selected quality layer of the selected tile.

This option calls the `aw_j2k_set_output_j2k_layer_size_ex` function, described on page 134.

`--set-output-j2k-layer-psnr <layer index> <psnr>` Sets the target pSNR, in dB, of the selected quality layer.

This option calls the `aw_j2k_set_output_j2k_layer_psnr` function, described on page 137.

`--set-output-j2k-layer-psnr-ex <tile index> <layer index> <psnr>` Sets the target pSNR, in dB, for a specific quality layer of the selected tile.

This option calls the `aw_j2k_set_output_j2k_layer_psnr_ex` function, described on page 137.

`--set-output-j2k-channel-quantization <channel index> <quantization type>` Sets the quantization type for a channel of a JPEG2000 image. The channel index may be "`ALL`" to set the quantization for all channels. The supported quantization types are:

`REVERSIBLE` Reversible quantization (used only with the 5-3 wavelet)

`DERIVED` Derived irreversible quantization (used with the 9-7 wavelet)

`EXPOUNDED` Expounded irreversible quantization (used with the 9-7 wavelet)

This option calls the `aw_j2k_set_output_j2k_channel_quantization` function, described on page 140.

`-q <channel index> <quantization type>` Sets the quantization type for a channel of a JPEG2000 image. The channel index may be "`ALL`" to set the quantization for all channels. The supported quantization types are:

`REVERSIBLE` Reversible quantization (used only with the 5-3 wavelet)

`DERIVED` Derived irreversible quantization (used with the 9-7 wavelet)

`EXPOUNDED` Expounded irreversible quantization (used with the 9-7 wavelet)

This option calls the `aw_j2k_set_output_j2k_channel_quantization` function, described on page 140.

`--quantization-type <channel index> <quantization type>` Sets the quantization type for a channel of a JPEG2000 image. The channel index may be "`ALL`" to set the quantization for all channels. The supported quantization types are:

`REVERSIBLE` Reversible quantization (used only with the 5-3 wavelet)

`DERIVED` Derived irreversible quantization (used with the 9-7 wavelet)

`EXPOUNDED` Expounded irreversible quantization (used with the 9-7 wavelet)

This option calls the `aw_j2k_set_output_j2k_channel_quantization` function, described on page 140.

`--set-output-j2k-quantization-ex <tile index> <channel index> <quantization type>` Sets the quantization type for a channel of a tile of a JPEG-2000 image. The tile index may be "`ALL`" to set the quantization for all the tiles; similarly, the channel index may be "`ALL`" to set the quantization for all channels. The supported quantization types are:

`REVERSIBLE` Reversible quantization (used only with the 5-3 wavelet)

`DERIVED` Derived irreversible quantization (used with the 9-7 wavelet)

`EXPOUNDED` Expounded irreversible quantization (used with the 9-7 wavelet)

This option calls the `aw_j2k_set_output_j2k_quantization_ex` function, described on page 140.

`--set-output-j2k-quantization-binwidth-scale <binwidth scale>` Scales the default binwidths by the factor `<binwidth scale>`. This scale is only applied to the derived and expounded quantization options.

This option calls the `aw_j2k_set_output_j2k_quantization_binwidth_scale` function, described on page 145.

`--binwidth-scale <binwidth scale>` Scales the default binwidths by the factor `<binwidth scale>`. This scale is only applied to the derived and expounded quantization options.

This option calls the `aw_j2k_set_output_j2k_quantization_binwidth_scale` function, described on page 145.

`--set-output-j2k-quantization-binwidth-scale-ex <tile> <channel> <binwidth scale>` Scales the default binwidths for the selected channel of a tile by the factor `<binwidth scale>`. This scale is only applied to the derived and expounded quantization options.

This option calls the `aw_j2k_set_output_j2k_quantization_binwidth_scale_ex` function, described on page 145.

`--set-output-j2k-guard-bits <guard bits>` Sets the number of guard bits, which are used to ensure the the block encoder does not overflow. The legal values for this number range from 0 to 7. The default is 2.

This option calls the `aw_j2k_set_output_j2k_guard_bits` function, described on page 147.

`-g <guard bits>` Sets the number of guard bits, which are used to ensure the the block encoder does not overflow. The legal values for this number range from 0 to 7. The default is 2.

This option calls the `aw_j2k_set_output_j2k_guard_bits` function, described on page 147.

`--guard-bits <guard bits>` Sets the number of guard bits, which are used to ensure the the block encoder does not overflow. The legal values for this number range from 0 to 7. The default is 2.

This option calls the `aw_j2k_set_output_j2k_guard_bits` function, described on page 147.

`--set-output-j2k-guard-bits-ex <tile index> <channel index> <guard bits>` Sets the number of guard bits, which are used to ensure the the block encoder does not overflow, for a specified channel of a tile. The legal values for this number range from 0 to 7. The default is 2.

This option calls the `aw_j2k_set_output_j2k_guard_bits_ex` function, described on page 147.

`--set-output-j2k-channel-precinct-size <channel index> <resolution index> <precinct height> <precinct width>` Sets the dimensions of the precincts in a JPEG2000 image. `<Channel index>` can either be "`ALL`" for all channels or the index of a channel for which the precinct sizes are set. `<Resolution index>` runs from 0 for the lowest resolution to one more than the number of transform levels. `<Precinct height>` and `<precinct width>` are the base-2 logarithm of the desired precinct sizes. They can range from 0 to 15. Precinct size of 0 is only allowed for resolution level 0.

This option calls the `aw_j2k_set_output_j2k_channel_precinct_size` function, described on page 149.

`-P <channel index> <resolution index> <precinct height> <precinct width>` Sets the dimensions of the precincts in a JPEG2000 image. `<Channel index>` can either be "`ALL`" for all channels or the index of a channel for which the precinct sizes are set. `<Resolution index>` runs from 0 for the lowest resolution to one more than the number of transform levels. `<Precinct height>` and `<precinct width>` are the base-2 logarithm of the desired precinct sizes. They can range from 0 to 15. Precinct size of 0 is only allowed for resolution level 0.

This option calls the `aw_j2k_set_output_j2k_channel_precinct_size` function, described on page 149.

`--precinct <channel index> <resolution index> <precinct height> <precinct width>` Sets the dimensions of the precincts in a JPEG2000 image. `<Channel index>` can either be "`ALL`" for all channels or the index of a channel for which the precinct sizes are set. `<Resolution index>` runs from 0 for the lowest resolution to one more than the number of transform levels. `<Precinct height>` and `<precinct width>` are the base-2 logarithm of the desired precinct sizes. They can range from 0 to 15. Precinct size of 0 is only allowed for resolution level 0.

This option calls the `aw_j2k_set_output_j2k_channel_precinct_size` function, described on page 149.

`--set-output-j2k-channel-precinct-size-ex <tile index> <channel index> <resolution index> <precinct height> <precinct width>` Sets the dimensions of the precincts for a channel in a tile. `<Resolution index>` runs from 0 for the lowest resolution to one more than the number of transform levels. `<Precinct height>` and `<precinct width>` are the base-2 logarithm of the desired precinct sizes. They can range from 0 to 15. Precinct size of 0 is only allowed for resolution level 0.

This option calls the `aw_j2k_set_output_j2k_channel_precinct_size_ex` function, described on page 149.

`--set-output-j2k-codeblock-size <codeblock height> <codeblock width>` Sets the dimensions of the codeblocks. The values for `<codeblock height>` and `<codeblock width>` are the base-2 logarithm of the actual codeblock sizes. Each of height and width may range from 2 to 10, and their sum must be less than or equal to 12.

This option calls the `aw_j2k_set_output_j2k_codeblock_size` function, described on page 152.

`-cb <codeblock height> <codeblock width>` Sets the dimensions of the codeblocks. The values for `<codeblock height>` and `<codeblock width>` are the base-2 logarithm of the actual codeblock sizes. Each of height and width may range from 2 to 10, and their sum must be less than or equal to 12.

This option calls the `aw_j2k_set_output_j2k_codeblock_size` function, described on page 152.

`--codeblock-size <codeblock height> <codeblock width>` Sets the dimensions of the codeblocks. The values for `<codeblock height>` and `<codeblock width>` are the base-2 logarithm of the actual codeblock sizes. Each of height and width may range from 2 to 10, and their sum must be less than or equal to 12.

This option calls the `aw_j2k_set_output_j2k_codeblock_size` function, described on page 152.

`--set-output-j2k-codeblock-size-ex <tile index> <channel index> <codeblock height> <codeblock width>` Sets the dimensions of the codeblocks in a channel of a tile. The values for `<codeblock height>` and `<codeblock width>` are the base-2 logarithm of the actual codeblock sizes. Each width¿of height and width may range from 2 to 10, and their sum must width¿be less than or equal to 12.

This option calls the `aw_j2k_set_output_j2k_codeblock_size_ex` function, described on page 152.

`--context-reset <context reset frequency>` Sets how frequently the arithmetic coder is reset. The allowed values are:

`BLOCK` only at the end of codeblock

`PASS` at the end of each coding pass

Reset at end of codeblock is the default

This option calls the `aw_j2k_set_output_j2k_arithmetic_coding_option` function, described on page 154.

`--ac-term <arithmetic termination frequency>` Sets how frequently the arithmetic coder is terminated. The allowed values are:

`BLOCK` only at the end of codeblock

`PASS` at the end of each coding pass

Termination at end of codeblock is the default

This option calls the `aw_j2k_set_output_j2k_arithmetic_coding_option` function, described on page 154.

`--pred-term <predictable termination flag>` Determines whether predictable termination will be used. The allowed values are:

`YES` use predictable termination

`NO` do not use predictable termination (use normal termination)

Normal termination is the default.

This option calls the `aw_j2k_set_output_j2k_arithmetic_coding_option` function, described on page 154.

`--vert-causal <vertically causal context flag>` Sets whether the aritmetic coder's context formation will be vertically causal. The allowed values are:

`YES` use vertically causal context formation

`NO` do not use vertically causal context formation (use normal context formation)

Normal context formation is the default.

This option calls the `aw_j2k_set_output_j2k_arithmetic_coding_option` function, described on page 154.

`--set-output-j2k-arithmetic-coding-option <option> <value>` Sets any of the arithmetic coding options. The following options, and their corresponding values, are supported:

`CONTEXT` sets how frequently the arithmetic coder is reset. The allowed values are:

`BLOCK` only at the end of codeblock

`PASS` at the end of each coding pass

Reset at end of codeblock is the default

`ARITH` sets how frequently the arithmetic coder is terminated. The allowed values are:

`BLOCK` only at the end of codeblock

`PASS` at the end of each coding pass

Termination at end of codeblock is the default

`PRED` determines whether predictable termination will be used. The allowed values are:

`YES` use predictable termination

`NO` do not use predictable termination (use normal termination)

Normal termination is the default.

`VERT` sets whether the aritmetic coder's context formation will be vertically causal. The allowed values are:

`YES` use vertically causal context formation

`NO` do not use vertically causal context formation (use normal context formation)

Normal context formation is the default.

This option calls the `aw_j2k_set_output_j2k_arithmetic_coding_option` function, described on page 154.

`--set-output-j2k-arithmetic-coding-option-ex <tile index> <channel index> <option> <value>` Sets any of the arithmetic coding options for a channel of a tile. The following options, and their corresponding values, are supported:

`CONTEXT` sets how frequently the arithmetic coder is reset. The allowed values are:

`BLOCK` only at the end of codeblock

`PASS` at the end of each coding pass

Reset at end of codeblock is the default

`ARITH` sets how frequently the arithmetic coder is terminated. The allowed values are:

`BLOCK` only at the end of codeblock

`PASS` at the end of each coding pass

Termination at end of codeblock is the default

`PRED` determines whether predictable termination will be used. The allowed values are:

`YES` use predictable termination

`NO` do not use predictable termination (use normal termination)

Normal termination is the default.

`VERT` sets whether the aritmetic coder's context formation will be vertically causal. The allowed values are:

`YES` use vertically causal context formation

`NO` do not use vertically causal context formation (use normal context formation)

Normal context formation is the default.

This option calls the `aw_j2k_set_output_j2k_arithmetic_coding_option_ex` function, described on page 154.

`--set-output-j2k-coding-predictor-offset <predictor offset>` Sets the offset from the predicted point at which coding will stop. A smaller number will result in faster compression, though with a possible reduction in quality. The default value is 2; a value of 1 will speed up compression and reduce quality; 3 will reduce speed and increase quality; 0 will cause the library to code everything, resulting in the lowest speed but best quality.

This option calls the `aw_j2k_set_output_j2k_coding_predictor_offset` function, described on page 158.

`--predictor-offset <predictor offset>` Sets the offset from the predicted point at which coding will stop. A smaller number will result in faster compression, though with a possible reduction in quality. The default value is 2; a value of 1 will speed up compression and reduce quality; 3 will reduce speed and increase quality; 0 will cause the library to code everything, resulting in the lowest speed but best quality.

This option calls the `aw_j2k_set_output_j2k_coding_predictor_offset` function, described on page 158.

`--set-output-j2k-coding-predictor-offset-ex <tile index> <predictor offset>` Sets the offset from the predicted point at which coding will stop for the given tile. A smaller number will result in faster compression, though with a possible reduction in quality. The default value is 2; a value of 1 will speed up compression and reduce quality; 3 will reduce speed and increase quality; 0 will cause the library to code everything, resulting in the lowest speed but best quality.

This option calls the function `aw_j2k_set_output_j2k_coding_predictor_offset_ex`, described on page 158.

`--set-output-j2k-component-registration <channel> <x registration offset> <y registration offset>` Sets a color channel's registration offset in a JPEG2000 image. The registration offsets indicate how much to offset each color channel relative to the image grid for display. The offsets are given in units of 1/65536 of a pixel.

This option calls the `aw_j2k_set_output_j2k_component_registration` function, described on page 160.

`--set-component-registration <channel> <x registration offset> <y registration offset>` Sets a color channel's registration offset in a JPEG2000 image. The registration offsets indicate how much to offset each color channel relative to the image grid for display. The offsets are given in units of 1/65536 of a pixel.

This option calls the `aw_j2k_set_output_j2k_component_registration` function, described on page 160.

`--set-output-j2k-tile-toc` Adds to the JPEG2000 image a Tile Length Marker (TLM) segment.

This option calls the `aw_j2k_set_output_j2k_tile_toc` function, described on page 161.

`--add-tile-toc` Adds to the JPEG2000 image a Tile Length Marker (TLM) segment.

This option calls the `aw_j2k_set_output_j2k_tile_toc` function, described on page 161.

`--set-output-j2k-tile-data-toc` Adds to the JPEG2000 image a Packet Length, Tilepart Header Marker (PLT) segment.

This option calls the `aw_j2k_set_output_j2k_tile_data_toc` function, described on page 162.

`--add-tile-data-toc` Adds to the JPEG2000 image a Packet Length, Tilepart Header Marker (PLT) segment.

This option calls the `aw_j2k_set_output_j2k_tile_data_toc` function, described on page 162.

`--set-output-j2k-tile-data-toc-ex <tile index> <inclusion>` Adds to the specified tile of a JPEG2000 image a Packet Length, Tilepart Header Marker (PLT) segment.

This option calls the `aw_j2k_set_output_j2k_tile_data_toc_ex` function, described on page 162.

`--set-output-j2k-header-option <header> <end_marker>` Determines whether the output j2k code stream will have a header and an end of stream marker. By default both of the parameters are true.

This option calls the `aw_j2k_set_output_j2k_header_option` function, described on page 166.

`--set-output-j2k-clear-comments` Clears all comments from a JPEG2000 image.

This option calls the `aw_j2k_set_output_j2k_clear_comments` function, described on page 163.

`--output-reset-options` Resets all of the options set by the output image functions.

This option calls the `aw_j2k_output_reset_options` function, described on page 167.

`--set-output-j2k-roi-from-image <ROI image file name>` Defines a region of interest from an image. The image is input to the library through the memory-buffer interface, and its type is automatically determined.

This option calls the `aw_j2k_set_output_j2k_roi_from_image` function, described on page 170.

`--roi-image <ROI image file name>` Defines a region of interest from an image. The image is input to the library through the memory-buffer interface, and its type is automatically determined.

This option calls the `aw_j2k_set_output_j2k_roi_from_image` function, described on page 170.

`--set-output-j2k-roi-from-image-type <ROI image file name> <image type>` Defines a region of interest from an image. The image is input to the library through the memory-buffer interface, and its type is as given.

This option calls the `aw_j2k_set_output_j2k_roi_from_image_type` function, described on page 171.

`--set-output-j2k-roi-from-image-raw <image file name> <rows> <columns> <bit_depth> [<interleaved>]` Defines a region of interest from an image. The raw image is input to the library through the memory-buffer interface. The size of the image is specified using `<rows>`, `<columns>`, and `<bit_depth>`. Only one channel of the image is read.

This option calls the `aw_j2k_set_output_j2k_roi_from_image_raw` function, described on page 173.

`--set-output-j2k-roi-from-file <ROI image file name>` Defines a region of interest from an image. The image is input to the library through the file interface, and its type is automatically determined.

This option calls the `aw_j2k_set_output_j2k_roi_from_file` function, described on page 175.

`--set-output-j2k-roi-from-file-raw <ROI image file name> <rows> <columns> <bit_depth> [<interleaved>]` Defines a region of interest from an image. The raw image is input to the library through the file interface. The size of the image is specified using `<rows>`, `<columns>`, and `<bit_depth>`. Only one channel of the image is read.

This option calls the `aw_j2k_set_output_j2k_roi_from_file_raw` function, described on page 176.

`--roi-rect <left> <top> <width> <height>` A filled rectangle is added to the region of interest.

This option calls the `aw_j2k_set_output_j2k_roi_add_shape` function, described on page 177.

`--roi-ellipse <left> <top> <width> <height>` A filled ellipse is added to the region of interest.

This option calls the `aw_j2k_set_output_j2k_roi_add_shape` function, described on page 177.

`--set-output-j2k-roi-add-shape <shape> <left> <top> <width> <height>` A filled shape is added to the region of interest. The shapes are:

`RECTANGLE`

`ELLIPSE`

This option calls the `aw_j2k_set_output_j2k_roi_add_shape` function, described on page 177.

`--set-output-j2k-clear-roi` Clears all region of interest information.

This option calls the `aw_j2k_set_output_j2k_clear_roi` function, described on page 178.

`--set-output-jpg-options <quality>` Sets the quality parameter for JPEG compression (not JPEG2000). The quality parameter ranges from 0 to 100.

This option calls the `aw_j2k_set_output_jpg_options` function, described on page 185.

`--quality <quality>` Sets the quality parameter for JPEG compression (not JPEG-2000). The quality parameter ranges from 0 to 100. Use the value of "-1" to request lossless compression.

This option calls the `aw_j2k_set_output_jpg_options` function, described on page 185.

`--set-output-com-geometry-rotation <angle>` Before compression, the input image is rotated by `<angle>` degrees. `<Angle>` must be an whole multiple of 90.

This option calls the `aw_j2k_set_output_com_geometry_rotation` function, described on page 186.

`--rotate <angle>` Rotates the input image by `<angle>` degrees. `<Angle>` must be an whole multiple of 90.

This option calls the `aw_j2k_set_output_com_geometry_rotation` function, described on page 186.

`--set-output-raw-endianness <endianness>` Sets the endianness of an output raw image. The endianness values are:

`BIG`

`LITTLE`

This option calls the `aw_j2k_set_output_raw_endianness` function, described on page 186.

`--set-input-j2k-error-handling <mode>` Sets the error handling mode of the JPEG2000 decoder. The modes are:

`RESILIENT` attempts to recover from errors

`STRICT` stops decoding on the first error detection

This option calls the `aw_j2k_set_input_j2k_error_handling` function, described on page 83.

`--get-input-j2k-decoder-status` Returns the status of the JPEG2000 decoder, which will signal whether a JPEG 2000 image was successfully decoded, or whether errors were found

This option calls the `aw_j2k_get_input_j2k_decoder_status` function, described on page 71.

`--set-output-jp2-add-metadata-box <type> <data file name> [<uuid file name>]` Add the data found in `<data file name>` to a metadata box of a JP2 image. `<type>` is one of:

`JP2I` to add intellectual property metadata

`XML` to add XML

`UUID` to add an UUID box

The `<uuid file name>` argument should be used only if adding a UUID box. The UUID contained in that file must be 16 bytes long.

This option calls the `aw_j2k_set_output_jp2_add_metadata_box` function, described on page 182.

`--set-output-jp2-add-UUIDInfo-box <UUID list file name> <url>` Adds a UUID Information box to a JP2 image. `<UUID list file>` is a file containing a list of the UUIDs to add to the box. Each `<uuid>` is 16 bytes long.

This option calls the `aw_j2k_set_output_jp2_add_UUIDInfo_box` function, described on page 184.

`--set-output-enum-colorspace <color space>` Sets the color space of the output image, which can be one of:

`GREYSCALE`

`sRGB`

`sYCC`

`DEFAULT`

The default action is to set the color space to greyscale for single channel images, and sRGB for multi-channel images.

This option calls the `aw_j2k_set_output_enum_colorspace` function, described on page 180.

`--set-output-jp2-restricted-ICCProfile <ICC Profile file>` Reads a restricted ICC color space profile from the given file and uses it to set the output color space of a JP2 image, without performing any color conversions.

This option calls the `aw_j2k_set_output_jp2_restricted_ICCProfile` function, described on page 181.

`--get-output-j2k-layer-psnr-estimate <layer index> <psnr>` Returns an estimate of the pSNR of the specified layer of the JPEG2000 output image. Note that this option must be used after the -o option, and can only be used with output image types of JPEG2000, JP2 and DICOM J2K.

This option calls the `aw_j2k_get_output_j2k_layer_psnr_estimate` function, described on page 164.

`--get-input-jp2-num-metadata-boxes <box type>` Prints the count of metadata box of the given type in an input JP2 image. The `<box type>` can be one of:

`JP2I` for Intellectual Property boxes;

`XML` for XML boxes;

`UUID` for UUID boxes; or

`UUIDINFO` for UUID Information boxes.

This option calls the `aw_j2k_get_input_jp2_num_metadata_boxes` function, described on page 86.

`--get-input-jp2-metadata-box <box type> <box index> [<data file name> [<UUID file name>]]` Prints, or optionally stores to the named files, the contents of the box of the given type and index. The `<box type>` can be one of:

`JP2I` for Intellectual Property boxes;

`XML` for XML boxes;

`UUID` for UUID boxes; or

The `<UUID file name>` may only be supplied if the `<box type>` is "uuid".

This option calls the `aw_j2k_get_input_jp2_metadata_box` function, described on page 87.

`--get-input-jp2-UUIDInfo-box <box index> [<UUID file name> [<URL file name>]]` Prints, or optionally stores to the named files, the contets of the UUID Information box denoted by the given index. If a single file name is given, The UUID list is stored to that file and the URL is printed to screen.

This option calls the `aw_j2k_get_input_jp2_UUIDInfo_box` function, described on page 89.

`--get-input-jp2-enum-colorspace` Prints the color space of a JP2 input image, if the color space is stored as an enumerated color space.

This option calls the `aw_j2k_get_input_jp2_enum_colorspace` function, described on page 91.

`--get-input-jp2-restricted-ICCProfile [<file name>]` Prints, or optionally stores to the named file, the color space of a JP2 input image, if the color space is stored as a Restricted ICC Profile.

This option calls the `aw_j2k_get_input_jp2_restricted_ICCProfile` function, described on page 92.

`--verbose` Increases the verbosity level of the program.

`-v` Increases the verbosity level of the program.

`--benchmark [<repetitions>]` Enables benchmarking mode.

`--verbose-to-stdout` Redirects all verbose and diagnostic output from stderr to stdout. Note that this disables the use of stdout for retrieving images.

Appendix A

ActiveX Control Interface

In addition to the C Library interface, Aware offers a non-visual ActiveX control for JPEG 2000 image compression/decompression. The control allows Visual Basic the full functionality of the C library. In addition, the control provides a method to export JPEG 2000 images via the Visual Basic picture object.

The usage of the ActiveX through Visual Basic is demonstrated in the example Visual Basic application included in the SDK. The source code for the application is installed in `<InstallDir>/Example Files/AwJ2KActiveX Demo`

The interface to the control is similar to the C Interface. The main differences are:

1. The method calls for the control are not prefixed with `aw_j2k_`
2. The method calls omit the `aw_j2k_object` parameter
3. The `aw_j2k_create` and `aw_j2k_destroy` function do not need to be called
4. The buffer size and buffer pointer parameters are replaced with a single variant parameter which accepts an array of byte.

In addition, the control offers the function, `get_output_vb_picture`, that returns a Visual Basic picture object. The output of this function can be assigned to the.picture property of another control (e.g. picturebox or image control).

Appendix B

Error Codes

The functions in the library will return `NO_ERROR` (0) upon successful completion. A non-zero return value indicates an error. Error codes are as follows:

`AW_J2K_ERROR_MALLOC_ERROR` This error code is returned when a memory allocation error occurs in the library.

`AW_J2K_ERROR_UNSUPPORTED_JP2K_TYPE_ERROR` This error code is returned when a JPEG 2000 image with unsupported options or features is encountered. Bit depths over 16 bits per channel are not supported at this time.

`AW_J2K_ERROR_MISSING_COD_MARKER` This error code is returned when a JPEG 2000 image with a missing COD marker is supplied as an input image. One COD marker must appear in the main header of the image. The COD marker specifies the coding style, wavelet transform type, codeblock sizes, precinct sizes, number of decomposition levels and number of layers.

`AW_J2K_ERROR_MISSING_QCD_MARKER` This error code is returned when a JPEG 2000 image with a missing QCD marker is supplied as an input image. One QCD marker must appear in the main header of the image. The QCD marker defines the type of quantization and the quantization step sizes.

`AW_J2K_ERROR_NULL_DATA` This error code is returned when null data is encountered in either an image or memory buffer supplied to the library.

`AW_J2K_ERROR_BYTESTREAM_ERROR` This error code is returned when the library encounters an illegal JPEG 2000 image. For example, if the order of the markers in an image is incorrect, this error would be returned.

`AW_J2K_ERROR_USER_ABORT` This error code is returned when the TestAbortProc callback returns non-zero. See the documentation of `aw_j2k_register_callbacks` on page 31 for more details.

`AW_J2K_ERROR_OUT_OF_DATA_ERROR` This error code is returned when the library encounters a truncated packet in a JPEG 2000 image. This error is only returned when the error handling is set to strict mode. For details on error handling modes, go to the description of the function `aw_j2k_set_input_j2k_error_handling` on page 83.

`AW_J2K_ERROR_SHORT_DATA_ERROR` This error code is returned when the user attempts to decode to a progression level (or to decode a tile) which is not available in the JPEG 2000 input image.

`AW_J2K_ERROR_BAD_PRECINCT_SPECIFICATION` This error code is returned when the an illegal precinct specification is encountered. For example, this error would be returned when the user attempts to set the precinct size for the third resolution level in an image with only one decomposition level.

`AW_J2K_ERROR_INVALID_OBJECT` This error code is returned by all functions (except `aw_j2k_create`) when the user passes a pointer to a null or image invalid image object as an argument.

`AW_J2K_ERROR_BUFFER_TOO_SHORT` This error code is returned when the length of the output buffer supplied by the user is insufficient to hold the library output.

`AW_J2K_ERROR_FILE_OPEN` This error code is returned when the library fails to open the specified input file.

`AW_J2K_ERROR_FILE_CREATE` This error code is returned when the library fails to create the specified output file.

`AW_J2K_ERROR_FILE_READ` This error code is returned when the library encounters an error while reading bytes from the specified input file.

`AW_J2K_ERROR_FILE_WRITE` This error code is returned when the library encounters an error while writing bytes to the specified output file.

`AW_J2K_ERROR_INPUT_FORMAT_MISMATCH` This error occurs when the user attempts an operation that can't be performed on the given input image. A common example is calling a JPEG 2000 specific information function on a non JPEG 2000 input image.

`AW_J2K_ERROR_INPUT_NOTSET` This error is returned when the user calls a function that requires an input image to be available in the image object, before one has been set.

`AW_J2K_ERROR_UNKNOWN_IMAGE_TYPE` This error occurs when the library fails to automatically identify the type of the input image.

AW_J2K_ERROR_UNSUPPORTED_IMAGE_TYPE This error occurs if the library identifies the type of the input image, yet determines that the file is an unsupported variant of the given image type. For example, BMP images with Runlength coding are not supported.

AW_J2K_ERROR_INVALID_IMAGE This error occurs if the library identifies the type of the input image, but encounters an error while parsing the file header.

AW_J2K_ERROR_INVALID_PARAMETER This is a generic error code returned when a parameter passed to a function is not a legitimate value for the given operation.

AW_J2K_JP2_PARSE_ERROR This error code is returned when the library encounters an error while parsing the JP2 header.

AW_J2K_JP2_NO_ICCPROFILE This error code is returned by `aw_j2k_get_input_jp2_restricted_ICCProfile` if the input JP2 file does not have an embedded ICC profile.

AW_J2K_JP2_NO_ENUM_COLORSPACE This error code is returned by `aw_j2k_get_input_jp2_enum_colorspace` if the input JP2 file does not use an enumerated colorspace.

AW_J2K_ERROR_DICOM_LIBRARY This error code is returned if the library encounters an error during an I/O operation on a DICOM file.

AW_J2K_ERROR_INVALID_CALL_SEQUENCE This error code is returned by `aw_j2k_get_output_j2k_layer_psnr_estimate` if the function is called before a `get_output` function is called.

AW_J2K_ERROR_UNSUPPORTED_FUNCTION This error code is returned if the current build of the library does not support the function called by the user.

AW_J2K_ERROR_MEMORYMAP_OUTPUT The library sometimes uses memory mapping of the input file for file I/O efficiency. This error code is returned when the input image memory mapping fails.

AW_J2K_ERROR_MEMORYMAP_INPUT The library sometimes uses memory mapping of the output file for file I/O efficiency. This error code is returned when the output image memory mapping fails.

AW_J2K_ERROR_TIFF This error code is returned when the library encounters an error while parsing a TIFF file header or TIFF compressed stream.

AW_J2K_ERROR_LIBRARY_EXPIRED This error code is returned only by the expiring version of the Aware J2K library. This error code indicates that the library has expired, and all the functionalities are disabled.

AW_J2K_ERROR_LIBRARY_INTERNAL This error code is returned when an internal data structure inconsistency is detected by the library.