6

Import Elements

Overview

Element categories

Import elements are used to bring external image data into a **Photoflo**. Import elements have no **Phototag** sources and have one output connection. There are two types of import elements:

- **ImportResource** Elements obtain data from a server resource (`DRAWABLE`, **LUT**, **Photomap**, or **ROI**).
- **ImportClient** Elements require data from the client. This data is transported to the server via the **PutClientData** protocol request.

Multi-source images

Multiple source images (for example, tiled images, multi-client transport, and so on) require a separate import element to describe each segment of the image. The composite image can then be assembled using process elements (for example, **BandCombine**, **Blend**, and **PasteUp**).

Events generated

Certain import elements can generate events to notify the client about abnormal behavior. Two such events are defined:

- **DecodeNotify** Notifies the client when anomalies are encountered while decoding a compressed image. It can also be returned if less data are sent than are required to complete an uncompressed image.
- **ImportObscured** Notifies the client that one or more regions to be imported from a `WINDOW` are obscured and unavailable from `BACKINGSTORE`.

Import from Client

`ImportClient` elements require data from the client. Element parameters fully specify all the attributes of the client data and describe the format or organization the data will have as it is transported through the protocol stream. Data is sent to the `ImportClient` element via the **PutClientData** protocol request after the **Photoflo** becomes ***Active***.

ImportClientLUT

XieFloImportClientLUT

class: XieTypDataClass
band-order: XieTypOrientation
length: XieTypTripletofCARD32
levels: XieTypLevels

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

ImportClientLUT accepts lookup table data from the protocol stream. The transport of data through the protocol stream is accomplished using **PutClientData**. These data are accepted by the **Point**, **ExportLUT**, and **ExportClientLUT** elements.

Class indicates the number of lookup arrays to expect. *Length* specifies the number of entries per array. *Levels* is the number of quantization levels represented per array. *Band-order* declares the order of ***TripleBand*** arrays, or the order in which pixels from a ***TripleBand*** image should be combined to form indices for a ***SingleBand*** array[1].

The *length* of each array should match the number of source image levels that will be remapped through the array. When a ***TripleBand*** image is to be remapped through a ***SingleBand*** array, the *length* of the array should match the product of the source image levels of all three bands[1]. Table 6-1 summarizes the class of output data to expect from **Point**, based on the *class* of **LUT** and the class of image provided to **Point**.

Table 6-1 Relationship between LUT *class* and image *class*

LUT *class*	Source image: ***SingleBand***	Source image: ***TripleBand***
SingleBand (1 array)	Output image: ***SingleBand*** the single image band is re-mapped through the single array Usage examples: achromatic --> achromatic achromatic --> index	Output image: ***SingleBand*** pixels from each image band are combined[1] and then re-mapped through the single array Usage examples: trichromatic --> index trichromatic --> achromatic
TripleBand (3 arrays)	Output image: ***TripleBand*** the single image band is re-mapped through each array separately Usage examples: achromatic --> false-color index----------> trichromatic	Output image: ***TripleBand*** each image band is re-mapped through the corresponding array Usage examples: trichromatic --> trichromatic

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
FloMatch	*levels* is incompatible with the server's depth handling capabilities
FloValue	Invalid *class* or *band-order*

When the client targets data at **ImportClientLUT**, one or more complete array entries must be sent per protocol request (see **PutClientData**).

[1] For ***SingleBand*** **LUT**s, **Point** combines ***TripleBand*** *src* pixel values to provide a compact array indexing scheme. The algorithm is presented in Figure 7-4.

ImportClientPhoto

XieFloImportClientPhoto

notify: BOOL
class: XieTypDataClass
width: XieTypTripletofCARD32
height: XieTypTripletofCARD32
levels: XieTypLevels
decode: XieTypDecodeTechnique
decode-params: <technique-dependent>

Errors: ***FloAlloc, FloMatch, FloValue, FloTechnique***
Events: DecodeNotify

Attribute	Value
class	*class* of imported image
type	***Constrained***
width	*width* of imported image (in pixels)
height	*height* of imported image (in pixels)
levels	*levels* of imported image

ImportClientPhoto accepts image data from the protocol stream. This data may be processed for display or used as **ProcessDomain** data. The attributes and organization of the expected data stream are fully specified by the parameters. The actual transport of image data through the protocol stream is requested using the **PutClientData** protocol request.

Notify enables **DecodeNotify** events to be sent if anomalies are encountered while interpreting the imported image data (see **DecodeNotify** and **DecodeTechnique** descriptions). *Class* specifies the ***DataClass*** of the data being imported. *Width* and *height* are the per-band image dimensions in pixels. *Levels* is the number of quantization levels per band. *Decode* is the **DecodeTechnique** that will be required to decompress the image. *Decode-params* is the list of additional parameters required by *decode*.

Only ***Constrained*** data can be sent through the protocol stream; therefore, *levels* must be valid.

Error	Cause
FloAlloc	Insufficient resources
FloMatch	*levels* is incompatible with the server's depth handling capabilities
FloValue	Invalid *width*, *height*, *levels* (zero) Invalid *class*
FloTechnique	Invalid *decode* technique or *decode-params*

If the imported client data is compressed, **ImportClientPhoto** is generally responsible for decoding the data prior to forwarding the data to downstream elements. An exception to this would be when an export element is a direct recipient of the data and the element desires data that are encoded using the same technique as *decode*.

If individual bands of ***TripleBand*** data have different widths or heights (for example, down-sampled YCbCr data), it is generally the client's responsibility to make inter-band dimensions match to render the image (for example, use **Geometry** with appropriate *band-mask* and *sample* technique). The ***JPEGBaseline*** technique that is described in Appendix A, includes a Boolean flag that allows an image that is interleaved ***BandByPixel*** to be up-sampled as part of the decode function.

All data and a *final* indication must be sent for each band before the **Photoflo** can leave the ***Active*** state. The input data are sent by the client using the **PutClientData** protocol request. If a subset of client bands are desired, the bands should be imported as individual ***SingleBand*** images.

ImportClientROI

XieFloImportClientROI

rectangles: CARD32

Errors: ***FloAlloc***
Events: none

ImportClientROI accepts a *list-of-rectangles* from the protocol stream. These data can be used as input to a **ProcessDomain** or an **ExportROI** or **ExportClientROI** element. The actual transport of data through the protocol stream is accomplished using the **PutClientData** protocol request (the *band-number* parameter of **PutClientData** is ignored).

Rectangles specifies the number of rectangles expected.

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

Error	Cause
FloAlloc	Insufficient resources

When the client targets data at **ImportClientROI**, one or more complete **Rectangles** must be sent per protocol request (see **PutClientData**).

Import from Resource

`ImportResource` elements obtain data from server resources: core `DRAWABLE`s, and XIE **LUT**, **Photomap**, and **ROI** resources.

The attributes and data of XIE resources are bound to `ImportResource` elements when the **Photoflo** becomes ***Active***. The same resource may also be the target of an `ExportResource` element in the same **Photoflo**. However, the association between the resource's identifier (`XID`) and the new attributes and data does not occur until the **Photoflo** successfully completes (that is, leaves the ***Active*** state). (See "**Binding Resources to Photoflos**" in Chapter 4.)

ImportDrawable

XieFloImportDrawable

notify: `BOOL`
drawable: `DRAWABLE`
src-x: `INT16`
src-y: `INT16`
width: `CARD16`
height: `CARD16`
fill: `CARD32`

Errors: ***FloAlloc, FloDrawable, FloValue***
Events: **ImportObscured**

Attribute	Value
class	***SingleBand***
type	***Constrained***
width	*width*
height	*height*
levels	2^{depth} (that is, *drawable* depth)

ImportDrawable allows access to data existing in a `DRAWABLE`. This data may be processed for display or, if *drawable* is a `BITMAP`, used as **ProcessDomain** data.

Notify enables **ImportObscured** events to be sent if data for one or more regions of a `WINDOW` are obscured and unavailable from `BACKINGSTORE` (see **ImportObscured**). *Drawable* is the `DRAWABLE` resource supplying the data. *Src-x*, *src-y*, *width*, and *height* specify the region of data to be imported from *drawable*. *Fill* is the `COLORMAP` index to use for all regions that are obscured.

Error	Cause
FloAlloc	Insufficient resources
FloDrawable	Invalid *drawable*
FloValue	Invalid region (*src-x, src-y, width, height*)

The data are imported as *`ZPixmap`* format `COLORMAP` indices (that is, ***SingleBand*** index data). Index data are appropriate for only a few process elements (for example, where a nearest-neighbor technique is used). **ConvertFromIndex** can be used to convert index data prior to feeding it to other process elements. **Compare** can also be used to produce bitonal data from index data.

ImportDrawablePlane

XieFloImportDrawablePlane

notify: `BOOL`
drawable: `DRAWABLE`
src-x: `INT16`
src-y: `INT16`
width: `CARD16`
height: `CARD16`
fill: `CARD32`
bit-plane: `CARD32`

Errors: ***FloAlloc, FloDrawable, FloValue***
Events: ImportObscured

Attribute	Value
class	***SingleBand***
type	***Constrained***
width	*width*
height	*height*
levels	2

ImportDrawablePlane allows access to a single plane of data existing in a `DRAWABLE`. This data may be processed for display or used as **ProcessDomain** data.

Notify enables **ImportObscured** events to be sent if data for one or more regions of a `WINDOW` are obscured and unavailable from `BACKINGSTORE` (see **ImportObscured**). *Drawable* is the `DRAWABLE` resource supplying the data. *Src-x*, *src-y*, *width*, and *height* specify the region of data to be imported from *drawable*. *Fill* is the value to substitute for all regions that are obscured. *Bit-plane* specifies the plane to be imported from *drawable*.

Bit-plane must have exactly one bit set to one (1), and the value of *bit-plane* must be less than 2^n, where n is the depth of *drawable*. This single bit selects the corresponding bit to be extracted from pixels within *drawable*.

Error	Cause
FloAlloc	Insufficient resources
FloDrawable	Invalid *drawable*
FloValue	Invalid *bit-plane* or region (*src-x, src-y, width, height*)

ImportLUT

XieFloImportLUT

lut: XieTypLUT

Errors: ***FloAlloc, FloAccess, FloLUT***
Events: none

ImportLUT allows access to lookup table data existing in a **LUT** resource. These data are accepted by the **Point**, **ExportLUT**, and **ExportLUT** elements.

Lut is the **LUT** resource supplying the lookup table. Attributes of the lookup table data are inherited from *lut*.

Error	Cause
FloAlloc	Insufficient resources
FloAccess	Attempting to import from *lut* before it has been populated
FloLUT	Invalid *lut*

ImportPhotomap

XieFloImportPhotomap

notify: BOOL
photomap: XieTypPhotomap

Errors: ***FloAlloc, FloAccess, FloPhotomap***
Events: **DecodeNotify**

Attribute	Value
class	Same as *photomap*
type	Same as *photomap*
width	Same as *photomap*
height	Same as *photomap*
levels	Same as *photomap*

ImportPhotomap allows access to image data existing in a **Photomap**. This data may be processed for display or used as **ProcessDomain** data.

Notify enables **DecodeNotify** events to be sent if anomalies are encountered while decoding compressed data (see **DecodeNotify** and the **DecodeTechnique** descriptions). *Photomap* is the **Photomap** resource supplying image data.

Attributes of the source data are inherited from *photomap*.

Error	Cause
FloAlloc	Insufficient resources
FloAccess	Attempting to import from *photomap* before it has been populated
FloPhotomap	Invalid *photomap*

If the data imported from *photomap* are compressed, **ImportPhotomap** is generally responsible for decoding the data prior to forwarding the data to downstream elements. An exception to this would be when an export element is a direct recipient of the data and the data are already encoded using the technique desired by the export element.

If individual bands of ***TripleBand***, ***BandByPlane*** data have different widths or heights (for example, down-sampled YCbCr data), it is generally the client's responsibility to make inter-band dimensions match to render the image (for example, use **Geometry** with appropriate *band-mask* and *sample* technique). If **ImportPhotomap**'s input data are ***TripleBand***, ***BandByPixel***, the dimensions of the output bands will match even if the encoded input data are down-sampled.

ImportROI

XieFloImportROI

roi: XieTypROI

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

ImportROI allows access to a *list-of-rectangles* existing in a **ROI** resource. This data may be referenced by a **ProcessDomain**.

Roi is the **ROI** resource supplying the *list-of-rectangles*.

Error	Cause
FloAlloc	Insufficient resources
FloAccess	Attempting to import from *roi* before it has been populated
FloROI	Invalid *roi*

almost blank