Acquisition Module Interface for Adobe Photoshop™

Introduction

This document describes version 4 of Adobe Photoshop's acquisition module interface. Plug-ins based on the following specifications should contain a 4 in their 'PiMI' resource.

If the plug-in does not use the fields after the version 4 fields comment, it may choose to label itself a version 3 plug-in, to remain compatible with older versions of host software.

The code resource and file type for acquisition modules is '8BAM'.

Record Structure

The *stuff* parameter contains a pointer to a structure of the following type:

```
AcquireRecord = RECORD

    serialNumber:    LONGINT;
    abortProc:       ProcPtr;
    progressProc:    ProcPtr;
    maxData:         LONGINT;
    imageMode:       INTEGER;
    imageSize:       Point;
    depth:           INTEGER;
    planes:          INTEGER;
    imageHRes:       Fixed;
    imageVRes:       Fixed;
    redLUT:          PACKED ARRAY [0..255] OF CHAR;
    greenLUT:        PACKED ARRAY [0..255] OF CHAR;
    blueLUT:         PACKED ARRAY [0..255] OF CHAR;
    data:            Ptr;
    theRect:         Rect;
    loPlane:         INTEGER;
    hiPlane:         INTEGER;
    colBytes:        INTEGER;
    rowBytes:        LONGINT;
    planeBytes:      LONGINT;
    filename:        Str255;
    vRefNum:         INTEGER;
    dirty:           BOOLEAN;

    { Version 4 fields }

    hostSig:         OSType;
    hostProc:        ProcPtr;
```

```
    hostModes:      LONGINT;
    planeMap:       ARRAY [0..15] OF INTEGER;
    canTranspose:   BOOLEAN;
    needTranspose:  BOOLEAN;
    duotoneInfo:    Handle;
    diskSpace:      LONGINT;
    spaceProc:      ProcPtr;
    monitor:        AcquireMonitor;

    reserved:       PACKED ARRAY [0..255] OF CHAR;

    END;

AcquireMonitor = RECORD
    gamma:      Fixed;                                  redX:
    Fixed;      redY:                                   Fixed;
    greenX:     Fixed;
    greenY:     Fixed;                                  blueX:
    Fixed;      blueY:                                  Fixed;
    whiteX:     Fixed;
    whiteY:     Fixed;
    ambient:    Fixed;                                  END;
```

Calling Order

When the user invokes the plug-in by selecting its name from the Acquire submenu, Photoshop loads the plug-in's resource into memory and calls it with the following sequence of *selector* values (see the header file for their actual values):

(1) *acquireSelectorPrepare*

This allows the plug-in to adjust Photoshop's memory allocation algorithm. Before this call, Photoshop sets the *maxData* field to the maximum number of bytes it would be able to free up. If plug-in module then has the option of reducing this number during this call. Reducing the number can speed up operation, since freeing up the maximum amount of memory requires Photoshop to move all of the image data for any currently open images out of of RAM and into its virtual memory file. Furthermore, in order to keep this amount of memory free, Photoshop is required to write any newly acquired image data to the virtual memory file as it is received.

If the plug-in knows that it memory requirements will be limited (if it can return the image data in strips, or if the maximum resolution image it can return is small), it should reduce *maxData* to its actual requirements during this call. This will allow small acquisitions to be performed entirely in RAM.

If, as is often the case, the plug-in only needs a small amount memory, but will operate faster if given more, a tradeoff has to be made. One solution is to divide the *maxData* field by 2, thus allocating half the memory to Photoshop and half to the plug-in.

(2) *acquireSelectorStart*

This call lets Photoshop know the mode, size and resolution of the image being returned, so it can allocate and initialize its data structures. Most plug-ins will display their dialog box, if any, during this call.

During this call, the plug-in module should set the *imageMode*, *imageSize*, *depth*, *planes*, *imageHRes* and *imageVRes* fields. If an indexed color image is being returned, it should also set the *redLUT*, *greenLUT* and *blueLUT* fields. If a duotone mode image is being returned, it should also set the *duotoneInfo* field.

(3) *acquireSelectorContinue*

This call returns an area of the image to Photoshop. Photoshop will continue to call this routine until it either returns an error, or sets the *data* field to NIL.

The area of the image being returned is specified by *theRect* and by the *loPlane* and *hiPlane* fields. The *data* field contains a pointer to the actual data being returned. The *colBytes*, *rowBytes* and the *planeBytes* fields specify the organization of the data.

Photoshop is very flexible in the format in which image data can be returned. For example, to return just the red plane of an RGB color image, the *loPlane* and the *hiPlane* fields should be set to 0, the *colBytes* fields should be set to 1, and the *rowBytes* field should be set to the width of the area being returned (the *planeBytes* field is ignored in this case, since *loPlane* = *hiPlane*).

If instead, you wish to return the RGB data in interleaved form (RGBRGB...), the *loPlane* should be set to 0, *hiPlane* to 2, *planeBytes* to 1, *colBytes* to 3, and *rowBytes* to 3 times the width of area being returned.

The area of the image being returned is specified by the *theRect* field. If the resolution of the acquired image is always going to be very small (i.e. NTSC frame grabbers), the plug-in can simply set *theRect* to the entire image area. However, if you wish to be able to scan large images, the plug-in must use the *theRect* field to return the image in strips. There are no restrictions on how the strips tile the image (i.e. vertical strips are allowed), but horizontal strips are most efficient when spooling to the virtual memory file. Each strip should contain no more than *maxData* bytes (less the size of any large tables or scratch areas allocated by the plug-in).

The *data* field contains a pointer to the data being returned. Most plug-in will allocate a buffer for the data using the NewPtr trap. The plug-in is responsible for freeing this buffer in the *acquireSelectorFinish* call. (Note: this is a change from pre-version 3 interfaces, which freed the block for the plug-in!)

(4) *acquireSelectorFinish*

This call allows the plug-in to clean up after an image acquisition. This call is made *if and only if* the *acquireSelectorStart* routine returns without error (even if the *acquireSelectorContinue* routine returns an error).

Most plug-in will at least need to free the buffer used to return the image data.

If Photoshop detects a command-period while processing the results of *acquireSelectorContinue* call, it

will call the *acquireSelectorFinish* routine (be careful here, since normally the plug-in would be expecting another *acquireSelectorContinue* call).

Record Fields

(1) *serialNumber*

Contains Adobe Photoshop's serial number. Plug-in modules can use this value for copy protection, if desired.

(2) *abortProc*

Contains a pointer to a function with the following Pascal calling conventions:

```
FUNCTION TestAbort: BOOLEAN;
```

The plug-in may call this function several times a second during long operations to allow the user to abort the operation. If the function returns TRUE, the operations should be aborted. As a side effect, this changes to cursor to a watch, and moves the watch hands periodically.

(3) *progressProc*

Contains a pointer to a procedure with the following Pascal calling conventions:

```
PROCEDURE UpdateProgress (done, total: LONGINT);
```

The plug-in may call this two-argument procedure periodically to update a progress indicator. The first parameter is the number of operations completed; the second is the total number of operations.

This procedure should only be called during the actual main operation of the plug-in, not during long operations during the preliminary user interface. For example, it should not be used during a preview operation that computes a low resolution preview image for cropping. It should be used during the main, high-resolution scan.

Photoshop automatically suppresses display of the progress graph during short operations.

(4) *maxData*

Photoshop initializes this field to the maximum of number of bytes it can free up. The plug-in may reduce this value during the *acquireSelectorPrepare* routine. The *acquireSelectorContinue* routine should return the image in strips no larger than *maxData*, less the size of any large tables or scratch areas it has allocated.

(5) *imageMode*

The *acquireSelectorStart* routine should set this field to inform Photoshop what mode image is being acquired (Gray Scale, RGB Color, etc.). See the header file for possible values.

(6) *imageSize*

The *acquireSelectorStart* routine should set this field to inform Photoshop of the image's width *(imageSize.h)* and height *(imageSize.v)* in pixels.

(7) *depth*

The *acquireSelectorStart* routine should set this field to inform Photoshop of the image's resolution in bits per sample. The only valid settings are 1 for bitmap mode images, and 8 for all other modes.

(8) *planes*

The *acquireSelectorStart* routine should set this field to inform Photoshop of the number of channels in the image. For example, if an RGB image without alpha channels is being returned, this field should be set to 3.

(9) *imageHRes* and *imageVRes*

The *acquireSelectorStart* routine should set these fields to inform Photoshop of the image's horizontal and vertical resolution in terms of pixels per inch. This is a fixed point number (16 binary digits). Photoshop initializes these fields to 72 pixels per inch.

The current version of Photoshop only supports square pixels, so it ignores the imageVRes field. Plug-ins should set both fields anyway in case future versions of Photoshop support non-square pixels.

(10) *redLUT*, *greenLUT* and *blueLUT*

If an indexed color mode image is being returned, the *acquireSelectorStart* routine should return the image's color table in these fields.

(11) *data*

The *acquireSelectorContinue* routine should return a pointer to the image's data in this field. After all of the image has been returned, the *acquireSelectorContinue* should set this field to NIL.

Note that the plug-in is responsible of freeing any memory pointed to by this field. This is a change from previous version's of Photoshop's plug-in interface.

(12) *theRect*

The *acquireSelectorContinue* routine should set this field to the area being returned.

(13) *loPlane* and *hiPlane*

The *acquireSelectorContinue* routine should set these fields to the first and last planes being returned. For example, if interleaved RGB data is being returned, they should be set to 0 and 2, respectively.

(14) *colBytes*

The *acquireSelectorContinue* routine should set this field to the offset in bytes between columns of

returned data. This is usually 1 for non-interleaved data, or (hiPlane - loPlane + 1) for interleaved data.

(15) *rowBytes*

The *acquireSelectorContinue* routine should set this field to the offset in bytes between rows of returned data.

(16) *planeBytes*

The *acquireSelectorContinue* routine should set this field to the offset in bytes between planes of returned data. This field is ignored if *loPlane* = *hiPlane*. It should be set to 1 for interleaved data.

(17) *filename*

By default, Photoshop opens newly acquired images as "Untitled-..." . File importing acquisition modules should set this field to the file's name in their *acquireSelectorStart* routines, so Photoshop can display the correct window title. Scanning modules should ignore this field.

(18) *vRefNum*

If the plug-in sets the *filename* field, it should also the the *vRefNum* field to the file's volume reference number.

(19) *dirty*

By default, newly acquired images are marked as dirty, meaning that the user will be prompted to save the unsaved changes when closing the file. File importing acquisition modules should set this field to false to prevent this.

(20) *hostSig*

The host program's signature. Photoshop's signature is '8BIM'.

(21) *hostProc*

A host-defined callback procedure that can do anything the host wishes. Plug-ins should verify the *hostSig* before calling this procedure. This provides a mechanism for hosts to extend the plug-in interface to support application specific features.

(22) *hostModes*

Used by the host to inform the plug-in which *imageMode* values it supports. If the corresponding bit (LSB = bit 0) is 1, the mode is supported. This field can be used by plug-ins to disable features (such as color scanning) if not supported by the host.

(23) *planeMap*

This is initialized by the host to a linear map (*planeMap* [i] = i). This is used to map plane (channel) numbers between the plug-in and the host.

For example, Photoshop stores RGB images with an alpha channel in the order RGBA, whereas most frame buffers store the data in ARGB order. To return the data in this order, the plug-in should set planeMap [0] to 3, planeMap [1] to 0, planeMap [2] to 1, and planeMap [3] to 2.

(24) *canTranspose*

If the host supports transposing images during or after scanning, it should set this field to true. Photoshop always sets this field to true.

(25) *needTranspose*

Initialized by the host to false. If the plug-in wishes to have the image transposed, and the *canTranspose* field is true, it should set this field to true during its *acquireSelectorStart* routine.

The logical effect is to transpose the image after scanning is complete, although some hosts may find it more efficient to transpose the data during scanning.

Using this feature in usually much faster than returning the image in vertical strips.

(26) *duotoneInfo*

If the plug-in is acquiring a duotone mode image, it should allocate a handle and return the duotone information here. The format of the information is the same as that provided by export modules, and should be treated as a black box by plug-ins.

The plug-in is responsible for freeing the handle in its *acquireSelectorFinish* routine.

(27) *diskSpace*

The number free bytes on the host's scratch disk or disks. If the host does not use a scratch disk, it should set this field to -1.

(28) *spaceProc*

If not NIL, contains a pointer to a function with the following Pascal calling conventions:

```
FUNCTION SpaceProc: LONGINT;
```

This function examines *imageMode*, *imageSize*, *depth*, and *planes* fields and returns the number of bytes of scratch disk space required to hold the image. Returns -1 if the values are not valid.

(29) *monitor*

Information on the current monitor. The *gamma* field contains the monitor's gamma value, or zero if the entire *monitor* record is undefined. The *redX, redY, greenX, greenY, blueX, blueY* fields specify the chromaticity coordinates of the monitor's phosphors. The *whiteX* and *whiteY* fields specify the chromaticity coordinates of the monitor's white point. The *ambient* value specifies the relative amount of ambient light in the room. Zero means a relatively dark room, 0.5 means an average room, and 1.0

means a bright room.

(30) *reserved*

Set to zero by the host for future expansion of the plug-in standard. Must not be used by plug-ins.

Supporting Older Plug-in Versions

This section describes how hosts can be compatible with older versions of the plug-in specification. It is of interest only to authors of host software.

(1) If the version is less than 3, do not call the *acquireSelectorPrepare* routine.

(2) If the version is less than 3, the host is required to dispose of the returned data (using DisposPtr) instead of the plug-in.

All other changes are backward compatible.