8

Export Elements

Element categories

Export elements are used to send image data out of a Photoflo. Each export element has a single Phototag source but does not provide an output connection. There are two types of export elements:

- `ExportResource` elements export data to a server resource (`DRAWABLE`, **LUT**, **Photomap**, or **ROI**).
- `ExportClient` elements produce data for client retrieval. They provide the input data for the **GetClientData** protocol request.

Export to Client

`ExportClient` elements provide data for the client. Element parameters fully describe the format or organization the data is to have as it is transported through the protocol stream. The data itself must be retrieved using the **GetClientData** protocol request while the **Photoflo** is ***Active***.

Servers are required to buffer a non-zero amount of data per `ExportClient` element. Beyond that point execution may be suspended until the client retrieves sufficient data.

Events generated

`ExportClient` elements can generate **ExportAvailable** events to notify the client when the data is available for retrieval. **ExportAvailable** events may be:

- disabled (i.e. no event is sent),
- enabled only when data first becomes available from the element,
- enabled each time the element's data buffer transitions from empty to non-empty.

ExportClientHistogram

XieFloExportClientHistogram

notify	XieTypExportNotify
src	XieTypPhototag
domain	XieTypProcessDomain

Errors: ***FloAlloc, FloSource, FloDomain, FloMatch, FloValue***
Events: none

ExportClientHistogram generates a histogram of the pixel values found in the source data. It prepares histogram data that can be retrieved by the client using the **GetClientData** protocol request. An event can be requested that will notify the client when histogram data becomes available.

Notify allows the client to be notified when the histogram data becomes available. *Src* is the **Phototag** of the element supplying ***SingleBand Constrained*** source data. *Domain* may control the subset of the source data from which the distribution will be generated.

The data generated for the client is a list of **HistogramData** where each entry consists of an *value* (i.e. a pixel value) followed by the *count* of pixels found with that value. If the number of pixels for a given *value* exceeds the capacity of *count* (type `CARD32`), that *count* will be returned at the maximum value (i.e. 2^{32}-1). Pixel values that are not found in the data are not included in the histogram data (i.e. no entries are returned where *count* is zero).

If *notify* is true, the total number of histogram entries are reported in the *data* field of the **ExportAvailable** event.

The histogram is returned as numeric values (see **HistogramData**), as such, the byte order of the data is determined at core protocol connection setup time.

Error	Cause
FloAlloc	insufficient resources
FloSource	invalid *src*
FloDomain	invalid *domain*
FloMatch	***Unconstrained*** *src* data ***TripleBand*** *src* data
FloValue	invalid *notify*

All data that are generated by **ExportClientHistogram** must be retrieved by the client using **GetClientData** before the **Photoflo** can exit from the ***Active*** state.

Histograms of ***TripleBand*** images are not supported directly. However, **BandSelect** may be used to choose a single band of data, or **BandExtract** may be used to combine data from multiple bands into a single band of data. A histogram of this single band of data could then be generated using **ExportClientHistogram**.

ExportClientLUT

XieFloExportClientLUT

notify	XieTypExportNotify
src	XieTypPhototag
band-order	XieTypOrientation
start	XieTypTripletofCARD32
length	XieTypTripletofCARD32

Errors: ***FloAlloc, FloSource, FloValue***
Events: none

ExportClientLUT allows data imported from an **ImportLUT** or **ImportClientLUT** element to be retrieved by the client. The actual transport of lookup table data through the protocol stream is requested using the **GetClientData** protocol request.

Notify allows the client to be notified when data become available. *Src* is the **Phototag** of the element supplying lookup table data. *Band-order* is the order in which ***TripleBand*** arrays are transmitted through the protocol stream. *Start* is the index of the first array entry that should be returned. *Length* is the number of array entries that should be returned.

If *notify* is requested, the **ExportAvailable** event's data field will report the total number of array entries that can be retrieved from the band specified by the event.

The size of each array entry is either 1, 2, or 4 bytes; the smallest size into which the output quantization *levels* can be stored. When array entries require multiple bytes, the byte order per entry is determined in the same manner as other numeric data: it is the byte orientation established at core X connection setup time.

Error	Cause
FloAlloc	insufficient resources
FloSource	invalid *src*
FloMatch	*start* + *length* exceeds number of entries in an array
FloValue	invalid *notify* or *band-order*

When the client attempts to retrieve data from an **ExportClientLUT** element, the server responds with one or more complete array entries per protocol reply (see **GetClientData**). All data must be retrieved by the client before the **Photoflo** can leave the ***Active*** state.

ExportClientPhoto

XieFloExportClientPhoto

notify	XieTypExportNotify
src	XieTypPhototag
encode	XieTypEncodeTechnique
encode-params	<technique-dependent>

Errors: ***FloAlloc, FloSource, FloMatch, FloValue, FloTechnique***
Events: none

ExportClientPhoto makes image data available to the protocol stream. The attributes of the exported data are determined by the attributes of the source data. The format of the data is specified by the *encode* technique and *encode-params.* The actual transport of image data through the protocol stream is requested using the **GetClientData** protocol request.

Notify allows the client to be notified when image data become available. *Src* is the **Phototag** of the element supplying ***Constrained*** data. *Encode* is the **EncodeTechnique** that will be used to compress of format the exported data. *Encode-params* is the list of additional parameters required by *encode.*

Error	Cause
FloAlloc	insufficient resources
FloSource	invalid *src*
FloMatch	***Unconstrained*** *src* data
FloValue	invalid *notify*
FloTechnique	invalid *encode* technique or *encode-params*

ServerChoice is not a valid *encode* technique for **ExportClientPhoto**.

Depending on *encode*, the export process may cause the data to be decoded, encoded, or otherwise reformatted by the server. If **ExportClientPhoto** is fed directly from an **ImportClientPhoto** or **ImportPhotomap** element, and the attributes of the raw data that are given to the import element match the requirements of the *encode* technique, the raw data can bypass the normal decode function of the import element and be forwarded directly to **ExportPhotomap** (this is an implementation-specific optimization).

If ***TripleBand*** data are given to an *encode* technique that will interleave the output data ***BandByPixel***, the dimensions of each *src* band must match.

All data must be retrieved by the client, using the **GetClientData** protocol request, before the **Photoflo** can exit from the ***Active*** state.

ExportClientROI

XieFloExportClientROI

notify	XieTypExportNotify
src	XieTypPhototag

Errors: ***FloAlloc, FloSource, FloValue***
Events: none

ExportClientROI allows a *list-of-rectangles*, imported from an **ImportROI** or **ImportClientROI** element, to be retrieved by the client. The actual transport of lookup table data through the protocol stream is requested using the **GetClientData** protocol request.

Notify allows the client to be notified when data become available. *Src* is the **Phototag** of the element supplying the *list-of-rectangles*.

If *notify* is requested, the **ExportAvailable** event's *data* field will report the total number of rectangles that can be retrieved.

Each rectangle is described using numeric values (see **Rectangle**), as such, the byte order of the data is determined at core protocol connection setup time.

Error	Cause
FloAlloc	insufficient resources
FloSource	invalid *src*
FloValue	invalid *notify*

When the client attempts to retrieve data from an **ExportClientROI** element, the server responds with one or more complete **Rectangles** per protocol reply (see **GetClientData**). All data must be retrieved by the client before the **Photoflo** can leave the ***Active*** state.

Export to Resource

`ExportResource` elements emit data to server resources: either core X `DRAWABLE`s, or XIE **LUT**, **Photomap**, or **ROI** resources.

For `ExportResource` elements that assign attributes to XIE resources, the association between the resource identifier (`XID`) and the new attributes and data does not occur until the **Photoflo** successfully completes (i.e. leaves the ***Active*** state). (See "**LUT, Photomap, and ROI resources**" in Chapter 4.)

ExportDrawable

XieFloExportDrawable

src	XieTypPhototag
drawable	`DRAWABLE`
gc	`GCONTEXT`
dst-x	`INT16`
dst-y	`INT16`

Errors: ***FloAlloc, FloSource, FloDrawable, FloGC, FloMatch, FloValue***
Events: none

ExportDrawable allows `COLORMAP` index data to be exported to a `WINDOW` or `PIXMAP`.

Src is the **Phototag** of the element supplying ***Constrained*** source data (index data assumed). *Drawable* is the `WINDOW` or `PIXMAP` into which the data will be written. *Gc* is the `GCONTEXT` to be used when transferring pixels to *drawable*. *Dst-x* and *dst-y* specify where the data should be placed in *drawable*.

The following components are used from *gc*: `function`, `plane-mask`, `subwindow-mode`, `clip-x-origin`, `clip-y-origin`, and `clip-mask`.

The *levels* of *src* must exactly match the depth of *drawable* and *gc* (i.e. *levels* must be 2^{depth}).

Error	Cause
FloAlloc	insufficient resources
FloSource	invalid *src*
FloDrawable	invalid *drawable*
FloGC	invalid *gc*
FloMatch	invalid *src* data (***TripleBand***, *levels* doesn't match depth, or ***Unconstrained***)

ExportDrawable assumes its input is `COLORMAP` index data. Sources of such data include: **ImportClientPhoto**, **ImportDrawable**, **ImportPhotomap**, **BandExtract**, **ConvertToIndex**, **Point**, and elements that have processed index data.

ExportDrawablePlane

XieFloExportDrawablePlane

src	XieTypPhototag
drawable	`DRAWABLE`
gc	`GCONTEXT`
dst-x	`INT16`
dst-y	`INT16`

Errors: ***FloAlloc, FloSource, FloDrawable, FloGC, FloMatch, FloValue***
Events: none

ExportDrawablePlane allows ***SingleBand*** single bit (bitonal) data to be exported to a `WINDOW`, `PIXMAP`, or `BITMAP`.

Src is the **Phototag** of the element supplying ***Constrained*** bitonal source data. *Drawable* is the `WIN-DOW`, `PIXMAP`, or `BITMAP` into which the data will be written. *Gc* is the `GCONTEXT` to be used when transferring pixels to *drawable*. *Dst-x* and *dst-y* specify where the data should be placed in *drawable*.

The following components are used from *gc*: `function`, `plane-mask`, `foreground`, `back-ground`, `fill-style`, `subwindow-mode`, `clip-x-origin`, `clip-y-origin`, and `clip-mask`. For the `fill-style` component of *gc*, values of *`FillSolid`* and *`FillTiled`* are treated as synonyms for *`FillOpaqueStippled`*.

Error	Cause
FloAlloc	insufficient resources
FloSource	invalid *src*
FloDrawable	invalid *drawable*
FloGC	invalid *gc*
FloMatch	invalid *src* data (***TripleBand***, *levels* > 2, or not ***Constrained***)

Bitonal data can be exported to a `DRAWABLE` of any depth if an appropriate `GCONTEXT` is supplied. The `foreground` and `background` pixel values of *gc* can be used to map image pixels to `COLORMAP` colors during the export process.

ExportLUT

XieFloExportLUT

src	XieTypPhototag
lut	XieTypLUT
merge	BOOL
start	XieTypTripletofCARD32

Errors: ***FloAlloc, FloSource***
Events: none

ExportLUT allows data imported from an **ImportLUT** or **ImportClientLUT** element to be saved in an existing **LUT** resource.

Src is the **Phototag** of the element supplying lookup table data. *Lut* is the **LUT** to receive the data. *Merge* specifies that new array entries from *src* should replace entries that already exist within *lut*. *Start* is the index of the first array entry that should be written in *lut*.

If *merge* is false, *start* must be 0 for each band. In this case, *lut* will inherit the attributes of *src* and be populated with its data. The previous attributes and data of *lut* are overwritten when the **photoflo** completes.

If *merge* is true and *lut* has existing attributes, the data from *src* will replace the data from *lut*, beginning at position *start*. The attributes of *src* must match those of *lut*, and the combination of *start* and the length of *src* must specify a valid sub-range existing within *lut*. If *merge* is true but *lut* has not yet been populated, it is an error.

Error	Cause
FloAlloc	insufficient resources
FloSource	invalid *src*
FloLUT	invalid *lut*
FloMatch	*merge* true and attributes do not match between *src* and *lut* *merge* true and *start* + *src* length is not a sub-range of *lut*
FloValue	*merge* false and *start* is non-zero

ExportPhotomap

XieFloExportPhotomap

src	XieTypPhototag
photomap	XieTypPhotomap
encode	XieTypEncodeTechnique
encode-params	<technique-dependent>

Errors: ***FloAlloc, FloSource, FloPhotomap, FloTechnique***
Events: none

ExportPhotomap allows data resulting from **Photoflo** operations to be saved in an existing **Photomap**.

Src is the **Phototag** of the element supplying source data. *Photomap* is the **Photomap** to receive the data. *Encode* is the **EncodeTechnique** by which the image is to be compressed or formatted. *Encode-params* is the list of additional parameters required by *encode*.

Photomap will inherit the attributes of *src* and be populated with its data. The previous attributes and data of *photomap* are overwritten when the **Photoflo** completes.

Error	Cause
FloAlloc	insufficient resources
FloSource	invalid *src*
FloPhotomap	invalid *photomap*
FloTechnique	invalid *encode* technique or *encode-params*

Depending on *encode*, the export process may cause the data to be decoded, encoded, or otherwise reformatted by the server. If **ExportPhotomap** is fed directly from an **ImportClientPhoto** or **ImportPhotomap** element, and the attributes of the raw data that are given to the import element match the requirements of the *encode* technique, the raw data can bypass the normal decode function of the import element and be forwarded directly to **ExportPhotomap** (this is an implementation-specific optimization).

If *encode* is ***ServerChoice***, the server is free to choose an *encode* technique for the exported data. An optional "hint" can be provided to help the server make its choice, but the server can ignore the hint. *PreferTime* hints that retrieval performance is the desired metric, whereas *PreferSpace* indicates that frugal use of storage space is more important. If **ExportPhotomap** is receiving data from an adjacent upstream import element, ***ServerChoice*** may choose to pass the import element's input data directly to the **Photomap**, otherwise a lossless technique will be chosen. The actual technique chosen by the server can be obtained using **QueryPhotomap** after the **Photoflo** completes.

If ***TripleBand*** data are given to an *encode* technique that will interleave the output data ***BandByPixel***, the dimensions of each *src* band must match.

ExportROI

XieFloExportROI

src	XieTypPhototag
roi	XieTypROI

Errors: ***FloAlloc, FloSource, FloROI***
Events: none

ExportROI allows data imported from an **ImportROI** or **ImportClientROI** element to be saved in an existing **ROI**.

Src is the **Phototag** of the element supplying a *list-of-rectangles*. *Roi* is the **ROI** to receive the data.

Roi will be populated with new data. The previous data of *roi* are overwritten after the **Photoflo** completes.

Error	Cause
FloAlloc	insufficient resources
FloSource	invalid *src*
FloROI	invalid *roi*